



PY32F07X Series

32-bit ARM® Cortex®-M0+ Microcontroller

PY32F07X Series

32-bit ARM® Cortex®-M0+ Microcontrollers

Reference Manual



Puya Semiconductor (Shanghai) Co., Ltd.

Contents

1. List of abbreviations for register	23
2. System architecture	24
3. Memory and bus architecture	25
3.1. System architecture.....	25
3.2. Memory structure.....	26
3.2.1. Memory Structure Introduction	26
3.3. Embedded SRAM	29
3.4. Boot mode	30
3.4.1. Memory physical image.....	30
3.4.2. Embedded bootstrap program.....	30
4. Embedded Flash.....	31
4.1. Flash memory main features	31
4.2. Introduction to Flash memory functions	31
4.2.1. Flash memory architecture	31
4.2.2. Flash read operations and access latency	31
4.2.3. Flash Write and Erase Operations	32
4.3. Flash Option Bytes	35
4.3.1. Flash option word	35
4.3.2. Flash Option Byte Write	38
4.4. Flash configuration byte	39
4.4.1. HSI_TRIMMING_FOR_USER.....	41
4.4.2. Calibration values for temperature sensors	42
4.4.3. HSI_4M/8M/16M/22.12M/24M_EPPARA0.....	42
4.4.4. HSI_4M/8M/16M/22.12M/24M_EPPARA1.....	42
4.4.5. HSI_4M/8M/16M/22.12M/24M_EPPARA2.....	43
4.4.6. HSI_4M/8M/16M/22.12M/24M_EPPARA3.....	43
4.4.7. HSI_4M/8M/16M/22.12M/24M_EPPARA4.....	43
4.5. Flash memory protection	44
4.5.1. Flash memory read protection.....	44
4.5.2. Flash write protection	45
4.5.3. Option Byte Write Protect.....	45
4.6. Flash memory interruption	45
4.7. Flash Register Description	46
4.7.1. Flash Access Control Register (FLASH_ACR)	46
4.7.2. Flash Key Register (FLASH_KEYR)	47
4.7.3. Flash Option Key Register (FLASH_OPTKEYR)	47
4.7.4. Flash Status Register (FLASH_SR)	48
4.7.5. Flash Control Register (FLASH_CR)	48

4.7.6.	Flash Options Register (FLASH_OPTR).....	51
4.7.7.	Flash BORCR Address Register (FLASH_BORCR).....	52
4.7.8.	Flash WRP Address Register (FLASH_WRPR)	52
4.7.9.	Flash Sleep Time Configuration Register (FLASH_STCR)	54
4.7.10.	Flash TS0 register (FLASH_TS0)	54
4.7.11.	Flash TS1 Register (FLASH_TS1).....	55
4.7.12.	Flash TS2P Register (FLASH_TS2P)	56
4.7.13.	Flash TPS3 register (FLASH_TPS3)	56
4.7.14.	Flash TS3 register (FLASH_TS3)	57
4.7.15.	Flash Page Erase (PAGE ERASE) TPE Register (FLASH_PERTPE)	57
4.7.16.	Flash SECTOR/MASS ERASE TPE Register (FLASH_SMERTPE)	58
4.7.17.	Flash PROGRAM TPE register (FLASH_PRGTPE)	58
4.7.18.	Flash PRE-PROGRAM TPE Register (FLASH_PRETPE)	59
5.	Power control.....	60
5.1.	Power supply.....	60
5.1.1.	Power Supply Block Diagram.....	60
5.2.	Voltage regulator.....	61
5.3.	Power monitoring.....	61
5.3.1.	Power-On reset (POR) / Power-down reset (PDR) / Brown-out reset (BOR).....	61
5.3.2.	Programmable Voltage Detector (PVD)	62
6.	Low-power control.....	63
6.1.	Low Power Mode.....	63
6.1.1.	Introduction to Low Power Mode	63
6.1.2.	Low Power Mode Switch	64
6.1.3.	Functions in each operating mode	64
6.2.	Sleep mode	65
6.2.1.	Entering sleep mode	65
6.2.2.	Exit sleep mode	65
6.3.	Stop mode.....	66
6.3.1.	Enter Stop mode.....	66
6.3.2.	Exiting Stop Mode	67
6.4.	Reduced system clock frequency	68
6.5.	Peripheral clock gating	68
6.6.	Power Management Registers.....	68
6.6.1.	Power Control Register 1 (PWR_CR1)	68
6.6.2.	Power Control Register 2 (PWR_CR2)	69
6.6.3.	Power Supply Status Register (PWR_SR).....	70
7.	Reset.....	72
7.1.	Reset source.....	72
7.1.1.	Power Reset.....	72

7.1.2.	System reset.....	72
7.1.3.	NRST pin (External reset)	72
7.1.4.	Watchdog reset	72
7.1.5.	Software reset	73
7.1.6.	Option byte loader reset	73
8.	Clocks	74
8.1.	Clock source	74
8.1.1.	External high-speed clock HSE	74
8.1.2.	Internal high-speed clock HSI	74
8.1.3.	Internal low-speed clock LSI	74
8.1.4.	HSI10M Clock.....	75
8.1.5.	PLL	75
8.1.6.	LSE Clock.....	75
8.2.	The Clock Tree	76
8.3.	Clock Safety System (CSS)	76
8.3.1.	Clock Configuration and Stateful Security.....	76
8.3.2.	Clock source HSE monitoring	77
8.3.3.	Clock source LSE monitoring	77
8.4.	Output Clock Capability	78
8.5.	Reset/Clock Register	78
8.5.1.	Clock Control Register (RCC_CR)	78
8.5.2.	Internal clock source calibration register (RCC_ICSCR)	80
8.5.3.	Clock Configuration Register (RCC_CFGR)	81
8.5.4.	PLL configuration register (RCC_PLLCFGR)	83
8.5.5.	External Clock Source Control Register (RCC_ECSCR).....	84
8.5.6.	Clock interrupt enable register (RCC_CIER)	85
8.5.7.	Clock Interrupt Flag Register (RCC_CIFR).....	86
8.5.8.	Clock Interrupt Clear Register (RCC_CICR).....	88
8.5.9.	I/O Interface Reset Register (RCC_IOPRSTR)	89
8.5.10.	AHB Peripheral Reset Register (RCC_AHBRSTR)	89
8.5.11.	APB Peripheral Reset Register 1 (RCC_APBRSTR1).....	90
8.5.12.	APB Peripheral Reset Register 2 (RCC_APBRSTR2).....	92
8.5.13.	I/O interface clock enable register (RCC_IOPENR).....	94
8.5.14.	AHB Peripheral Clock Enable Register (RCC_AHBENR)	95
8.5.15.	APB Peripheral Clock Enable Register 1 (RCC_APBENR1)	96
8.5.16.	APB Peripheral Clock Enable Register 2 (RCC_APBENR2)	98
8.5.17.	Peripheral Independent Clock Configuration Register (RCC_CCIPR)	100
8.5.18.	RTC Domain Control Register (RCC_BDCR)	101
8.5.19.	Control/Status Register (RCC_CSR)	102
9.	Clock Calibration Controller (CTC)	105

9.1.	Introduction	105
9.2.	CTC Main Characteristics	105
9.3.	CTC Functional Description.....	105
9.3.1.	CTC Block Diagram.....	105
9.3.2.	REF Synchronized Pulse Generator	106
9.3.3.	CTC Calibration Counter	106
9.3.4.	Frequency evaluation and automatic calibration process	107
9.3.5.	Software Programming Guide	108
9.4.	CTC Register	108
9.4.1.	CTC control register 0 (CTC_CTL0).....	108
9.4.2.	CTC control register 1 (CTC_CTL1).....	110
9.4.3.	CTC Status Register (CTC_SR).....	111
9.4.4.	CTC interrupt clear register (CTC_INTC).....	114
10.	General Purpose I/O (GPIO).....	116
10.1.	Introduction	116
10.2.	General Purpose IO Functional Description	116
10.3.	General Purpose IO Functional Description	116
10.3.1.	General Purpose I/O (GPIO)	117
10.3.2.	I/O pin-multiplexing function multiplexing and mapping	117
10.3.3.	I/O Control Registers	118
10.3.4.	I/O Data Register	119
10.3.5.	I/O data processing by bit.....	119
10.3.6.	I/O Multiplexing Function Input/Output Mode Configuration	119
10.3.7.	External interrupt/wakeup line	120
10.3.8.	I/O Input Configuration	120
10.3.9.	I/O Output Configuration	120
10.3.10.	Multiplexing Function Configuration	121
10.3.11.	Analog Configuration	122
10.3.12.	Using the HSE/LSE pins as GPIOs	122
10.4.	GPIO registers	124
10.4.1.	GPIO port mode register (GPIOx_MODER) (x=A, B, C, F)	124
10.4.2.	GPIO port output type register (GPIOx_OTYPER) (x = A, B, C, F)	124
10.4.3.	GPIO port output speed register (GPIOx_OSPEEDR) (x = A, B, C, F)	125
10.4.4.	GPIO port pull-down register (GPIOx_PUPDR) (x = A, B, C, F)	125
10.4.5.	GPIO port input data register (GPIOx_IDR) (x = A, B, C, F)	126
10.4.6.	GPIO port output data register (GPIOx_ODR) (x = A, B, C, F)	126
10.4.7.	GPIO port bit set/reset register (GPIOx_BSRR) (x = A, B, C, F)	127
10.4.8.	GPIO Port Configuration Lock Register (GPIOx_LCKR) (x = A, B, C, F)	127
10.4.9.	GPIO multiplexing function register (low) (GPIOx_AFR1) (x = A, B, C, F)	128
10.4.10.	GPIO multiplexing function register (high) (GPIOx_AFRH) (x = A, B, C, F)	129

10.4.11.	GPIO Port Bit Reset Register (GPIOx_BRR) (x = A, B, C, F)	130
11.	System Configuration Controller (SYSCFG).....	131
11.1.	System Configuration Register	131
11.1.1.	SYSCFG Configuration Register 1 (SYSCFG_CFGR1)	131
11.1.2.	SYSCFG Configuration Register 2 (SYSCFG_CFGR2)	133
11.1.3.	SYSCFG Configuration Register 3 (SYSCFG_CFGR3)	136
11.1.4.	SYSCFG Configuration Register 4 (SYSCFG_CFGR4)	138
11.1.5.	GPIOA filter enable (PA_ENS).....	138
11.1.6.	GPIOB filter enable (PB_ENS).....	139
11.1.7.	GPIOC filter enable (PC_ENS)	139
11.1.8.	GPIOF filter enable (PF_ENS)	140
11.1.9.	I2C configuration register (SYSCFG_EIIC).....	140
12.	Direct Memory Access (DMA).....	142
12.1.	Introduction	142
12.2.	DMA main features.....	142
12.3.	DMA Function Description	143
12.3.1.	DMA processing	143
12.3.2.	Arbiter	143
12.3.3.	DMA channel	144
12.3.4.	Data Transfer Width/Alignment/Size End.....	147
12.3.5.	Error management	149
12.3.6.	DMA interrupt	149
12.3.7.	DMA Peripheral Request Mapping.....	150
12.4.	DMA registers	150
12.4.1.	DMA Interrupt Status Register (DMA_ISR).....	150
12.4.2.	DMA Interrupt Flag Bit Clear Register (DMA_IFCR)	153
12.4.3.	DMA Channel 1 Configuration Register (DMA_CCR1).....	156
12.4.4.	DMA channel 1 data transfer count register (DMA_CNDTR1)	157
12.4.5.	DMA Channel 1 Peripheral Address Register (DMA_CPAR1)	158
12.4.6.	DMA channel 1 memory address register (DMA_CMAR1).....	159
12.4.7.	DMA Channel 2 Configuration Register (DMA_CCR2).....	159
12.4.8.	DMA Channel 2 Data Transfer Count Register (DMA_CNDTR2).....	161
12.4.9.	DMA Channel 2 Peripheral Address Register (DMA_CPAR2)	162
12.4.10.	DMA channel 2 memory address register (DMA_CMAR2)	162
12.4.11.	DMA Channel 3 Configuration Register (DMA_CCR3).....	163
12.4.12.	DMA Channel 3 Data Transfer Count Register (DMA_CNDTR3).....	164
12.4.13.	DMA Channel 3 Peripheral Address Register (DMA_CPAR3)	165
12.4.14.	DMA Channel 3 Memory Address Register (DMA_CMAR3)	166
12.4.15.	DMA Channel 4 Configuration Register (DMA_CCR4).....	166
12.4.16.	DMA channel 4 data transfer count register (DMA_CNDTR4).....	168

12.4.17.	DMA Channel 4 Peripheral Address Register (DMA_CPAR4)	168
12.4.18.	DMA Channel 4 Memory Address Register (DMA_CMAR4)	169
12.4.19.	DMA Channel 5 Configuration Register (DMA_CCR5)	169
12.4.20.	DMA channel 5 transfer count register (DMA_CNDTR5)	171
12.4.21.	DMA Channel 5 Peripheral Address Register (DMA_CPAR5)	171
12.4.22.	DMA Channel 5 Memory Address Register (DMA_CMAR5)	172
12.4.23.	DMA Channel 6 Configuration Register (DMA_CCR6)	172
12.4.24.	DMA Channel 6 Transmit Count Register (DMA_CNDTR6)	174
12.4.25.	DMA Channel 6 Peripheral Address Register (DMA_CPAR6)	175
12.4.26.	DMA Channel 6 Memory Address Register (DMA_CMAR6)	175
12.4.27.	DMA Channel 7 Configuration Register (DMA_CCR7)	176
12.4.28.	DMA channel 7 transfer count register (DMA_CNDTR7)	177
12.4.29.	DMA Channel 7 Peripheral Address Register (DMA_CPAR7)	178
12.4.30.	DMA Channel 7 Memory Address Register (DMA_CMAR7)	178
13.	Interrupts and Events	180
13.1.	Nested Vector Interrupt Controller (NVIC)	180
13.1.1.	Main characteristics	180
13.1.2.	SysTick Calibration Value Registers	180
13.1.3.	Interrupt and Exception Vector	180
13.2.	External Interrupt/Event Controller (EXTI)	181
13.2.1.	EXTI Key Features	181
13.2.2.	EXTI Block Diagram	182
13.2.3.	Interruption management	182
13.2.4.	Functional Description	183
13.2.5.	Hardware Interrupt Selection	183
13.2.6.	Hardware event selection	183
13.2.7.	Software interrupt/event selection	184
13.2.8.	EXTI Selector	184
13.3.	EXTI register	186
13.3.1.	Rising edge trigger select register (EXTI_RTISR)	186
13.3.2.	Falling edge trigger select register (EXTI_FTSR)	188
13.3.3.	Software Interrupt Event Register (EXTI_SWIER)	190
13.3.4.	Pending Register (EXTI_PR)	194
13.3.5.	External Interrupt Select Register 1 (EXTI_EXTICR1)	198
13.3.6.	External Interrupt Select Register 2 (EXTI_EXTICR2)	199
13.3.7.	External Interrupt Select Register 3 (EXTI_EXTICR3)	200
13.3.8.	External Interrupt Select Register 4 (EXTI_EXTICR4)	201
13.3.9.	Interrupt Mask Register (EXTI_IMR)	202
13.3.10.	Event Mask Register (EXTI_EMR)	205
14.	Cyclic Redundancy Check (CRC)	209

14.1.	Introduction	209
14.2.	CRC Key Features	209
14.3.	CRC Functional Description	209
14.3.1.	CRC Block Diagram	209
14.4.	CRC Register	210
14.4.1.	Data register (CRC_DR)	210
14.4.2.	Independent Data Register (CRC_IDR)	210
14.4.3.	Control Register (CRC_CR)	211
15.	Digital/Analog Conversion (DAC)	212
15.1.	DAC Introduction	212
15.2.	DAC Main Features	212
15.3.	DAC Functional Description	213
15.3.1.	Using DAC channels	213
15.3.2.	Using the DAC Output Buffer	213
15.3.3.	DAC Data Format	213
15.3.4.	DAC conversion	214
15.3.5.	DAC Output Voltage	215
15.3.6.	Select DAC Trigger	215
15.3.7.	DMA function	216
15.4.	DAC Register	223
15.4.1.	DAC Control Register (DAC_CR)	223
15.4.2.	DAC Software Trigger Register (DAC_SWTRIGR)	228
15.4.3.	12-bit right-aligned data hold register for DAC channel 1 (DAC_DHR12R1)	228
15.4.4.	12-bit left-aligned data hold register for DAC channel 1 (DAC_DHR12L1)	229
15.4.5.	8-bit right-aligned data hold register for DAC channel 1 (DAC_DHR8R1)	229
15.4.6.	12-bit right-aligned data hold register for DAC channel 2 (DAC_DHR12R2)	230
15.4.7.	12-bit left-aligned data hold register for DAC channel 2 (DAC_DHR12L2)	230
15.4.8.	8-bit right-aligned data hold register for DAC channel 2 (DAC_DHR8R2)	231
15.4.9.	12-bit right-aligned data hold register for dual DACs (DAC_DHR12RD)	231
15.4.10.	12-bit left-aligned data hold register for dual DACs (DAC_DHR12LD)	232
15.4.11.	8-bit right-aligned data hold register for dual DACs (DAC_DHR8RD)	232
15.4.12.	DAC channel 1 data output register (DAC_DOR1)	233
15.4.13.	DAC channel 2 data output register (DAC_DOR2)	233
15.4.14.	DAC Status Register (DAC_SR)	234
16.	Analog/Digital Conversion (ADC)	235
16.1.	Introduction	235
16.2.	ADC Main Characteristics	235
16.3.	ADC Functional Description	236
16.3.1.	ADC Block Diagram	236
16.3.2.	Calibrations	236

16.3.3.	ADC Switch Control.....	237
16.3.4.	ADC Clock	237
16.3.5.	Channel Selection	237
16.3.6.	Programmable sampling time.....	238
16.3.7.	Configurable resolution	238
16.3.8.	Single conversion mode	239
16.3.9.	Continuous Conversion Mode	239
16.3.10.	Scanning mode.....	239
16.3.11.	intermittent switching mode	240
16.3.12.	Injection channel management.....	241
16.3.13.	Stopping conversions in progress (ADSTP)	242
16.4.	Analog Watchdog.....	242
16.5.	Externally Triggered Conversion.....	243
16.6.	Data alignment.....	244
16.7.	Data overload.....	244
16.8.	DMA request	244
16.9.	Temperature sensor and internal reference voltage	244
16.10.	ADC Interrupt.....	246
16.11.	ADC Registers	246
16.11.1.	ADC Status Register (ADC_SR)	246
16.11.2.	ADC Control Register 1 (ADC_CR1).....	247
16.11.3.	ADC Control Register (ADC_CR2).....	251
16.11.4.	ADC Sample Time Register 1 (ADC_SMPR1).....	254
16.11.5.	ADC Sample Time Register 2 (ADC_SMPR2).....	254
16.11.6.	ADC Sample Time Register 3 (ADC_SMPR3).....	255
16.11.7.	ADC injection channel data offset register x (ADC_JOFRx) (x=1..4).....	256
16.11.8.	ADC Watchdog High Threshold Register (ADC_HTR)	256
16.11.9.	ADC Watchdog Low Threshold Register (ADC_LTR).....	257
16.11.10.	ADC Rule Sequence Register 1 (ADC_SQR1)	257
16.11.11.	ADC Rule Sequence Register 2 (ADC_SQR2)	258
16.11.12.	ADC Rule Sequence Register 3 (ADC_SQR3)	259
16.11.13.	ADC Injection Sequence Register (ADC_JSQR)	260
16.11.14.	ADC Injection Data Register x (ADC_JDRx) (x= 1..4)	261
16.11.15.	ADC rule data register (ADC_DR).....	261
16.11.16.	ADC Calibration Configuration and Status Register (ADC_CCSR)	262
17.	Liquid Crystal Controller (LCD).....	265
17.1.	Introduction	265
17.2.	LCD Main Characteristics.....	265
17.3.	LCD Block Diagram.....	266
17.4.	LCD clock.....	266

17.5.	LCD Driver Waveforms	266
17.5.1.	Static drive waveform	267
17.5.2.	1/2DUTY 1/2BIAS Drive Waveforms	267
17.5.3.	1/8DUTY 1/3BIAS Drive Waveforms	267
17.6.	LCD BIAS Generation Circuit	268
17.6.1.	Internal Resistance Mode	268
17.6.2.	External Resistance Mode	269
17.7.	DMA	269
17.8.	Disruptions	269
17.9.	LCD display mode	270
17.9.1.	LCD display mode 1 (MODE = 1)	270
17.9.2.	LCD display mode 0 (MODE = 0)	272
17.10.	LCD Register	275
17.10.1.	Configuration Register 0 (LCD_CR0)	275
17.10.2.	Configuration Register 1 (LCD_CR1)	276
17.10.3.	Interrupt Clear Register (LCD_INTCLR)	277
17.10.4.	Output configuration register (LCD_POEN0)	277
17.10.5.	Output Configuration Register 1 (LCD_POEN1)	278
17.10.6.	LCD_RAM0~7	280
17.10.7.	LCD_RAM8~F	280
18.	Comparator (COMP)	281
18.1.	Introduction	281
18.2.	COMP Key Features	281
18.3.	COMP Functional Description	282
18.3.1.	COMP Block Diagram	282
18.3.2.	COMP pin and internal signals	283
18.3.3.	COMP Reset and Clock	283
18.3.4.	Window comparator	283
18.3.5.	Hysteresis	284
18.3.6.	Power consumption mode	284
18.3.7.	Comparator Filter	284
18.3.8.	COMP interrupt	285
18.4.	COMP register	285
18.4.1.	COMP1 Control and Status Register (COMP1_CSR)	285
18.4.2.	COMP1 Filter Register (COMP1_FR)	287
18.4.3.	COMP2 Control and Status Register (COMP2_CSR)	288
18.4.4.	COMP2 Filter Register (COMP2_FR)	290
18.4.5.	COMP3 Control and Status Register (COMP3_CSR)	291
18.4.6.	COMP3 Filter Register (COMP3_FR)	292
19.	Operational amplifiers (OPA)	294

19.1.	OPA Introduction	294
19.2.	OPA Main Characteristics	294
19.3.	OPA Functional Description	294
19.3.1.	OPA Block Diagram	294
19.4.	OPA Register	295
19.4.1.	OPA output enable register (OPA_CR0)	295
19.4.2.	OPA Control Register (OPA_CR1)	295
20.	Hardware divider (DIV)	296
20.1.	DIV Introduction	296
20.2.	DIV Key Features	296
20.3.	DIV Function Description	296
20.3.1.	DIV operation flow	296
20.4.	DIV register	297
20.4.1.	DIV Divisor Register (DIV_DEND)	297
20.4.2.	DIV Divisor Register (DIV_SOR)	297
20.4.3.	DIV Business Register (DIV_QUOT)	297
20.4.4.	DIV Remainder Register (DIV_REMD)	298
20.4.5.	DIV symbol register (DIV_SIGN)	298
20.4.6.	DIV Status Register (DIV_STAT)	298
21.	Advanced Control Timer (TIM1)	300
21.1.	TIM1 Introduction	300
21.2.	TIM1 Main Features	300
21.3.	TIM1 Function Description	301
21.3.1.	Time base unit (in computing)	301
21.3.2.	Counter Mode	302
21.3.3.	Repetition counter	311
21.3.4.	Clock source	312
21.3.5.	Capture/Compare Channel	314
21.3.6.	Input Capture Mode	316
21.3.7.	PWM Input Mode	317
21.3.8.	Forced Output Mode	318
21.3.9.	Output Comparison Mode	318
21.3.10.	PWM mode	319
21.3.11.	Complementary outputs and deadband insertion	322
21.3.12.	Using the brake function	323
21.3.13.	Clearing the OCxREF signal on an external event	325
21.3.14.	Six-step PWM generation	326
21.3.15.	Single Pulse Mode	327
21.3.16.	Encoder Interface Mode	329
21.3.17.	Timer input heterodyne function	331

21.3.18.	Interfacing with Hall Sensors	331
21.3.19.	Timer and external trigger synchronization	332
21.3.20.	Timer synchronization	335
21.3.21.	Debug mode	335
21.4.	TIM1 Register Description	335
21.4.1.	TIM1 Control Register 1 (TIM1_CR1)	335
21.4.2.	TIM1 Control Register 2 (TIM1_CR2)	338
21.4.3.	TIM1 Slave Mode Control Register (TIM1_SMCR)	340
21.4.4.	TIM1 DMA/Interrupt Enable Register (TIM1_DIER)	343
21.4.5.	TIM1 status register (TIM1_SR)	345
21.4.6.	TIM1 Event Generation Register (TIM1_EGR)	348
21.4.7.	TIM1 Capture/Compare Mode Register 1 (TIM1_CMR1)	350
21.4.8.	TIM1 Capture/Compare Mode Register 2 (TIM1_CMR2)	354
21.4.9.	TIM1 Capture/Compare Enable Register (TIM1_CCER)	356
21.4.10.	TIM1 Counter (TIM1_CNT)	360
21.4.11.	TIM1 prescaler (TIM1_PSC)	360
21.4.12.	TIM1 Auto-Reload Register (TIM1_ARR)	361
21.4.13.	TIM1 Repeat Counter Register (TIM1_RCR)	361
21.4.14.	TIM1 Capture/Compare Register 1 (TIM1_CCR1)	362
21.4.15.	TIM1 Capture/Compare Register 2 (TIM1_CCR2)	363
21.4.16.	TIM1 Capture/Compare Register 3 (TIM1_CCR3)	364
21.4.17.	TIM1 Capture/Compare Register 4 (TIM1_CCR4)	364
21.4.18.	TIM1 Brake and Deadband Register (TIM1_BDTR)	365
21.4.19.	TIM1 DMA Control Register (TIM1_DCR)	368
21.4.20.	TIM1 continuous mode DMA address (TIM1_DMAR)	369
22.	General purpose timer (TIM2/3)	370
22.1.	Introduction to TIM2/TIM3	370
22.2.	TIM2/3 Main Features	370
22.3.	TIM2/3 Functional Description	371
22.3.1.	time base unit (in computing)	371
22.3.2.	Counter Mode	372
22.3.3.	Clock source	380
22.3.4.	Capture/Compare Channel	382
22.3.5.	Input Capture Mode	383
22.3.6.	PWM Input Mode	384
22.3.7.	Forced Output Mode	385
22.3.8.	Output Comparison Mode	385
22.3.9.	PWM mode	386
22.3.10.	Single Pulse Mode	389
22.3.11.	Encoder Interface Mode	390

22.3.12.	Timer input heterodyne function	392
22.3.13.	Synchronization of timers and external triggers	392
22.3.14.	Timer synchronization	395
22.3.15.	Debug mode	399
22.4.	Register description	399
22.4.1.	TIM2/3 Control Register 1 (TIMx_CR1)	400
22.4.2.	TIM2/3 Control Register 2 (TIMx_CR2)	402
22.4.3.	TIM2/3 Slave Mode Control Register (TIMx_SMCR)	403
22.4.4.	TIM2/3 DMA/Interrupt Enable Register (TIMx_DIER)	406
22.4.5.	TIM2/3 status register (TIMx_SR)	408
22.4.6.	TIM2/3 Event Generation Register (TIMx_EGR)	410
22.4.7.	TIM2/3 Capture/Compare Mode Register 1 (TIMx_CMR1)	412
22.4.8.	TIM2/3 Capture/Compare Mode Register 2 (TIMx_CMR2)	416
22.4.9.	TIM2/3 Capture/Compare Enable Register (TIMx_CCER)	418
22.4.10.	TIM2/3 counter (TIMx_CNT).....	420
22.4.11.	TIM2/3 prescaler (TIMx_PSC).....	421
22.4.12.	TIM2/3 Automatic Reload Register (TIMx_ARR)	421
22.4.13.	TIM2/3 Capture/Compare Register 1 (TIMx_CCR1)	422
22.4.14.	TIM2/3 Capture/Compare Register 2 (TIMx_CCR2)	423
22.4.15.	TIM2/3 Capture/Compare Register 3 (TIMx_CCR3)	423
22.4.16.	TIM2/3 Capture/Compare Register 4 (TIMx_CCR4)	424
22.4.17.	TIM2/3 DMA Control Register (TIMx_DCR)	425
22.4.18.	DMA address for TIM2/3 continuous mode (TIMx_DMAR).....	426
23.	Basic Timer (TIM6/7).....	428
23.1.	Introduction to TIM6 and TIM7	428
23.2.	Key features of TIM6 and TIM7	428
23.3.	TIM6 and TIM7 Functional Description	428
23.3.1.	Time base unit (in computing)	428
23.3.2.	Clock source.....	433
23.3.3.	Debug mode.....	433
23.4.	TIM6 and TIM7 registers	434
23.4.1.	TIM6 and TIM7 control register 1 (TIMx_CR1)	434
23.4.2.	TIM6 and TIM7 Control Register 2 (TIMx_CR2)	435
23.4.3.	TIM6 and TIM7 DMA/Interrupt Enable Registers (TIM14_DIER).....	436
23.4.4.	TIM6 and TIM7 Status Registers (TIMx_SR)	437
23.4.5.	TIM6 and TIM7 Event Generation Registers (TIMx_EGR)	437
23.4.6.	TIM6 and TIM7 counters (TIMx_CNT)	438
23.4.7.	TIM6 and TIM7 prescalers (TIMx_PSC)	438
23.4.8.	TIM6 and TIM7 Automatic Reload Registers (TIMx_ARR)	439
24.	General purpose timer (TIM14).....	440

24.1.	TIM14 Introduction	440
24.2.	TIM14 main features.....	440
24.3.	TIM14 Functional Description	441
24.3.1.	Time base unit (in computing)	441
24.3.2.	Clock source.....	445
24.3.3.	Capture/Compare Channel	445
24.3.4.	Input Capture Mode.....	446
24.3.5.	Forced Output Mode	447
24.3.6.	Output Comparison Mode	448
24.3.7.	PWM mode.....	448
24.3.8.	Single Pulse Mode	449
24.3.9.	Timer synchronization	450
24.3.10.	Debug mode	451
24.4.	TIM14 Register	451
24.4.1.	TIM14 Control Register 1 (TIM14_CR1)	451
24.4.2.	TIM14 DMA/Interrupt Enable Register (TIM14_DIER).....	452
24.4.3.	TIM14 Status Register (TIM14_SR)	453
24.4.4.	TIM14 Event Generation Register (TIM14_EGR)	455
24.4.5.	TIM14 Capture/Compare Mode Register 1 (TIM14_CMR1)	455
24.4.6.	TIM14 Capture/Compare Enable Register (TIM14_CCER)	458
24.4.7.	TIM14 Counter (TIM14_CNT)	460
24.4.8.	TIM14 Prescaler (TIM14_PSC)	460
24.4.9.	TIM14 Automatic Reload Register (TIM14_ARR)	461
24.4.10.	TIM14 Capture/Compare Register 1 (TIM14_CCR1).....	461
24.4.11.	TIM14 option register (TIMx_OR).....	462
25.	General-purpose timers (TIM15/16/17)	463
25.1.	Introduction	463
25.2.	TIM15 main features.....	463
25.3.	TIM16_17 main features.....	464
25.4.	TIM15_16_17 functional description	465
25.4.1.	Time base unit	465
25.4.2.	Counter mode.....	466
25.4.3.	Repetition counter	469
25.4.4.	Clock source.....	470
25.4.5.	Capture/Compare Channel	472
25.4.6.	Input Capture mode.....	473
25.4.7.	PWM input mode (TIM15 only).....	474
25.4.8.	Forced output mode	475
25.4.9.	Output compare mode.....	476
25.4.10.	PWM mode.....	477

25.4.11.	Complementary outputs and dead-time insertion.....	478
25.4.12.	Using the brake function.....	479
25.4.13.	One-pulse mode	481
25.5.	TIMx timer and external trigger synchronization (TIM15 only).....	483
25.5.1.	Slave mode: Reset mode	483
25.5.2.	Slave mode: Gated mode.....	484
25.5.3.	Slave mode: Trigger mode	484
25.5.4.	Slave mode: External clock mode 2 + Trigger mode	485
25.5.5.	TIM and external trigger synchronization	486
25.6.	Timer synchronization (TIM15 only).....	489
25.6.1.	Using one timer as a prescaler for another timer	489
25.6.2.	Using one timer to enable another timer	490
25.6.3.	Using one timer to start another timer	491
25.6.4.	Using an external trigger to synchronously start two timers	492
25.6.5.	Debug mode.....	493
25.7.	TIM15 register description	493
25.7.1.	TIM15 control register 1 (TIMx_CR1).....	493
25.7.2.	TIM15 control register 2 (TIMx_CR2).....	495
25.7.3.	TIM15 slave mode control register (TIMx_SMCR).....	497
25.7.4.	TIM15 DMA/Interrupt Enable Register (TIMx_DIER)	499
25.7.5.	TIM15 status register (TIMx_SR)	500
25.7.6.	TIM15 event generation register (TIMx_EGR)	502
25.7.7.	TIM15 Capture/Compare Mode Register 1 (TIMx_CCMR1).....	503
25.7.8.	TIM15 Capture/Compare Enable Register (TIMx_CCER)	507
25.7.9.	TIM15 counter (TIMx_CNT)	511
25.7.10.	TIM15 prescaler (TIMx_PSC).....	511
25.7.11.	TIM15 auto-reload register (TIMx_ARR)	511
25.7.12.	TIM15 repetition counter register (TIMx_RCR)	512
25.7.13.	TIM15 Capture/Compare Register 1 (TIMx_CCR1)	512
25.7.14.	TIM15 Capture/Compare Register 2 (TIMx_CCR2)	513
25.7.15.	TIM15 brake and dead-time register (TIMx_BDTR).....	514
25.7.16.	TIM15 DMA control register (TIMx_DCR)	516
25.7.17.	DMA address for TIM15 continuous mode (TIMx_DMAR).....	518
25.8.	TIM16_17 Register Description.....	518
25.8.1.	TIM16_17 Control Register 1 (TIMx_CR1)	518
25.8.2.	TIM16_17 Control Register 2 (TIMx_CR2)	520
25.8.3.	TIM16_17 DMA/Interrupt Enable Register (TIMx_DIER).....	521
25.8.4.	TIM16_17 Status Register (TIMx_SR)	522
25.8.5.	TIM16_17 Event Generation Register (TIMx_EGR)	524
25.8.6.	TIM16_17 Capture/Compare Mode Register 1 (TIMx_CCMR1).....	525

25.8.7.	TIM16_17 Capture/Compare Enable Register (TIMx_CCER)	528
25.8.8.	TIM16_17 counter (TIMx_CNT)	531
25.8.9.	TIM16_17 prescaler (TIMx_PSC).....	531
25.8.10.	TIM16_17 auto-reload register (TIMx_ARR)	532
25.8.11.	TIM16_17 repetition counter register (TIMx_RCR)	532
25.8.12.	TIM16_17 Capture/Compare Register 1 (TIMx_CCR1)	533
25.8.13.	TIM16_17 Brake and Dead-Time Register (TIMx_BDTR)	533
25.8.14.	TIM16_17 DMA control register (TIMx_DCR)	536
25.8.15.	DMA address for TIM16_17 continuous mode (TIMx_DMAR).....	538
26.	Low-power Timer (LPTIM).....	539
26.1.	Introduction	539
26.2.	LPTIM Key Features	539
26.3.	Low-power Timer (LPTIM) Functional Description	539
26.3.1.	LPTIM Block Diagram	539
26.3.2.	LPTIM Pins and Internal Signals	539
26.3.3.	LPTIM Reset and Clock	540
26.3.4.	Prescaler	540
26.3.5.	Operating mode.....	540
26.3.6.	Register update	541
26.3.7.	Counter mode.....	541
26.3.8.	Counter reset.....	541
26.3.9.	Debug mode	542
26.4.	LPTIM low-power modes	542
26.5.	LPTIM interrupt.....	542
26.6.	LPTIM registers	542
26.6.1.	LPTIM interrupt and status register (LPTIM_ISR)	542
26.6.2.	LPTIM Interrupt Clear Register (LPTIM_ICR)	543
26.6.3.	LPTIM Interrupt Enable Register (LPTIM_IER).....	544
26.6.4.	LPTIM configuration register (LPTIM_CFGR)	544
26.6.5.	LPTIM control register (LPTIM_CR).....	545
26.6.6.	LPTIM auto-reload register (LPTIM_ARR)	546
26.6.7.	LPTIM counter register (LPTIM_CNT)	547
27.	Independent watchdog (IWDG)	548
27.1.	Introduction	548
27.2.	IWDG key features	548
27.3.	IWDG functional description.....	548
27.3.1.	IWDG block diagram	548
27.3.2.	Hardware watchdog	549
27.3.3.	Register protection	549
27.3.4.	Debug mode and STOP mode	549

27.4.	IWDG registers	550
27.4.1.	Key Register (IWDG_KR)	550
27.4.2.	Prescaler register (IWDG_PR)	550
27.4.3.	Reload register (IWDG_RLR)	551
27.4.4.	Status register (IWDG_SR)	552
28.	Window Watchdog (WWDG)	553
28.1.	Introduction	553
28.2.	WWDG Key Features	553
28.3.	WWDG Functional Description	553
28.3.1.	WWDG block diagram	554
28.3.2.	Activate the watchdog	554
28.3.3.	Control the downcounter	554
28.3.4.	Advanced watchdog interrupt function	555
28.3.5.	How to program the watchdog timeout	555
28.3.6.	Debug mode	556
28.4.	WWDG registers	556
28.4.1.	Control register (WWDG_CR)	556
28.4.2.	Configuration register (WWDG_CFR)	556
28.4.3.	Status Register (WWDG_SR)	557
29.	Real-Time Clock (RTC)	558
29.1.	Introduction	558
29.2.	RTC main features	558
29.3.	RTC functional description	558
29.3.1.	Overview	558
29.3.2.	Reset RTC registers	559
29.3.3.	Read RTC registers	559
29.3.4.	Configuring RTC registers	560
29.3.5.	Setting of RTC Flags	561
29.3.6.	RTC Calibration	562
29.4.	RTC Register	562
29.4.1.	RTC Control Register (RTC_CRH)	562
29.4.2.	RTC Control Register (RTC_CRL)	563
29.4.3.	RTC Prescaler Load Register (RTC_PRLH)	565
29.4.4.	RTC Prescaler Divider Register (RTC_PRLL)	566
29.4.5.	RTC Prescaler Divider Factor Register High (RTC_DIVH)	567
29.4.6.	RTC prescaler divider factor register low (RTC_DIVL)	567
29.4.7.	RTC Counter Register High (RTC_CNTH)	567
29.4.8.	RTC counter register low bits (RTC_CNTL)	568
29.4.9.	RTC Alarm Register High (RTC_ALRH)	569
29.4.10.	RTC alarm register high (RTC_ALRL)	569

29.4.11.	RTC clock calibration and output configuration register (BKP_RTCCR)	570
30.	I2C interface	571
30.1.	Introduction	571
30.2.	I2C key features	571
30.3.	I2C functional description	572
30.3.1.	I2C block diagram	572
30.3.2.	Mode selection	572
30.3.3.	I2C initialization	573
30.3.4.	I2C slave mode.....	573
30.3.5.	I2C master mode	576
30.3.6.	Error Status	582
30.3.7.	SDA/SCL Control.....	583
30.3.8.	DMA Request	583
30.3.9.	SMBus	585
30.4.	I2C Interrupt	587
30.5.	I2C registers.....	587
30.5.1.	I2C Control Register 1 (I2C_CR1).....	587
30.5.2.	I2C Control Register 2 (I2C_CR2).....	590
30.5.3.	I2C own address register 1 (I2C_OAR1)	592
30.5.4.	I2C Own Address Register 2 (I2C_OAR2)	593
30.5.5.	I2C Data Register (I2C_DR).....	593
30.5.6.	I2C Status Register (I2C_SR1)	594
30.5.7.	I2C Status Register 2 (I2C_SR2)	598
30.5.8.	I2C Clock Control Register (I2C_CCR)	600
30.5.9.	I2C TRISE Register (I2C_TRISE)	601
31.	Serial Peripheral Interface (SPI)	603
31.1.	Introduction	603
31.2.	SPI main features	603
31.2.1.	SPI main features	603
31.2.2.	I2S main features	604
31.3.	SPI functional description	605
31.3.1.	Overview.....	605
31.3.2.	Single master and single slave communication	605
31.3.3.	Multi-Slave Communication.....	608
31.3.4.	Multi-Master Communication.....	608
31.3.5.	Slave select (NSS) pin management	609
31.3.6.	Communication format	610
31.3.7.	SPI Configuration	611
31.3.8.	SPI enable process	612
31.3.9.	Data transmission and reception process	612

31.3.10.	Status flags	616
31.3.11.	Error Flags	617
31.3.12.	SPI interrupt.....	618
31.3.13.	CRC calculation	618
31.4.	I2S Functional Description	620
31.4.1.	Overview.....	620
31.4.2.	Audio Protocol	621
31.4.3.	Clock Generator	630
31.4.4.	I2S Master Mode	632
31.4.5.	I2S Slave Mode	634
31.4.6.	Error flags	636
31.4.7.	I2S interrupt.....	636
31.4.8.	DMA Function.....	636
31.5.	SPI and I2S Registers	636
31.5.1.	SPI Control Register 1 (SPI_CR1) (Not used in I2S mode)	637
31.5.2.	SPI Control Register 2 (SPI_CR2)	639
31.5.3.	SPI Status Register (SPI_SR)	641
31.5.4.	SPI Data Register (SPI_DR)	642
31.5.5.	SPI CRC polynomial register (SPI_CRCPR)	643
31.5.6.	SPI Rx CRC Register (SPI_RXCR)	643
31.5.7.	SPI Tx CRC Register (SPI_TXCR)	644
31.5.8.	SPI_I2S Configuration Register (SPI_I2S_CFGR)	645
31.5.9.	SPI_I2S Prescaler Register (SPI_I2SPR)	646
32.	Universal Synchronous Asynchronous Receiver Transmitter (USART)	648
32.1.	Introduction	648
32.2.	USART main features	648
32.3.	USART functional description	649
32.3.1.	USART feature description.....	650
32.3.2.	Transmitter	651
32.3.3.	Receiver	654
32.3.4.	Fractional baud rate generation	657
32.3.5.	USART receiver tolerance.....	658
32.3.6.	USART auto-baud rate detection	659
32.3.7.	Multiprocessor communication.....	660
32.3.8.	LIN (Local Interconnect Network) mode.....	662
32.3.9.	USART Synchronous Mode	663
32.3.10.	Single-wire half-duplex communication.....	665
32.3.11.	Smart Card	666
32.3.12.	IrDA SIR ENDEC Functional Block	667
32.3.13.	Continuous communication using DMA	669

32.3.14.	Hardware flow control	671
32.4.	USART interrupt request	672
32.5.	USART Registers	673
32.5.1.	Status Register (USART_SR)	673
32.5.2.	Data Register (USART_DR)	677
32.5.3.	Baud rate register (USART_BRR)	678
32.5.4.	Control register 1 (USART_CR1)	678
32.5.5.	Control register 2 (USART_CR2)	680
32.5.6.	Control Register 3 (USART_CR3)	682
32.5.7.	Guard time and prescaler (USART_GTPR)	684
33.	CAN2.0 controller	686
33.1.	Introduction	686
33.2.	CAN main features	686
33.3.	CAN functional description	687
33.3.1.	Block diagram	687
33.3.2.	Active mode	687
33.3.3.	Baud rate configuration	687
33.3.4.	Transmit Buffer	689
33.3.5.	Receive Buffer	690
33.3.6.	Receive Filter Register Group	690
33.3.7.	LLC Frame Format Definition	691
33.3.8.	Data transmission	693
33.3.9.	Cancel Data Transmission	693
33.3.10.	Data Reception	694
33.3.11.	Error Handling	694
33.3.12.	Bus-off	695
33.3.13.	Arbitration failure position capture	695
33.3.14.	Loopback mode	695
33.3.15.	Silent mode	696
33.3.16.	Software Reset Function	696
33.3.17.	Time-Triggered TTCAN	698
33.3.18.	Interrupt	700
33.4.	Register description	701
33.4.1.	Node configuration register (CAN_TSNCR)	701
33.4.2.	Bit timing configuration register (CAN_ACBTR)	702
33.4.3.	Limitation and Prescaler Configuration Register (CAN_RLSSP)	702
33.4.4.	Status Register (CAN_IFR)	703
33.4.5.	Interrupt Enable Register (CAN_IER)	706
33.4.6.	Transmission Status Register (CAN_TSR)	707
33.4.7.	Global Configuration Register (CAN_MCR)	707

33.4.8.	Error Warning Register (CAN_WECR)	713
33.4.9.	Reference ID Register (CAN_REFMSG)	714
33.4.10.	TTCAN Configuration Register (CAN_TTCR)	715
33.4.11.	TTCAN Trigger Register (CAN_TTTR)	717
33.4.12.	Memory Status Register (CAN_SCMS)	717
33.4.13.	Filter group control register (CAN_ACFCR)	719
33.4.14.	Filter group code register (CAN_ACFC).....	719
33.4.15.	Filter Group Mask Register (CAN_ACFM)	720
33.4.16.	CAN Receive Buffer Register (CAN_RBUF)	720
33.4.17.	CAN Transmit Buffer Register (CAN_TBUF)	720
34.	USB Full-Speed Device Interface (USBD)	721
34.1.	Introduction	721
34.2.	Key Features of USB	721
34.3.	USB Device Block Diagram	721
34.4.	Functional Description	721
34.4.1.	Functional Module Description	722
34.4.2.	System reset and power-on reset	722
34.4.3.	USB reset state	722
34.4.4.	USB suspend/wakeup mode	723
34.4.5.	IN packet (for data transmission)	723
34.4.6.	OUT packets (used for data reception)	724
34.4.7.	Control Transfer.....	724
34.4.8.	Isochronous transfer.....	727
34.4.9.	Bulk transfer	731
34.4.10.	Interrupt transfer	734
34.5.	USB registers.....	735
34.5.1.	USB Control Register (USB_CR)	735
34.5.2.	USB Interrupt Status Register (USB_INTR).....	736
34.5.3.	USB Interrupt Enable Register (USB_INTRE)	737
34.5.4.	USB Frame Register (USB_FRAME)	738
34.5.5.	USB Endpoint 0 Control Register (USB_EP0CSR)	738
34.5.6.	USB IN Endpoint Control Register (USB_INEPxCSR)	739
34.5.7.	USB OUT endpoint control register (USB_OUTEPxCSR)	741
34.5.8.	USB OUT Endpoint Count Register (USB_OUTCOUNT).....	742
34.5.9.	USB FIFO Register (USB_FIFO)	742
35.	Debug Support	744
35.1.	Overview	744
35.2.	Pin distribution and debug port pins	744
35.2.1.	SWD Debug Port	744
35.2.2.	Flexible SW-DP pin assignment.....	745

35.2.3.	Internal pull-up and pull-down on SWD pins	745
35.3.	ID code and locking mechanism	745
35.4.	SWD Debug Port	745
35.4.1.	SWD Protocol Introduction	745
35.4.2.	SWD Protocol Sequence.....	745
35.4.3.	SW-DP state machine (reset, idle states, ID code)	746
35.4.4.	DP and AP read/write accesses.....	746
35.4.5.	SW-DP registers.....	747
35.4.6.	SW-AP register.....	747
35.5.	Core Debug	748
35.6.	BPU Breakpoint Unit (Break Point Unit)	748
35.6.1.	BPU Functionality	748
35.7.	Data Watchpoint DWT (Data Watchpoint).....	748
35.7.1.	DWT Functionality	748
35.7.2.	DWT Program Counter Sample Register	749
35.8.	MCU Debug Module (DBGMCU).....	749
35.8.1.	Debug Support for Low-Power Modes	749
35.8.2.	Supports debugging for timers, watchdogs, CAN, and I2C.	749
35.9.	DBG Register	750
35.9.1.	DBG Device ID Code Register (DBG_IDCODE).....	750
35.9.2.	Debug MCU Configuration Register (DBGMCU_CR)	750
35.9.3.	DBG APB Freeze Register 1 (DBG_APB_FZ1).....	751
35.9.4.	DBG APB Freeze Register 2 (DBG_APB_FZ2).....	753
36.	Version History	755

1. List of abbreviations for register

Abbreviation	Descriptions
Read/write (rw)	Software can read and write to this bit.
Read-only (r)	Software can only read this bit.
Write-only (w)	Software can only write to this bit. Reading this bit returns the reset value.
Read/clear write0 (rc_w0)	Software can read as well as clear this bit by writing 0. Writing '1' has no effect on the bit value.
Read/clear write1 (rc_w1)	Software can read as well as clear this bit by writing 1. Writing '0' has no effect on the bit value.
Read/clear write (rc_w)	Software can read as well as clear this bit by writing register. Writing to this bit has no effect.
Read/clear by read (rc_r)	Software can read this bit. Reading this bit automatically clears it to '0'. Writing this bit has no effect on the bit value.
Read/set by read (rs_r)	The software can read this bit. Reading this bit automatically sets it to 1. Writing this bit does not affect the bit value.
Read/set (rs)	Software can read as well as set this bit to '1'. Writing '0' has no effect on the bit value.
Toggle (t)	Software can toggle this bit by writing '1'. Writing '0' has no effect.
Reserved (Res)	Reserved bit, must be kept at reset value.

2. System architecture

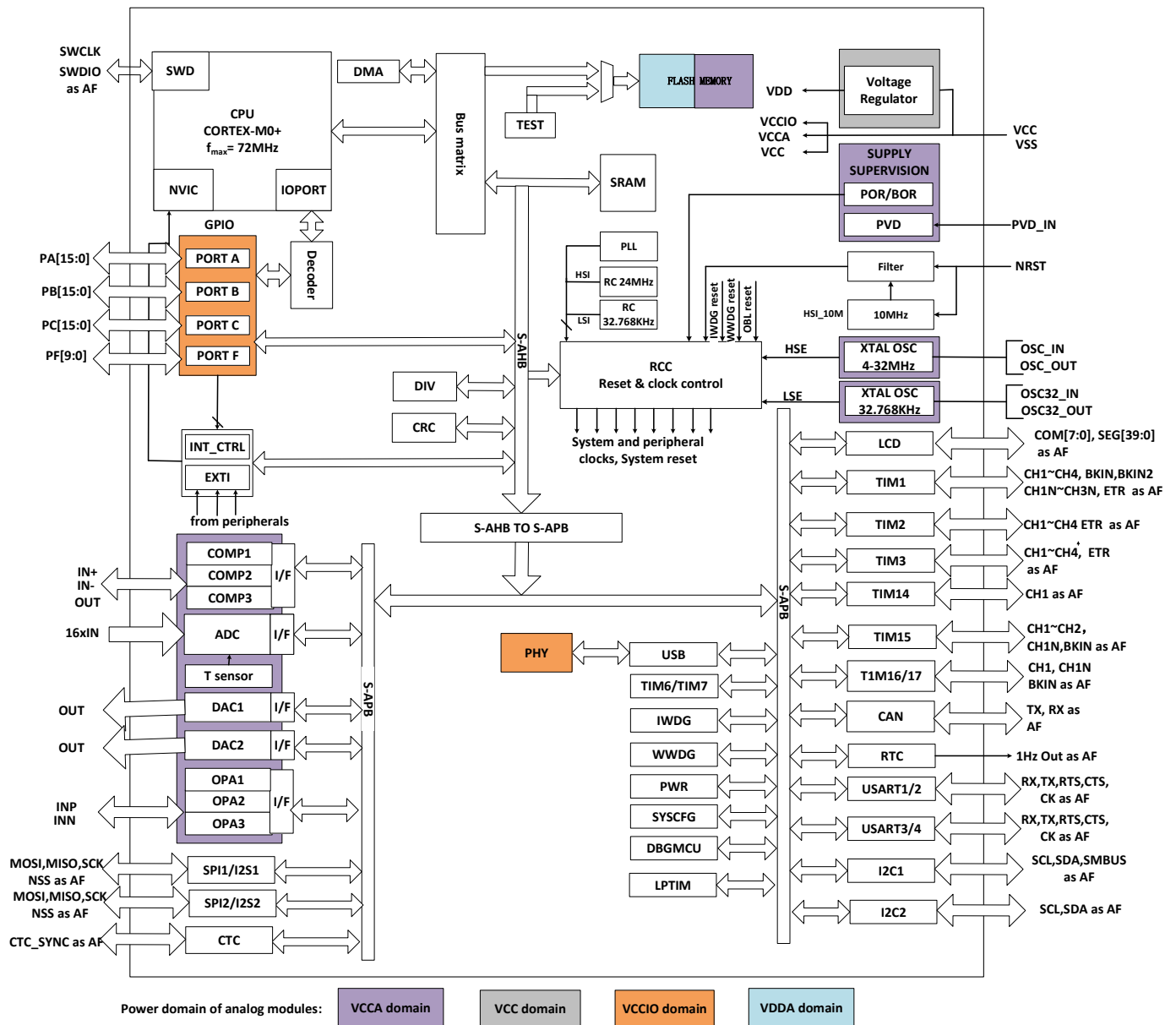


Figure 2-1 System architecture diagram

3. Memory and bus architecture

3.1. System architecture

The system consists of the following components:

- Two Masters
 - Cortex-M0+
 - Universal DMA
- Three Slaves
 - Internal SRAM
 - Internal Flash
 - AHB with AHB-APB Bridge

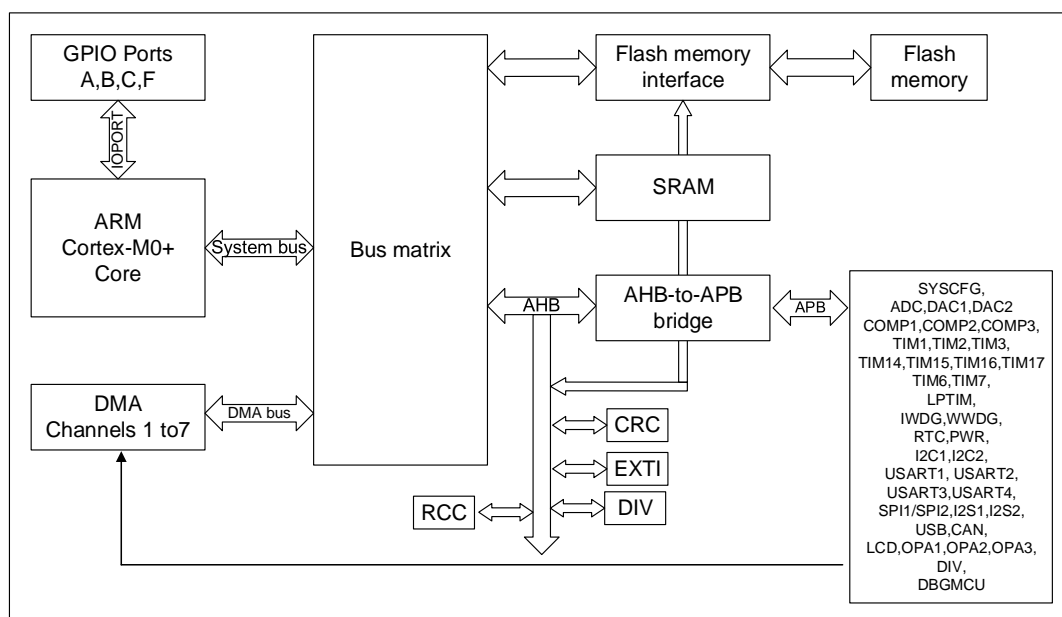


Figure 3-1 System Architecture

- CPU system bus

This bus connects the Cortex-M0+ system bus to the bus matrix.

- DMA bus

This bus connects the DMA's AHB master interface to the bus matrix, which manages peripheral accesses to SRAM, Flash memory, and AHB/APB by the CPU and DMA.

- Bus Matrix

The bus matrix manages access arbitration between the CPU bus and the DMA bus. This arbitration uses the Round Robin algorithm. The bus matrix consists of Master (CPU, DMA) and Slaves (Flash Memory, SRAM and AHB-to-APB bridge).

- AHB-to-APB Bridge (APB)

The AHB-to-APB bridge provides a synchronous connection between the AHB and APB buses, with the bridge's peripheral address mapping reference. [Table 3-2 Peripheral](#) register addresses

3.2. Memory structure

3.2.1. Memory Structure Introduction

Program memory, data memory, registers and IO ports are uniformly addressed in a linear 4 GB space. The address is encoded as a little-end code (the lowest byte in a word is allocated at the lowest address).

The entire addressing space is divided into eight 512 MB Block areas.

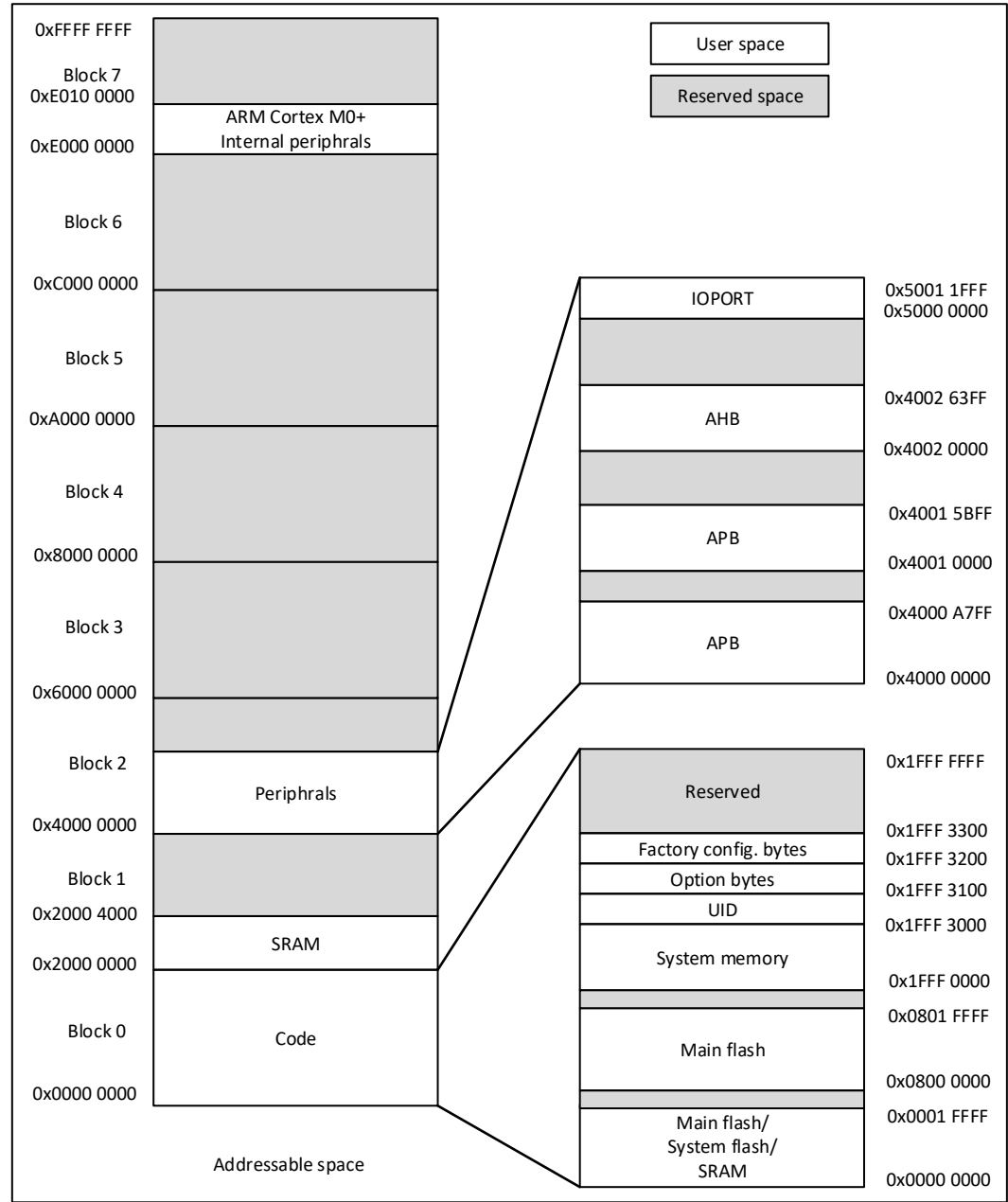


Figure 3-2 Memory mapping

Table 3-1 Memory address

Type	Boundary address	Size	Memory area	Description
SRAM	0x2000 4000-0x3FFF FFFF	-	Reserved ⁽¹⁾	-
	0x2000 0000-0x2000 3FFF	16 KB	SRAM	Up to 16 KB SRAM
Code	0x1FFF 3300-0x1FFF FFFF	-	Reserved	-
	0x1FFF 3200-0x1FFF 32FF	256 Bytes	FT info0 bytes	Factory config.bytes
	0x1FFF 3100-0x1FFF 31FF	256 Bytes	Option bytes	Option bytes information
	0x1FFF 3000-0x1FFF 30FF	256 Bytes	UID bytes	Unique ID
	0x1FFF 0000-0x1FFF 2FFF	12 KB	System memory	Boot loader
	0x0802 0000-0x1FFE FFFF	-	Reserved	-
	0x0800 0000-0x0801 FFFF	128 KB	Main flash memory	-
	0x0002 0000-0x07FF FFFF	-	Reserved	-
	0x0000 0000-0x0001 FFFF	128 KB	Selected based on Boot configuration: 1.Main flash memory 2.System memory 3.SRAM	-

Table 3-2 Peripheral register addresses

computer bus	Address range	Quantitative (science)	Peripherals
	0xE000 0000-0xE00F FFFF	-	M0+
IOPORT	0x5000 1800-0x5FFF FFFF	-	Res
	0x5000 1400-0x5000 17FF	1 KB	GPIOF
	0x5000 1000-0x5000 13FF	-	Res
	0x5000 0C00-0x5000 0FFF	-	Res
	0x5000 0800-0x5000 0BFF	1 KB	GPIOC
	0x5000 0400-0x5000 07FF	1 KB	GPIOB
	0x5000 0000-0x5000 03FF	1 KB	GPIOA
AHB	0x4002 4000-0x4FFF FFFF	-	Res
	0x4002 3c00-0x4002 3fff	-	Res
	0x4002 3800-0x4002 3BFF	1 KB	DIV
	0x4002 3400-0x4002 37FF	-	Res
	0x4002 3000-0x4002 33FF	1 KB	CRC
	0x4002 2400-0x4002 2FFF	-	Res
	0x4002 2000-0x4002 23FF	1 KB	Flash
	0x4002 1c00-0x4002 1fff	-	Res

computer bus	Address range	Quantitative (science)	Peripherals
	0x4002 1800-0x4002 1BFF	1 KB	EXTI
	0x4002 1400-0x4002 17FF	1 KB	Res
	0x4002 1000-0x4002 13FF	1 KB	RCC
	0x4002 0400-0x4002 0FFF	-	Res
	0x4002 0000-0x4002 03FF	1 KB	DMA
APB	0x4001 5c00 - 0x4001 ffff	-	Res
	0x4001 5800 - 0x4001 5BFF	1 KB	DBG
	0x4001 4C00 - 0x4001 57FF	-	Res
	0x4001 4800 - 0x4001 4BFF	1 KB	TIM17
	0x4001 4400 - 0x4001 47FF	1 KB	TIM16
	0x4001 4000 - 0x4001 43FF	1 KB	TIM15
	0x4001 3C00 - 0x4001 3FFF	-	Res
	0x4001 3800 - 0x4001 3BFF	1 KB	USART1
	0x4001 3400 - 0x4001 37FF	1 KB	Res
	0x4001 3000 - 0x4001 33FF	1 KB	SPI1/I2S1
	0x4001 2C00 - 0x4001 2FFF	1 KB	TIM1
	0x4001 2800 - 0x4001 2BFF	-	Res
	0x4001 2400 - 0x4001 27FF	1 KB	ADC
	0x4001 0400 - 0x4001 23FF	-	Res
	0x4001 0300 - 0x4001 03FF	1 KB	OPA
	0x4001 0200 - 0x4001 02FF		COMP
	0x4001 0000 - 0x4001 01FF		SYSCFG
	0x4000 8000- 0x4000 FFFF	-	Res
	0x4000 7C00 - 0x4000 7FFF	1 KB	LPTIM1
	0x4000 7800 - 0x4000 7BFF	-	Res
	0x4000 7400 - 0x4000 77FF	1 KB	DAC
	0x4000 7000 - 0x4000 73FF	1 KB	PWR
	0x4000 6C00 - 0x4000 6FFF	1 KB	CTC
	0x4000 6800 - 0x4000 6BFF	-	Res
	0x4000 6400 - 0x4000 67FF	1 KB	CAN
	0x4000 6000 - 0x4000 63FF	1 KB	USB SRAM
	0x4000 5C00 - 0x4000 5FFF	1 KB	USB
	0x4000 5800 - 0x4000 5BFF	1 KB	I2C2

computer bus	Address range	Quantitative (science)	Peripherals
	0x4000 5400 - 0x4000 57FF	1 KB	I2C1
	0x4000 5000 - 0x4000 53FF	-	Res
	0x4000 4C00 - 0x4000 4FFF	1 KB	USART4
	0x4000 4800 - 0x4000 4BFF	1 KB	USART3
	0x4000 4400 - 0x4000 47FF	1 KB	USART2
	0x4000 3C00 - 0x4000 43FF	2 KB	Res
	0x4000 3800 - 0x4000 3BFF	1 KB	SPI2/I2S2
	0x4000 3400 - 0x4000 37FF	-	Res
	0x4000 3000 - 0x4000 33FF	1 KB	IWDG
	0x4000 2C00 - 0x4000 2FFF	1 KB	WWDG
	0x4000 2800 - 0x4000 2BFF	1 KB	RTC
	0x4000 2400 - 0x4000 27FF	1 KB	LCD
	0x4000 2000 - 0x4000 23FF	1 KB	TIM14
	0x4000 1800 - 0x4000 1FFF	-	Res
	0x4000 1400 - 0x4000 17FF	1 KB	TIM7
	0x4000 1000 - 0x4000 13FF	1 KB	TIM6
	0x4000 0800 - 0x4000 0FFF	-	Res
	0x4000 0400 - 0x4000 07FF	1 KB	TIM3
	0x4000 0000 - 0x4000 03FF	1 KB	TIM2

3.3. Embedded SRAM

Up to 16 KB of SRAM is integrated on-chip. The SRAM can be accessed by means of bytes, half words (16bit) or full words (32bit). Hard faults are generated by software read and write operations to spaces outside the set range.

Flash memory consists of two distinct physical areas:

- Main flash area, 128 KB, which contains application and user data. Software access to space outside the set range generates a Hard fault.
- Information area, 14 KB, which includes the following sections:
 - FT info0 Bytes: 256 Bytes for TS DATA, HSI Clock Re-trimming Data
 - Option bytes: 256 Bytes, used to store the configuration information of chip hardware and software Option
 - UID: 256 Bytes, used to store the UID of the chip
 - System memory: 11.75 KB for the boot loader.

Flash controller implements instruction reading and data access based on AHB protocol, it also implements basic Program/Erase and other operations of Flash through registers.

3.4. Boot mode

With the BOOT0 pin and the boot configuration bit nBOOT1 (stored in the option byte), three different boot modes can be selected, as shown in the table below:

Table 3-3 Boot configuration

Boot mode configuration		Mode
nBOOT1 bit	BOOT0 pin	
X	0	Select Main flash as boot sector
1	1	Select System memory as boot sector
0	1	Select SRAM as boot sector

After this Startup delay, the CPU fetches the value of the top of the stack from address 0x0000 0000 and then begins executing instructions from address 0x0000 0004 in startup memory. Depending on the selected boot mode, Main flash, System memory, or SRAM is accessed as follows:

- Boot from main flash: Main flash is aligned with 0x0000 0000 of the boot memory space, but can still be accessed in its original memory space (0x0800 0000). That is, Flash space can be accessed from address 0x0000 0000 or 0x0800 0000.
- Boot from system memory: System memory is aligned in boot memory space 0x0000 0000, but can still be accessed from its original address space 0x1FFF 0000.
- Boot from SRAM: SRAM is aligned at 0x0000 0000 in the boot memory space, but can still be accessed at address 0x2000 0000.

3.4.1. Memory physical image

After selecting the startup mode, the application software can modify the memory that can be accessed in the program space. This modification is determined by the selection of the MEM_MODE bit in the SYSCFG_CFGR1 register (see the SYSCFG chapter for details).

3.4.2. Embedded bootstrap program

The bootloader is written during the production phase of the chip and is stored in system memory. It is used to re-write the Flash memory using the following serial interface:

- USART1, PA9/PA10
- USB, PA11/PA12

4. Embedded Flash

4.1. Flash memory main features

- Main flash block: up to 128 KB
- Information block: 14 KB
- Page size: 256 Bytes
- Sector size: 8 KB

The main features of the flash control interface circuit are as follows:

- Flash Write and Erase
- Programming Operations for Option Bytes
- read protection
- write-protected

4.2. Introduction to Flash memory functions

4.2.1. Flash memory architecture

Flash memory consists of 64-bit wide memory cells that can be used as program and data storage with a Page size of 256 Bytes and a Sector size of 8 KB.

Functionally, Flash memory is divided into Main flash, which has a maximum capacity of 128 KB, and Information flash, which has a capacity of 14 KB.

Table 4-1 Flash memory structure and boundary addresses

Block	Sector	Page	Base address	Size
Main flash	Sector 0	Page 0-31	0x0800 0000-0x0800 1FFF	8 KB
	Sector 1	Page 32-63	0x0800 2000-0x0800 3FFF	8 KB
	Sector 2	Page 64-95	0x0800 4000-0x0800 5FFF	8 KB

	Sector 14	Page 448-479	0x0801 c000-0x0801 dfff	8 KB
	Sector 15	Page 480-511	0x0801 e000-0x0801 ffff	8 KB
System flash	INFO	Page 0-46	0x1FFF 0000-0x1FFF 2EFF	11.75 KB
Res		Page 47	0x1FFF 2F00-0x1FFF 2FFF	256 Bytes
UID		Page 48	0x1FFF 3000-0x1FFF 30FF	256 Bytes
Option bytes		Page 49	0x1FFF 3100-0x1FFF 31FF	256 Bytes
FT info0		Page 50	0x1FFF 3200-0x1FFF 32FF	256 Bytes
Res		Page 51-55	0x1FFF 3300-0x1FFF 37FF	1280 Bytes

4.2.2. Flash read operations and access latency

Flash can be used as a general-purpose memory space that can be accessed by direct addressing.

The contents of the Flash memory can be read by a specialized read control timing. Finger picking

and data access are performed over the AHB bus. Read operations can be controlled by the Latency bit of the FLASH_ACR register, i.e., reading Flash adds a wait state or not.

FLASH_ACR (LATENCY) bit, when 0, does not increase the wait state for Flash read operations.

When 1, the Flash read operation adds 1 wait state. When it is 2, the Flash read operation adds 3 wait states. This mechanism is specially designed to match the high speed system clock with the relatively low Flash read speed.

4.2.3. Flash Write and Erase Operations

Program operations can be performed on the Flash through circuit programming (ICP, In-circuit programming) or application programming (IAP, In-application programming).

ICP: Used to update the contents of the entire Flash memory, either using the SWD protocol or the system loader (Boot loader) to load the user application into the MCU. ICP provides fast and efficient design iterations, avoiding unnecessary packaging and tube socketing.

IAP: You can use the communication interface supported by the chip to download the data to be Programmed into the Flash. IAP allows the user to Program flash memory again while the application is running. Then, at this point, part of the application program that was previously programmed in using ICP is already in the Flash memory.

If a reset occurs during a flash write and erase operation, the contents of the flash memory are unprotected.

Any operation that reads the flash memory blocks the bus during flash write and erase operations. Once the write or erase operation is finished, the read operation can be performed correctly. This also means that code and data reads cannot be performed while write and erase operations are in progress.

For write and erase operations, HSI must be turned on.

4.2.3.1. Flash memory unlock

After a reset, Flash memory is protected against unintended (e.g., caused by electrical interference) write and erase operations. Writing the FLASH_CR register is not allowed (except for the OBL_LAUNCH bit used as the reload option byte). Each write and erase operation to Flash must enable access to the FLASH_CR register by writing to the FLASH_KEYR register, generating Unlock timing.

The specific steps are as follows:

Step 1: Write KEY1 = 0x4567 0123 to FLASH_KEYR register

Step 2: Write KEY2 = 0xCDEF 89AB to FLASH_KEYR register

Any incorrect timing will latch the FLASH_CR register until the next reset. At incorrect KEY timing, bus errors are detected and Hard fault interrupts are generated. Such errors include a KEY1 mismatch on the first write cycle, or a KEY1 match but a KEY2 mismatch on the second write cycle.

The FLASH_CR register can be locked again by writing the LOCK bit of the FLASH_CR register in software.

Also, the FLASH_CR register cannot be written when the BSY bit of the FLASH_SR register is set. At this point, any attempt to perform a write to this register (FLASH_CR) causes the AHB bus to block until the BSY bit is cleared.

4.2.3.2. Flash Write

The Flash memory performs Program operations on the entire page in 32 bits word units at a time (half word or byte operations generate Hard faults). Program operation starts when the PG bit of the FLASH_CR register is set and the CPU writes 32 bist data to the Flash memory address space. Any write that is not 32 bits will result in a Hard fault interrupt.

If the Flash address space to be Programmed is an area set to protected by the FLASH_WRP register, the Program operation is ignored and the FLASH_SR register WRPERR bit is set. The EOP bit of the FLASH_SR register is set after the Program operation.

The procedure of the specific Flash program is shown below:

1. Check the BSY bit of the FLASH_SR register to determine if there are currently no Flash operations being continued
2. If there is no Flash erase or Program operation in progress, the software reads out the 64 Word of the Page (this step is performed if the Page already has data stored in it, otherwise it is skipped).
3. Write KEY1 and KEY2 sequentially to the FLASH_KEYR register to unprotect the FLASH_CR register
4. Setting the PG bit and the EOPIE bit of the FLASH_CR register
5. Program operation of the 1st to 63rd Word to the destination address (only 32 bits Programs are accepted)
6. Setting PGSTRT of the FLASH_CR register
7. Write the 64th Word
8. Wait for the BSY bit of the FLASH_SR register to be cleared to zero
9. Check the EOP flag bit of the FLASH_SR register (which is set when the Program operation has been successful), then software clears this bit
10. If there are no more Program operations, the software clears the PG bit

When step 7 above is successfully executed, the Program operation is automatically initiated while the BSY bit is set by hardware.

4.2.3.3. Flash erase operation

Flash memory can be erased on a page-by-page basis, or by performing Sector erase and Mass erase.

Page Erase

When a page is protected by WRP, it is not erased, at which time the WRPERR bit is set. When page erasure is to be performed, perform the following steps:

1. Check the FLASH_SR register BSY bit to confirm that there are no Flash operations in progress
2. Write KEY1 and KEY2 sequentially to the FLASH_KEYR register to unprotect the FLASH_CR register

3. Setting the PER bit and the EOPIE bit of the FLASH_CR register
4. Write arbitrary data to this Page (must be 32 bits of data)
5. Wait for the BSY bit to be cleared
6. Check that the EOP flag bit is set
7. Clear the EOP flag

chip erase

Chip Erase is used to erase the entire Main flash. In addition, when WRP is enabled, the chip erase function is disabled, no chip erase operation is generated, and the WRPERR bit is set.

The steps for sheet erasure are as follows:

1. Check the BSY bit to verify that there are no Flash operations in progress
2. Write KEY1 and KEY2 to FLASH_KEYR to unprotect the FLASH_CR register.
3. Set the MER bit and the EOPIE bit of the FLASH_CR register.
4. Write arbitrary data (32 bits of data) to any Main flash space in Flash.
5. Wait for the BSY bit to be cleared
6. Check that the EOP flag bit is set
7. Clear the EOP flag

Sector Erase

Sector Erase is used to perform an erase operation on an 8 KB Main flash. Also, when a sector is protected by WRP, it is not erased, at which point the WRPERR bit is set.

The steps for sector erase are as follows:

- Check the BSY bit to verify that there are no Flash operations in progress
- Write KEY1 and KEY2 sequentially to FLASH_KEYR register to unprotect the FLASH_CR register.
- Setting the SER bit and the EOPIE bit of the FLASH_CR register
- Write arbitrary data to this Sector
- Wait for the BSY bit to be cleared
- Check that the EOP flag bit is set
- Clear the EOP flag

4.2.3.4. Write and Erase Time Configuration

The Program and Erase times of Flash need to be strictly controlled, otherwise the operation will fail. By power-up default, the hardware design sets the time parameter for Program and Erase operations to a parameter of 24 MHz for HSI. When the HSI output frequency is changed, the Flash program and erase time control registers need to be configured correctly according to the table below.

Table 4-2 Write and Erase Time Configuration

Processor register	4 MHz	8 MHz	16 MHz	22.12 MHz	24 MHz
TS0	0x1E	0x3C	0x78	0xA6	0xB4
TS1	0x48	0x90	0x120	0x18F	0x1B0
TS2P	0x1E	0x3C	0x78	0xA6	0xB4
TPS3	0x120	0x240	0x480	0x639	0x6C0

TS3	0x1E	0x3C	0x78	0xA6	0xB4
PERTPE	0x36B0	0x6D60	0XDAC0	0x12E6C	0x14820
SMERTPE	0x36B0	0x6D60	0XDAC0	0x12E6C	0x14820
PRGTPE	0XFA0	0x1F40	0x3E80	0x5668	0x5DC0
PRETPE	0x320	0x640	0xC80	0x1148	0x12C0

4.3. Flash Option Bytes

4.3.1. Flash option word

A portion of the Information area of the on-chip Flash is used as an option byte to store hardware configurations required by the chip or by the user for the application. For example, the watchdog can be selected to be in hardware or software mode.

For data security, option bytes are stored separately in body and reverse code.

Table 4-3 Option Byte Format

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Inverse code of option byte 1								Inverse of option byte 0							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Option byte 1								Option byte 0							

The contents of the option byte can be read from the memory address described in the table Option Byte Structure or from the registers associated with the option byte below:

- Flash Options Register (FLASH_OPTR)
- Flash BORCR Address Register (FLASH_BORCR)
- Flash WRP Address Register (FLASH_WRPR)

Table 4-4 Option Byte Structure

Word address	descriptive
0x1FFF 3100	Flash user option bytes and their complements
0x1FFF 3108	Option bytes for BOR control and its complement
0x1FFF 3110	Res
0x1FFF 3118	Option byte for Flash WRP address and its complement
0x1FFF 3120	Res
0x1FFF 3128	Res
...	Res
...	Res
...	Res
0x1FFF 31F8	Res

4.3.1.1. Option bytes for Flash user options and their complements

Flash memory address: 0x1FFF 3100

Production value: 0x2755 D8AA

After the power-on reset (POR/BOR/OBL_LAUNCH) is released, the corresponding value is read from the option byte area of the Flash information memory and written to the corresponding option bit of this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
~IWDG_STOP	~ nBOOT1	~ NRST_MODE	~ WWDG_SW	~ IWDG_SW	Res	Res	Res	~RDP [7:0]							
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
IWDG_STOP	nBOOT1	NRST_MODE	WWDG_SW	IWDG_SW	Res	Res	Res	RDP [7:0]							
R	R	R	R	R	-	-	-	R	R	R	R	R	R	R	R

Bit	Name	R/W	Function
31	~IWDG_STOP	R	Inverse code of IWDG_STOP
30	~ nBOOT1	R	Inverse code of nBOOT1
29	~ NRST_MODE	R	Inverse code of NRST_MODE
28	~ WWDG_SW	R	Inverse code of WWDG_SW
27	~ IWDG_SW	R	Inverse code of IWDG_SW
26:24	Res	-	Res
23:16	~RDP	R	RDP inverse code
15	IWDG_STOP	R	Setting the IWDG timer operation state in stop mode 0: Freeze timer 1: Normal operation
14	nBOOT1	R	Together with the BOOT PIN, selects the chip boot mode
13	NRST_MODE	R	0: Reset input only 1: GPIO function
12	WWDG_SW	R	0: Hardware Watchdog 1: Software Watchdog
11	IWDG_SW	R	0: Hardware Watchdog 1: Software Watchdog
10: 8	Res	-	Res
7: 0	RDP	R	0xAA: Level 0, invalid read protection Non-0xAA: Level 1, read protection active

4.3.1.2. Option bytes for BOR control and its complement

Flash memory address: 0x1FFF 3108

Production value: 0xFFE0 0000

After the power-on reset (POR/BOR/OBL_LAUNCH) is released, the corresponding value is read from the option byte area of the Flash information memory and written to the corresponding option bit of this register.

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

Bit	Name	R/W	Function
31: 29	~BOR_LEV[2:0]	R	Inverse code of BOR_LEV[2:0]
28: 22	Res	-	Res
21	~BOR_EN	R	Inverse code of BOR_EN
20: 16	Res	-	Res
15: 13	BOR_LEV[2:0]	R	000: BOR rising threshold of 1.8 V, falling threshold of 1.7 V 001: BOR rising threshold of 2.0 V, falling threshold of 1.9 V 010: BOR rising threshold of 2.2 V, falling threshold of 2.1 V 011: BOR rising threshold of 2.4 V, falling threshold of 2.3 V 100: BOR rising threshold of 2.6 V, falling threshold of 2.5 V 101: BOR rising threshold of 2.8 V, falling threshold of 2.7 V 110: BOR rising threshold of 3.0 V, falling threshold of 2.9 V 111: BOR rising threshold of 3.2 V, falling threshold of 3.1 V
12: 6	Res	-	Res
5	BOR_EN	R	BOR Enable 0: BOR not enabled 1: BOR enabled, BOR_LEV active
4: 0	Res	-	Res

4.3.1.3. Option byte for Flash WRP address and its complement

Flash memory address: 0x1FFF 3118

Production value: 0x0000 FFFF

After the power-on reset (POR/BOR/OBL_LAUNCH) is released, the corresponding value is read from the option byte area of the flash information memory and written to the corresponding Option bit of the register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
~WRP [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
WRP [15:0]															
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Bit	Name	R/W	Function
31: 16	~WRP	R	Inverse code for WRP

Bit	Name	R/W	Function
15: 0	WRP	R	0: Sector[y] is protected 1: Sector[y] unprotected y=0~15

4.3.2. Flash Option Byte Write

After reset, the bits in the FLASH_CR register associated with the option byte are write-protected. The OPTLOCK bit in the FLASH_CR register must be cleared before the relevant operation is performed on the option byte.

The following steps are used to unlock this register:

1. Unlock FLASH_CR register write protection via Unlock timing
2. To FLASH_OPTKEYR register, write OPTKEY1=0x0819 2A3B
3. To FLASH_OPTKEYR register, write OPTKEY2=0x4C5D 6E7F

Any incorrect timing will Lock the FLASH_CR register until the next reset. At the wrong KEY timing, the bus error is detected and a Hard fault interrupt is generated.

User options (option bytes of Information flash) can be protected against unwanted erase/Program operations by software writing the OPTLOCK bit of the FLASH_CR register.

If the software sets the Lock bit, the OPTLOCK bit is also automatically set.

4.3.2.1. Modify user option bytes

The write operation (Program) of the option byte is not the same as the operation on the Main flash.

In order to modify the option byte, the following steps are required:

- Clear the OPTLOCK bit using the steps previously described
- Check the BSY bit to make sure there are no Flash operations in progress
- Write desired value to option byte register FLASH_OPTR/FLASH_BORCR/FLASH_WRP (1 to 3 words)
- Setting the OPTSTRT bit
- Write any 32 bit data to Main flash 0x4002 2080 address (triggers a formal write operation)
- Wait for the BSY bit to be cleared
- Waiting for EOP to pull up
- Software Zero EOP

Any change to the option byte, the hardware first erases the entire page corresponding to the option byte, and then writes to the option byte using the value of the FLASH_OPTR, FLASH_BORCR, or FLASH_WRP registers. And, the hardware automatically calculates the corresponding complement and writes the calculated value to the corresponding area of the option byte.

4.3.2.2. Reload option bytes

After the BSY bit is cleared, all new option bytes are written to Flash information memory, but not applied to the chip system. A read operation of the option byte register still returns the value in the

option byte that was last loaded. Only when they (the new values) are loaded do they work on the chip system.

Option bytes are loaded in the following two cases:

1. When the OBL_LAUNCH bit in the FLASH_CR register is set
2. After power-on reset (POR, BOR)

The operation performed by the "load option byte" is to read the option byte in the information memory area and store the read data in the internal option registers (FLASH_OPTR, FLASH_BORCR, and FLASH_WRPR). These internal registers configure the system and can be read by software. Setting the OBL_LAUNCH bit generates a reset so that the option byte can be loaded with the system reset.

Each Option bit has a corresponding complement at its identical double-word address (the next half-word). During option byte loading, the Option bit and its complement are verified, which ensures that the loading was done correctly.

If the positive complement matches, the option byte is copied into the option register.

If the positive complements do not match, the OPTVERR status bit of the FLASH_SR register is set.

Unmatched bits are written to the following values:

- For the User option
- BOR_LEV written as 000 (lowest threshold)
- BOR_EN bit written to 0 (BOR not enabled)
- NRST_MODE bit written to 0 (reset input only)
- The RDP bit is written as 0xff (i.e. level 1)
- The rest of the unmatched values are written as 1
- For the WRP option, the unmatched value is the default value "unprotected".

After a system reset, the contents of the option byte are copied to the following option register (software readable and writable):

- FLASH_OPTR
- FLASH_BORCR
- FLASH_WRPR

These registers are also used to modify the option byte. If these registers are not modified by the user, they reflect the state of the system Option.

4.4. Flash configuration byte

A portion of the interval (1 page in total) of the Information area of the Flash within the chip is used as FT info0.

Page 0 stores information for software to read (positive code only, no reverse code stored):

- HSI Frequency Selection Control Value, and Corresponding Trimming Value
- Configuration parameter values for erasure times corresponding to different frequencies of the HSI

Table 4-5 FT info0 Configuration

Page	Word	Address	Contents
0	0	0x1FFF 3200	Stores the HSI 4 MHz frequency selection control and the corresponding Trimming value
	1	0x1FFF 3208	Stores the HSI 8 MHz frequency selection control and corresponding Trimming value
	2	0x1FFF 3210	Stores the HSI 16 MHz frequency selection control and corresponding Trimming value
	3	0x1FFF 3218	Stores the HSI 22.12 MHz frequency selection control and corresponding Trimming value
	4	0x1FFF 3220	Stores the HSI 24 MHz frequency selection control and corresponding Trimming value
	5	0x1FFF 3228	TS_CAL1 , 30 °C temperature sensor calibration value
	6	0x1FFF 3230	TS_CAL2 , 105 °C temperature sensor calibration value
	7	0x1FFF 3238	Stores the configuration values of the corresponding FLASH_TS0, FLASH_TS1 registers at the HSI 4 MHz frequency
	8	0x1FFF 3240	Stores the configuration values of the corresponding FLASH_TS2P, FLASH_TPS3 registers at the HSI 4 MHz frequency
	9	0x1FFF 3248	Stores the configuration value of the FLASH_PERTPE register corresponding to the HSI 4 MHz frequency
	10	0x1FFF 3250	Stores the configuration value of the FLASH_SMERTPE register corresponding to the HSI 4 MHz frequency
	11	0x1FFF 3258	Stores the configuration values of the FLASH_PRGTPE, FLASH_PRETPE registers corresponding to the HSI 4 MHz frequency
	12	0x1FFF 3260	Stores the configuration values of the corresponding FLASH_TS0, FLASH_TS1 registers at the HSI 8 MHz frequency
	13	0x1FFF 3268	Stores the configuration values of the corresponding FLASH_TS2P, FLASH_TPS3 registers at the HSI 8 MHz frequency
	14	0x1FFF 3270	Stores the configuration value of the FLASH_PERTPE register corresponding to the HSI 8 MHz frequency
	15	0x1FFF 3278	Stores the configuration value of the FLASH_SMERTPE register corresponding to the HSI 8 MHz frequency
	16	0x1FFF 3280	Stores the configuration values of the FLASH_PRGTPE, FLASH_PRETPE registers corresponding to the HSI 8 MHz frequency
	17	0x1FFF 3288	Stores the configuration values of the corresponding FLASH_TS0, FLASH_TS1 registers at the HSI 16 MHz frequency
	18	0x1FFF 3290	Stores the configuration values of the corresponding FLASH_TS2P, FLASH_TPS3 registers at the HSI 16 MHz frequency

19	0x1FFF 3298	Stores the configuration value of the FLASH_PERTPE register corresponding to the HSI 16 MHz frequency
20	0x1FFF 32A0	Stores the configuration value of the FLASH_SMERTPE register corresponding to the HSI 16 MHz frequency
21	0x1FFF 32A8	Stores the configuration values of the FLASH_PRGTPE, FLASH_PRETPE registers corresponding to the HSI 16 MHz frequency
22	0x1FFF 32B0	Stores the configuration values of the corresponding FLASH_TS0, FLASH_TS1 registers at the HSI 22.12 MHz frequency
23	0x1FFF 32B8	Stores the configuration values of the corresponding FLASH_TS2P, FLASH_TPS3 registers at the HSI 22.12 MHz frequency
24	0x1FFF 32C0	Stores the configuration value of the FLASH_PERTPE register corresponding to the HSI 22.12 MHz frequency
25	0x1FFF 32C8	Stores the configuration value of the FLASH_SMERTPE register corresponding to the HSI 22.12 MHz frequency
26	0x1FFF 32D0	Stores the configuration values of the FLASH_PRGTPE, FLASH_PRETPE registers corresponding to the HSI 22.12 MHz frequency
27	0x1FFF 32D8	Stores the configuration values of the corresponding FLASH_TS0, FLASH_TS1 registers at the HSI 24 MHz frequency
28	0x1FFF 32E0	Stores the configuration values of the corresponding FLASH_TS2P, FLASH_TPS3 registers at the HSI 24 MHz frequency
29	0x1FFF 32E8	Stores the configuration value of the FLASH_PERTPE register corresponding to the HSI 24 MHz frequency
30	0x1FFF 32F0	Stores the configuration value of the FLASH_SMERTPE register corresponding to the HSI 24 MHz frequency
31	0x1FFF 32F8	Stores the configuration values of the FLASH_PRGTPE and FLASH_PRETPE registers corresponding to the HSI 24 MHz frequency.

4.4.1. HSI_TRIMMING_FOR_USER

Address: 0x1FFF 3200 ~ 0x1FFF 3220

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

The software needs to read data from this address and write it to the RCC_ICSCR registers corresponding to HSI_FS[2:0] and HSI_TRIM[12:0] for HSI frequency change.

4.4.2. Calibration values for temperature sensors

Address: 0x1FFF 3228 (30 °C), 0x1FFF 3230 (105 °C)

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

The software needs to read data from this address.

4.4.3. HSI_4M/8M/16M/22.12M/24M_EPPARA0

Address: 0x1FFF 3238 (4 MHz), 0x1FFF 3260 (8 MHz), 0x1FFF 3288 (16 MHz), 0x1FFF 32B0 (22.12 MHz), 0x1FFF 32D8 (24 MHz)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	TS1[8: 0]								
-	-	-	-	-	-	-	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
TS3[7: 0]								TS0[7: 0]							
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

The software needs to choose to read data from the corresponding address according to the HSI clock frequency that needs to be set, and then write it to the FLASH_TS0, FLASH_TS1, and FLASH_TS3 registers in order to realize the configuration of the erase and write time required for the corresponding HSI frequency.

4.4.4. HSI_4M/8M/16M/22.12M/24M_EPPARA1

Address: 0x1FFF 3240 (4 MHz), 0x1FFF 3268 (8 MHz), 0x1FFF 3290 (16 MHz), 0x1FFF 32B8 (22.12 MHz), 0x1FFF 32E0 (24 MHz)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	TPS3[10: 0]										
-	-	-	-	-	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
Res	Res	Res	Res	Res	Res	Res	Res	TS2P[7: 0]							
-	-	-	-	-	-	-	-	R	R	R	R	R	R	R	R

The software needs to choose to read data from the corresponding address according to the HSI clock frequency that needs to be set, and then write it to the FLASH_TS2P and FLASH_TPS3 registers in order to realize the configuration of the erase and write time required for the corresponding HSI frequency.

4.4.5. HSI_4M/8M/16M/22.12M/24M_EPPARA2

Address: 0x1FFF 3248 (4 MHz), 0x1FFF 3270 (8 MHz), 0x1FFF 3298 (16 MHz), 0x1FFF 32C0 (22.12 MHz), 0x1FFF 32E8 (24 MHz)

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

The software needs to choose to read the data from the corresponding address according to the HSI clock frequency that needs to be set, and then write it into the FLASH_PERTPE register in order to realize the configuration of the erase and write time required for the corresponding HSI frequency.

4.4.6. HSI_4M/8M/16M/22.12M/24M_EPPARA3

Address: 0x1FFF 3250 (4 MHz), 0x1FFF 3278 (8 MHz), 0x1FFF 32A0 (16 MHz), 0x1FFF 32C8 (22.12 MHz), 0x1FFF 32F0 (24 MHz)

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

The software needs to choose to read data from the corresponding address according to the HSI clock frequency to be set, and then write it into the FLASH_SMERTPE register to realize the configuration of the erase and write time required for the corresponding HSI frequency.

4.4.7. HSI_4M/8M/16M/22.12M/24M_EPPARA4

Address: 0x1FFF 3258 (4 MHz), 0x1FFF 3280 (8 MHz), 0x1FFF 32A8 (16 MHz), 0x1FFF 32D0 (22.12 MHz), 0x1FFF 32F8 (24 MHz)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	PRETPE [11:0]										
-	-	-	-	-	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
PRGTPE [15:0]															
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

The software needs to select the data to be read from the corresponding address according to the HSI clock frequency that needs to be set, and then write it into the FLASH_PRGTPE and FLASH_PRETPE registers in order to realize the configuration of the erase and write time required for the corresponding HSI frequency.

4.5. Flash memory protection

The protection of Flash main memory includes the following mechanisms:

- Read protection (RDP) to prevent access from the outside.
- Write protection (WRP) to prevent unwanted write operations (due to cluttering of the program memory pointer PC). The granularity of write protection is designed to be 8 KB.
- Option byte write protection, specialized unlock design.

4.5.1. Flash memory read protection

The read protection function can be activated by setting the option byte read protection and performing a system reset (POR/BOR or OPL reset) to load a new option byte read protection. RDP protects Flash main memory, option bytes, and SRAM.

If read protection is set while Debug via SWD is still connected, (at this point an access error is made) a power-up reset rather than a system reset is required.

Flash memory is protected when the option byte is read-protected and the complement is correctly present in pairs in the option byte.

Table 4-6 Flash Read-Protect Status

RDP byte value	RDP complemented byte value	Read protection level
0xAA	0x55	Level 0
Any value except the combination of (0xAA and 0x55)		Level 1

Regardless of the protection level, System memory can only be read-accessed and cannot be Programmed or erased.

■ Level 0: No protection

Read, Program and Erase operations are possible for Main flash, and any operation is possible for option bytes.

■ Level 1: Read Protection

Level 1 read protection is in effect when the RDP and its complement in the option byte contain any combination other than (0xAA, 0x55), Level 1 being the default protection level.

- User Mode: Programs executed in user mode (started from Main flash) can perform all operations on Main flash, option bytes.
- Debug mode: booting from SRAM and booting from System memory mode (Boot loader): In Debug mode, or when booting from SRAM or System memory (Boot loader), the Main flash is not accessible. In these modes, a read or write access to the Main flash will generate a bus error, as well as generate a Hard fault interrupt.

When already at Level 1 (any number other than 0xAA), the hardware performs a full chip erase operation on the Main flash if it is to be modified to Level 0 (write 0xAA).

Table 4-7 Relationship of access status to protection level and execution mode

shore	Read protection level	Boot from Main Flash (CPU)			Debugging/Executing from SRAM/ Execute from System memory			DMA		
		user-executed operation								
		phrase marked by pause	write	erase	phrase marked by pause	write	erase	phrase marked by pause	write	erase
System memory	0/1	Yes	No	No	Yes	No	No	No	No	No
		Yes	No	No	Yes	No	No	No	No	No
Option Byte Area	0/1	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No
		Yes	Yes	Yes	Yes	Yes	Yes	No	No	No
FT info0	0/1	Yes	No	No	Yes	No	No	No	No	No
		Yes	No	No	Yes	No	No	No	No	No
UID	0/1	Yes	No	No	Yes	No	No	No	No	No
		Yes	No	No	Yes	No	No	No	No	No

Notes:

- Any modification of Level 1 to Level 0 triggers a hardware full wipe of the Main flash.
- For executing a program from SRAM or System memory, there are two cases: one is Boot from SRAM or System memory; the other is Boot from another memory and the program jumps to SRAM or System memory.

4.5.2. Flash write protection

Flash can be set to be write-protected against unwanted write operations. Defines the control granularity per Bit of the WRP register to be a write-protected (WRP) area of 8 KB, i.e., 1 sector size. See the description of the WRP registers for details.

When the area being WRP is activated, no Erase or Program operation is allowed. Accordingly, the Mass erase function does not work even if only one area is set to write-protected.

In addition, if an Erase or Program operation is attempted on an area set to write-protect, the Write-Protect Error Flag (WRPERR) in the FLASH_SR register is set.

Note: Write-protection works only for Main flash, not for System memory.

4.5.3. Option Byte Write Protect

By default, option bytes are readable and write-protected. To obtain an erase or Program access to the option byte, the correct sequence needs to be written to the OPTKEYR register.

4.6. Flash memory interruption

Table 4-8 Flash Interrupt Requests

disruption event	event marker	Time marker/interrupt clearing method	Control Bit Enable
End of operation	EOP	Write EOP=1	EOPIE
Write protection	WRPERR	Write WRPERR=1	ERRIE

Note: The following events do not have separate interrupt identifiers, but do generate Hard faults:

1. Sequence error in the FLASH_CR register of unlocked Flash memory
2. Unlock Flash option byte write operation sequence error
3. Flash program operation without 32-bit data alignment
4. Flash erase (including page erase, sector erase, and mass erase) operations are not 32-bit data aligned.
5. Writes to option byte registers are not 32-bit data aligned

4.7. Flash Register Description

4.7.1. Flash Access Control Register (FLASH_ACR)

Address offset: 0x00

Reset value: 0x0000 0700

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

Bit	Name	R/W	Reset Value	Function
31: 2	Res	-	-	Res
1: 0	LATENCY [1:0]	RW	0	<p>The wait state corresponding to a Flash read operation:</p> <p>00: Flash read operation without wait state (SYSCLK<=24 MHz)</p> <p>01: Flash read operation has 1 wait state, i.e., each flash read requires two system clock cycles (24 MHz <SYSCLK<=48 MHz)</p> <p>10: Flash read operation has 3 wait states, i.e., each flash read requires four system clock cycles (48 MHz <SYSCLK<=72 MHz)</p> <p>11: Res</p> <p>Note: When operating the LATENCY register, the rest of the high bits of FLASH_ACR are not allowed to be written by 1</p>

4.7.2. Flash Key Register (FLASH_KEYR)

Address offset: 0x08

Reset value: 0x0000 0000

All register bits are Write-Only and return 0 on readout.

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

Bit	Name	R/W	Reset Value	Function
31: 0	KEY [31:0]	W	0	<p>The following values must be written consecutively to unlock the FLASH_CR register and enable the Flash program/erase operation</p> <p>KEY1: 0x4567 0123</p> <p>KEY2: 0xCDEF 89AB</p>

4.7.3. Flash Option Key Register (FLASH_OPTKEYR)

Address offset: 0x0C

Reset value: 0x0000 0000

All register bits are Write-Only and return 0 on readout.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OPTKEY [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
OPTKEY [15:0]															
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W

Bit	Name	R/W	Reset Value	Function
31: 0	OPTKEY [31:0]	W	0	<p>The following values must be written consecutively to unlock the flash's option register and enable program/erase operations on the option byte</p> <p>KEY1: 0x0819 2A3B</p> <p>KEY2: 0x4C5D 6E7F</p>

4.7.4. Flash Status Register (FLASH_SR)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	BSY
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	R
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OPTV ERR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	WRP ERR	Res	Res	Res	EOP
RC_W1	-	-	-	-	-	-	-	-	-	-	RC_W1	-	-	-	RC_W1

Bit	Name	R/W	Reset Value	Function
31: 17	Res	-	-	Res
16	BSY	R	0	<p>Busy Bit</p> <p>This bit indicates that Flash operation is in progress. This bit is set by hardware at the beginning of a Flash operation and cleared by hardware when the operation is completed or an error is generated.</p>
15	OPTVERR	RC_W1	0	<p>Option and trimming bit loading errors</p> <p>Hardware sets this bit when the OPTION and TRIMMING bit and their inverses do not match. Loading mismatched option bytes is forced to a safe value.</p> <p>The software writes 1 and clears.</p>
14: 5	Res	-	-	Res
4	WRPERR	RC_W1	0	<p>write-protect error</p> <p>Hardware sets this bit when the address to be program/erased is in the write-protected Flash area (WRP).</p> <p>The software writes 1 to clear the bit.</p>
3: 1	Res	-	-	Res
0	EOP	RC_W1	0	<p>When the program/erase operation of Flash is successfully completed, the hardware is set. This bit is only set if the EOPIE bit of the FLASH_CR register is enabled.</p> <p>The software writes 1 to clear the bit.</p>

4.7.5. Flash Control Register (FLASH_CR)

Address offset: 0x14

Reset value: 0xC000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	OPT LOCK	Res		OBL_LAUNCH	Res	ERR IE	EOP IE	Res				PGSTRT	Res	OPT STRT	Res
RS	RS	-		RC_W1		RW	RW	-				RW	-	RW	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res		SER	Res			Res				Res	MER	PER	PG
-	-	-		RW	-			-				-	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31	Lock	RS	1	<p>FLASH_CR Lock bit.</p> <p>The software can only set this bit. When set, the FLASH_CR register is Locked. When the unlock timing is successfully given, this bit is cleared by hardware, unlocking the FLASH_CR register.</p> <p>The software will set this bit after the program/erase operation is completed.</p> <p>When unsuccessful unlock timing is given, this bit remains set until the next system reset.</p>
30	OPTLOCK	RS	1	<p>Option Byte Lock bit.</p> <p>The software can only set this bit. When set, the bits in the FLASH_CR register associated with the option byte are Locked. When the unlock timing is successfully given, this bit is cleared by hardware, unlocking the FLASH_CR register.</p> <p>The software will set this bit after the program/erase operation is completed.</p> <p>When unsuccessful unlock timing is given, this bit remains set until the next system reset.</p>
29: 28	Res	-	-	Res
27	OBL_LAUNCH	RC_W1	0	<p>Force option byte reload.</p> <p>When set, this bit forces the system to perform a reload of the option byte. This bit is cleared by hardware only when the option byte load is completed. If the OPTLOCK bit is set, this bit cannot be written.</p> <p>0: Option byte reloading complete</p> <p>1: An option byte reload request is generated and the system generates a reset for option byte reloading.</p>
26	Res	-	-	Res

Bit	Name	R/W	Reset Value	Function
25	ERRIE	RW	0	The Error Interrupt Enable Bit, when the WRPERR bit of the FLASH_SR register is set, generates an interrupt request if this bit is enabled. 0: No interrupt generation 1: With interrupt generation
24	EOPIE	RW	0	End-of-operation interrupt enable When the EOP bit of the FLASH_SR register is set, this bit enables interrupt generation. 0: EOP interrupt off 1: EOP interrupt enable
23: 20	Res	-	-	Res
19	PGSTRT	RW	0	Startup bit for program operations in Flash main memory. This bit initiates program operation of the Flash main memory, is set by software, and is cleared by hardware after the BSY bit in the FLASH_SR register is cleared.
18	Res	-	-	Res
17	OPTSTRT	RW	0	Flash option byte modified startup bits This bit initiates the modification of the option byte. Software set and hardware cleared this bit after the BSY bit in the FLASH_SR register is cleared. Note: When the flash option byte is modified, the hardware automatically erases the entire 256 Bytes of the page and then performs the program operation, which also includes automatic complement writing.
16: 12	Res	-	-	Res
11	SER	RW	0	8 KB sector erase operation 0: Sector erase operation for Flash not selected 1: Selecting the sector erase operation for Flash Notes: Sector erase does not work on Flash information memory. Sector Erase does not work for areas set to WRP.
10: 3	Res	-	-	Res
2	MER	RW	0	Mass erase operation 0: Flash chip erase operation is not selected 1: Selecting the Flash's Chip Erase Operation Notes: Mass erase does not work on Flash information memory. Slice erase does not work when there is a WRP setting

Bit	Name	R/W	Reset Value	Function
1	PER	RW	0	Page Erase Operation 0: Page erase operation for Flash not selected 1: Selecting Flash's Page Erase Operation
0	PG	RW	0	Program operation 0: Flash program operation not selected 1: Selecting the Program Operation for Flash

4.7.6. Flash Options Register (FLASH_OPTR)

Address offset: 0x20

Reset value: 0x0000 D8AA.

After the power-on reset (POR/BOR/OBL_LAUNCH) is released, the corresponding value is read from the option byte area of the flash information memory and written to the corresponding option bit of this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IWDG_STOP	nBOOT1	NRST_MODE	WWDG_SW	IWDG_SW	Res			RDP [7:0]							
RW	RW	RW	RW	RW	-			RW							

Bit	Name	R/W	Reset Value	Function
31: 16	Res	-	-	Res
15	IWDG_STOP	RW	1	Setting the timer running state of IWDG in STOP mode 0: Freeze timer 1: Normal operation
14	nBOOT1	RW	1	Together with the BOOT PIN, selects the chip boot mode
13	NRST_MODE	RW	0	0: Reset input only 1: GPIO: GPIO function
12	WWDG_SW	RW	1	0: Hardware Watchdog 1: Software Watchdog
11	IWDG_SW	RW	1	0: Hardware Watchdog 1: Software Watchdog
10: 8	Res	-	-	Res
7: 0	RDP	RW	0xAA	0xAA: level 0, invalid read protection Non-0xAA: level 1, read protection active

4.7.7. Flash BORCR Address Register (FLASH_BORCR)

Address offset: 0x24

Reset value: 0x0000 0000.

After the power-on reset (POR/BOR/OBL_LAUNCH) is released, the corresponding value is read from the option byte area of the flash information memory and written to the corresponding option bit of the register.

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

Bit	Name	R/W	Reset Value	Function
31: 16	Res	-	-	Res
15: 13	BOR_LEV[2:0]	RW	0	000: BOR rising threshold 1.8 V, falling threshold bit 1.7 V 001: BOR rising threshold 2.0 V, falling threshold bit 1.9 V 010: BOR rising threshold 2.2 V, falling threshold bit 2.1 V 011: BOR rising threshold 2.4 V, falling threshold bit 2.3 V 100: BOR rising threshold 2.6 V, falling threshold bit 2.5 V 101: BOR rising threshold 2.8 V, falling threshold bit 2.7 V 110: BOR rising threshold 3.0 V, falling threshold bit 2.9 V 111: BOR rising threshold 3.2 V, falling threshold bit 3.1 V
12: 6	Res	-	-	Res
5	BOR_EN	RW	0	BOR Enable 0: BOR not enabled 1: BOR enabled, BOR_LEV active
4: 0	Res	-	-	Res

4.7.8. Flash WRP Address Register (FLASH_WRP)

Address offset: 0x2C

Reset value: 0x0000 FFFF

After the power-on reset (POR/BOR/OBL_LAUNCH) is released, the corresponding value is read from the option byte area of the flash information memory and written to the corresponding option bit of this register.

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

Bit	Name	R/W	Reset Value	Function
31: 16	Res	-	-	Res
15	WRP	RW	1	0: SECTOR 15, write-protected, program and erase not allowed 1: sector 15, no write protection
14	WRP	RW	1	0: SECTOR 14, write-protected, program and erase not allowed 1: sector 14, no write protection
13	WRP	RW	1	0: SECTOR 13, write-protected, program and erase not allowed 1: sector 13, no write protection
12	WRP	RW	1	0: SECTOR 12, write-protected, program and erase not allowed 1: sector 12, no write protection
11	WRP	RW	1	0: SECTOR 11, write-protected, program and erase not allowed 1: sector 11, no write protection
10	WRP	RW	1	0: SECTOR 10, write-protected, program and erase not allowed 1: sector 10, no write protection
9	WRP	RW	1	0: SECTOR 9, write-protected, program and erase not allowed 1: sector 9, no write protection
8	WRP	RW	1	0: SECTOR 8, write-protected, program and erase not allowed 1: sector 8, write-protected
7	WRP	RW	1	0: SECTOR 7, write-protected, program and erase not allowed 1: sector 7, no write protection
6	WRP	RW	1	0: SECTOR 6, write-protected, program and erase not allowed 1: sector 6, no write protection
5	WRP	RW	1	0: SECTOR 5, write-protected, program and erase not allowed 1: sector 5, no write protection
4	WRP	RW	1	0: SECTOR 4, write-protected, program and erase not allowed 1: sector 4, no write protection
3	WRP	RW	1	0: Sector 3, write-protected, program and erase not allowed 1: sector 3, no write protection
2	WRP	RW	1	0: SECTOR 2, write-protected, program and erase not permitted 1: sector 2, no write protection

Bit	Name	R/W	Reset Value	Function
1	WRP	RW	1	0: SECTOR 1, write-protected, program and erase not permitted 1: sector 1, no write protection
0	WRP	RW	1	0: SECTOR 0, write-protected, program and erase not permitted 1: sector 0, no write protection

4.7.9. Flash Sleep Time Configuration Register (FLASH_STCR)

Address offset: 0x90

Reset value: 0x0000 6400

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

Bit	Name	R/W	Reset Value	Function
31: 16	Res	-	-	Res
15: 8	SLEEP_TIME	RW	0x64	Flash sleep time counter (counter based on HSI_10M clock) When LSI or LSE is selected for the system clock, the function of this register is optional for more optimized Run mode power consumption (recommended only when LSI or LSE is the system clock). When this function is enabled, the width of time that the Flash is in the Sleep state for each half system clock low cycle is: $t_{HSI_10M} * SLEEP_TIME$ Notes: t_{HSI_10M} is the period of HSI_10M.
7: 1	Res	-	-	Res
0	SLEEP_EN	RW	0	Flash Sleep Enable 1: Enable Flash sleep 0: Flash sleep

4.7.10. Flash TS0 register (FLASH_TS0)

Address offset: 0x100

Reset value: 0x0000 00B4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	TS0							
-	-	-	-	-	-	-	-	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 8	Res	-	-	Res
7: 0	TS0	RW	0xB4	<p>If the HSI output frequency is different, you need to set the following corresponding values</p> <p>HSI for 4MHz: 0x1E</p> <p>HSI for 8MHz: 0x3C</p> <p>HSI for 16MHz: 0x78</p> <p>HSI for 22.12MHz: 0xA6</p> <p>HSI for 24MHz: 0xB4</p>

4.7.11. Flash TS1 Register (FLASH_TS1)

Address offset: 0x104

Reset value: 0x0000 01B0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	TS1								
-	-	-	-	-	-	-	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 9	Res	-	-	Res
8: 0	TS1	RW	0x1B0	<p>If the HSI output frequency is different, you need to set the following corresponding values</p> <p>HSI for 4 MHz: 0x48</p> <p>HSI for 8 MHz: 0x90</p> <p>HSI for 16 MHz: 0x120</p> <p>HSI for 22.12 MHz: 0x18F</p> <p>HSI for 24 MHz: 0x1B0</p>

4.7.12. Flash TS2P Register (FLASH_TS2P)

Address offset: 0x108

Reset value: 0x0000 00B4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	TS2P							
-	-	-	-	-	-	-	-	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 8	Res	-	-	Res
7: 0	TS2P	RW	0xB4	<p>If the HSI output frequency is different, you need to set the following corresponding values</p> <p>HSI for 4 MHz: 0x1E</p> <p>HSI for 8 MHz: 0x3C</p> <p>HSI for 16 MHz: 0x78</p> <p>HSI for 22.12 MHz: 0xA6</p> <p>HSI for 24 MHz: 0xB4</p>

4.7.13. Flash TPS3 register (FLASH_TPS3)

Address offset: 0x10C

Reset value: 0x0000 06C0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	TPS3										
-	-	-	-	-	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 11	Res	-	-	Res
10: 0	TPS3	RW	0x6C0	<p>If the HSI output frequency is different, you need to set the following corresponding values</p> <p>HSI for 4 MHz: 0x120</p> <p>HSI for 8 MHz: 0x240</p> <p>HSI for 16 MHz: 0x480</p> <p>HSI for 22.12 MHz: 0x639</p>

				HSI for 24 MHz: 0x6C0
--	--	--	--	-----------------------

4.7.14. Flash TS3 register (FLASH_TS3)

Address offset: 0x110

Reset value: 0x0000 00B4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	TS3							
-	-	-	-	-	-	-	-	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 8	Res	-	-	Res
7: 0	TS3	RW	0xB4	<p>If the HSI output frequency is different, you need to set the following corresponding values</p> <p>HSI for 4 MHz: 0x1E</p> <p>HSI for 8 MHz: 0x3C</p> <p>HSI for 16 MHz: 0x78</p> <p>HSI for 22.12 MHz: 0xA6</p> <p>HSI for 24 MHz: 0xB4</p>

4.7.15. Flash Page Erase (PAGE ERASE) TPE Register (FLASH_PERTPE)

Address offset: 0x114

Reset value: 0x0001 1940

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

Bit	Name	R/W	Reset Value	Function
31: 17	Res	-	-	Res
16: 0	PERTPE	RW	0x11940	<p>If the HSI output frequency is different, you need to set the following corresponding values</p> <p>HSI for 4 MHz: 0x36B0</p> <p>HSI for 8 MHz: 0x6D60</p>

				HSI for 16 MHz: 0XDAC0 HSI for 22.12 MHz: 0x12E6C HSI for 24 MHz: 0x14820
--	--	--	--	---

4.7.16. Flash SECTOR/MASS ERASE TPE Register (FLASH_SMERTPE)

Address offset: 0x118

Reset value: 0x0001 1940

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

Bit	Name	R/W	Reset Value	Function
31: 17	Res	-	-	Res
16: 0	SMERTPE	RW	0x11940	If the HSI output frequency is different, you need to set the following corresponding values HSI for 4 MHz: 0x36B0 HSI for 8 MHz: 0x6D60 HSI for 16 MHz: 0XDAC0 HSI for 22.12 MHz: 0x12E6C HSI for 24 MHz: 0x14820

4.7.17. Flash PROGRAM TPE register (FLASH_PRGTPE)

Address offset: 0x11C

Reset value: 0x0000 A8C0

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

Bit	Name	R/W	Reset Value	Function
31: 16	Res	-	-	Res
15: 0	PRGTPE	RW	0xA8C0	If the HSI output frequency is different, you need to set the following corresponding values

				HSI for 4 MHz: 0xFA0 HSI for 8 MHz: 0x1F40 HSI for 16 MHz: 0x3E80 HSI for 22.12 MHz: 0x5668 HSI for 24 MHz: 0x5DC0
--	--	--	--	--

4.7.18. Flash PRE-PROGRAM TPE Register (FLASH_PRETPE)

Address offset: 0x120

Reset value: 0x0000 12C0

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

Bit	Name	R/W	Reset Value	Function
31: 14	Res	-	-	Res
13: 0	PRETPE	RW	0x12C0	If the HSI output frequency is different, you need to set the following corresponding values HSI for 4 MHz: 0x320 HSI for 8 MHz: 0x640 HSI for 16 MHz: 0xC80 HSI for 22.12 MHz: 0x1148 HSI for 24 MHz: 0x12C0

5. Power control

5.1. Power supply

5.1.1. Power Supply Block Diagram

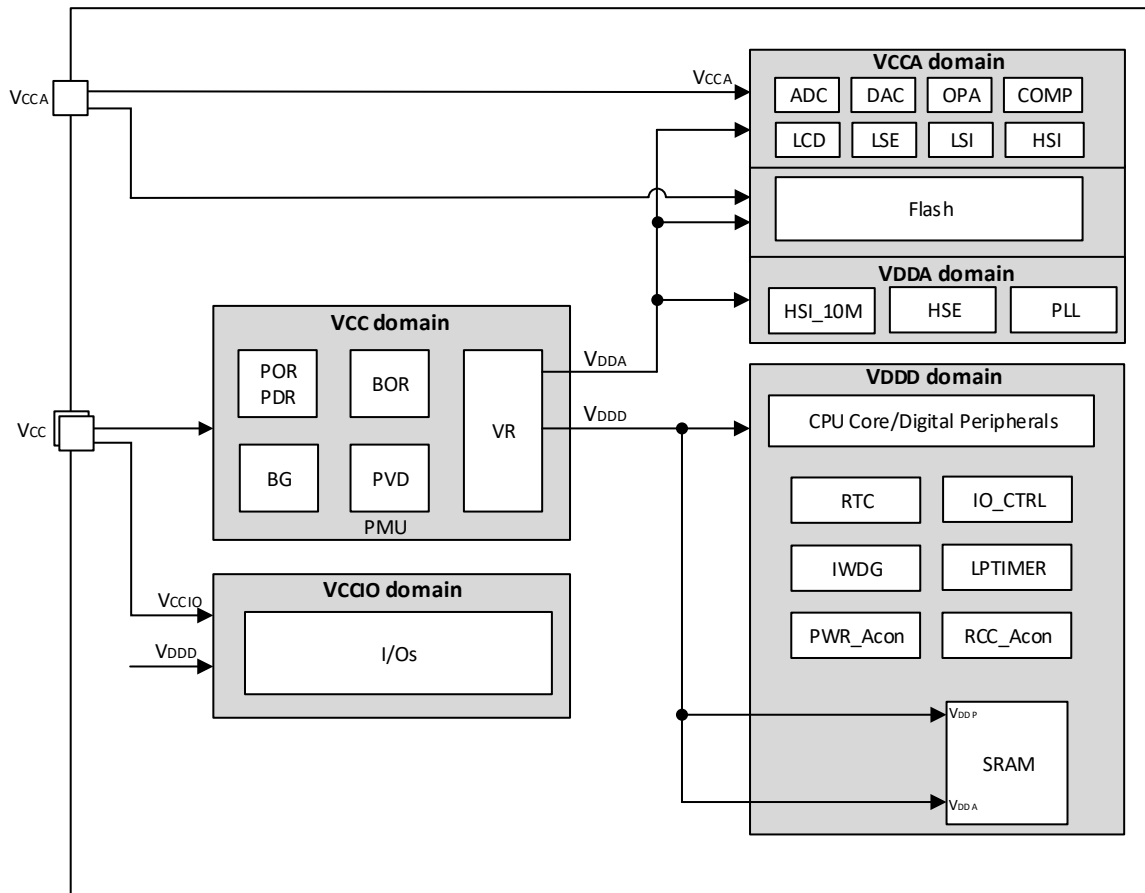


Figure 5-1 Power Supply Block Diagram

Table 5-1 Power Supply Block Diagram

No.	Power supply	Power value	Descriptions
1	V _{CC}	1.7 to 5.5 V	External power supply for I/Os and the internal regulator. It is provided externally through V _{CC} pins.
2	V _{CCA}	1.7 to 5.5 V	External analog power supply for ADC, DAC, COMP, OPA, LCD, RCs and PLL. It is provided externally through V _{CCA} pins.
3	V _{DDx} (V _{DDD} /V _{DDA})	1.2 V/1.0 V/0.9 V/0.8 V	Two embedded linear voltage regulators, MR and LPR, supply most of digital circuitry, SRAM in the device. When the MR is powered, it outputs 1.2 V.

			When entering the Stop mode it powered by MR or LPR(LPR.PWR_CR1 register),and the LPR output is determined to be 1.2 V/1.0 V/0.9 V/0.8 V(VOS.PWR_CR1 register).
--	--	--	---

5.2. Voltage regulator

The chip is designed with two voltage regulators:

- MR (Main regulator) keeps working during the normal operation state of the chip.
- LPR (Low power regulator) provides lower power consumption options in Stop mode.

The power for the VDDX comes from the MR or LPR depending on the operating mode of the chip.

In Chip Run mode, the MR stays operational, outputs 1.2 V, and the LPR turns off.

In Stop mode, power can be supplied from MR or LPR at the discretion of the software. Similarly, it is determined by software that the VDDX for the LPR supply case is 1.2 V/1.0 V/0.9 V/0.8 V after entering Stop.

5.3. Power monitoring

5.3.1. Power-On reset (POR) / Power-down reset (PDR) / Brown-out reset (BOR)

The POR/PDR module is designed within the chip and placed under the VDDx power domain to provide power-up and power-down reset for the chip. The module remains operational in all modes.

In addition to POR/PDR, BOR (Brown-out reset) is also implemented. The BOR can be enabled and disabled only by the option byte.

When the BOR is turned on, the threshold of the BOR can be selected via the option byte and both the rise and fall detection points can be individually configured.

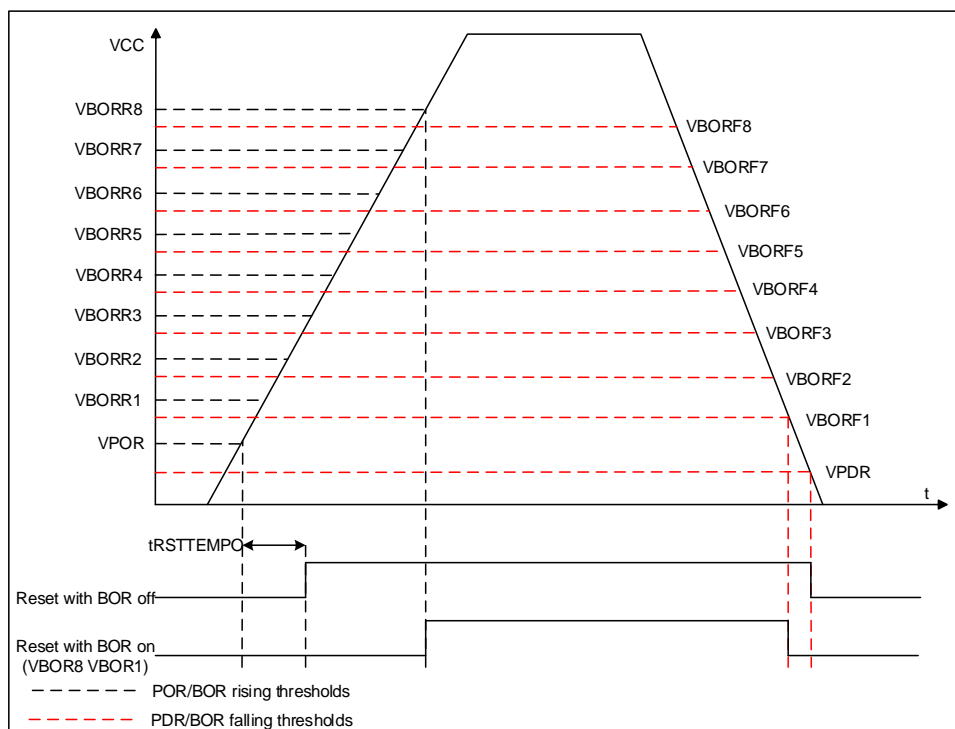


Figure 5-2 POR/PDR/BOR thresholds

5.3.2. Programmable Voltage Detector (PVD)

The module can be used to detect the VCC power supply (and also the voltage at the PB7 pin), and the detection point can be configured via registers. When VCC is above or below the detection point of the PVD, the corresponding mark is generated.

This event is internally connected to EXTI line 16, depending on the EXTI line 16 rising/falling edge configuration, and generates an interrupt when VCC rises above the detection point of the PVD, or VCC drops below the detection point of the PVD, which allows the user to perform an emergency shutdown (Shutdown) task in the interrupt service program.

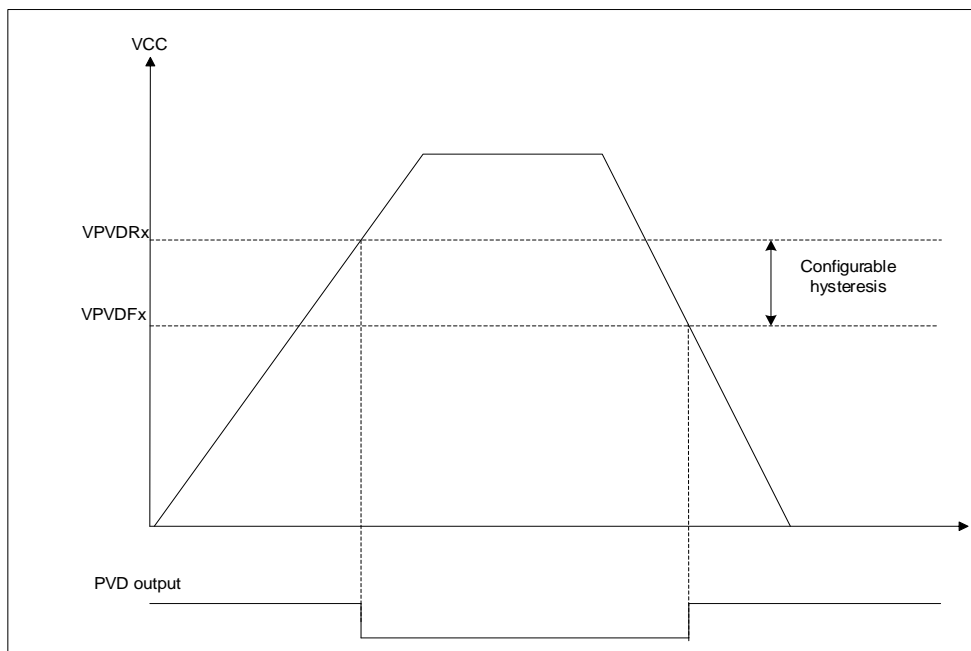


Figure 5-3 PVD Threshold

6. Low-power control

By default, the chip enters normal operation Run mode after a system or power reset. The chip can enter a low-power mode when the CPU does not need to work continuously. For example, when waiting for an external event, the software can compromise between power consumption, wake-up time, and wake-up source.

6.1. Low Power Mode

6.1.1. Introduction to Low Power Mode

The chip has 2 low power modes in addition to the normal Run mode:

- Sleep mode: CPU clock off (NVIC, SysTick can work), peripherals can be configured to keep working. (It is recommended to enable only the modules that must work and to shut down the module when it has finished working).
- Stop mode: In this mode the contents of SRAM and registers are held, HSI, HSE and PLL are turned off, and the clocks of most modules in the VDD domain are stopped.

In Stop mode, the LSI and LSE can remain operational and the RTC, LPTIMER, IWDG, etc. can remain operational. Specifically the operation of the modules in this mode, cf. [Table 6-2 Functions in each operating mode](#) ⁽¹⁾

In Stop mode, the corresponding VR state can be controlled by software and set to be powered by MR or LPR. When LPR power supply, the chip power consumption is greatly reduced, but the wake-up time is longer; when the case of keeping MR power supply, the chip power consumption is larger, but with a few cycles of fast wake-up ability.

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

1. Reduced system clock frequency
2. For unused peripherals, turn off the peripheral clocks (system clock and module clock)

In summary, the low power mode conversion diagram for this project is described below.

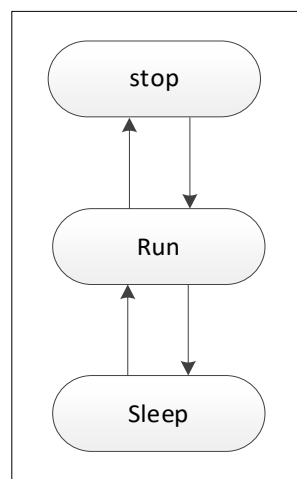


Figure 6-1 Power Mode

6.1.2. Low Power Mode Switch

Table 6-1 Low Power Mode Switches

paradigm	go into	source of wake-up call	wake-up call	Impact on the clock	Voltage regulator	
					MR	LPR
Sleep (sleep-now or sleep-on-exit)	WFI or Return from ISR	Any interruptions	Same as before entering sleep	CPU clock stops, no effect on other clocks and clock sources.	Open (1)	mountain pass
	WFE	wake-up event				
Stop	SLEEPDEEP bit+ WFI or Return from ISR or WFE Note: LSI cannot be selected for the system clock	Any EXTI Line configured for wakeup (EXTI register configuration), IWDG, NRST	HSISYS The HSI maintains the frequency configuration before entering STOP, without crossovers	HSI shutdown; HSE closed; PLL off; The LSI can be selected on or off; LPTIMER, RTC, IWDG: Configured by software to work or not; Modules such as low-power wake-up and partial RCC keep working; The remaining peripheral modules are clocked off.	Software Configuration Switches	Software configurable switch, if on, output voltage 1.2 V/1.0 V/0.9 V/0.8 V configurable

Note 1: The software has to configure the state of the VR to be in MR mode in order to enter Sleep mode.

6.1.3. Functions in each operating mode

Table 6-2 Functions in each operating mode ⁽¹⁾

Peripherals	Run	Sleep	Stop	
			VR@LPR or VR@MR	wake-up call (computing)
CPU	Y	-	-	-
Flash memory	Y	Y	- (2)	-
SRAM	Y	O(3)	- (4)	-
Brown-out reset (BOR)	Y	Y	O	O
PVD	O	O	O	O
DMA	O	O	-	-
HSI	O	O	-	-
HSE	O	O	-	-
LSI	O	O	O	-
PLL	O	O	-	-

HSE Clock Security System (CSS)	O	O	-	-
RTC	O	O	O	O
USART1	O	O	-	-
USART2	O	O	-	-
I2C	O	O	-	-
SPI1	O	O	-	-
ADC	O	O	-	-
COMP1/COMP2/3	O	O	O	O
Temperature sensor	O	O	-	-
Timers (TIM1/TIM3 /TIM14/TIM16/TIM17)	O	O	-	-
LPTIM	O	O	O	O
IWDG	O	O	O	O
WWDG	O	O	-	-
SysTick timer	O	O	-	-
CRC	O	O	-	-
GPIOs	O	O	O	O

1. Y = Yes (enable); O = Optional (off by default, can be enabled by software); - = Not available
2. Flash is not powered down, but no clock is supplied, entering the lowest power state.
3. The SRAM clock can be turned on or off.
4. The SRAM does not power down, but no clock is supplied and it enters the lowest power state.

6.2. Sleep mode

6.2.1. Entering sleep mode

Sleep mode is entered by executing the WFI (Wait for interrupt) or WFE (Wait for event) instruction. Depending on the SLEEPONEXIT bit of the Cortex M0+'s System Control Register, there are two optional mechanisms for entering Sleep mode.

- Sleep-now: if the SLEEPONEXIT bit is 0, it enters Sleep mode immediately after executing WFI or WFE.
- Sleep-on-exit: If the SLEEPONEXIT bit is 1, it enters Sleep mode when exiting a low priority interrupt ISR.

In Sleep mode, all IO pins remain in the same state as in Run mode.

6.2.2. Exit sleep mode

If Sleep mode is entered with WFI, any peripheral interrupt obtained by the NVIC can wake up the chip from sleep mode.

If Sleep mode is entered with WFE, the chip exits Sleep mode when an event occurs. Wakeup events can be generated in the following ways:

- Enable interrupts in the Peripheral Control Register, not in the NVIC, and enable the SEVONPEND bit of the Cortex M0+. The peripheral interrupt Pending bit and the peripheral NVIC IRQ channel Pending bit (in the NVIC's interrupt clear Pending register) must be cleared when the chip continues execution after waking up from WFE.
- Alternatively, configure the external or internal EXTI line for event mode. When the CPU continues execution after waking up from WFE, it does not have to clear the peripheral interrupt Pending bit, or the NVIC IRQ channel Pending bit corresponding to event Line is not set.

This mode has the shortest Wakeup time and wastes no time on interrupt entry and exit.

Table 6-3 Sleep-now

Sleep-now mode	Description
Mode entry	WFI or WFE, and: <ul style="list-style-type: none"> - SLEEPDEEP = 0 and - SLEEPONEXIT = 0
Mode exit	If sleep mode is entered via WFI, the exit method is: interrupt. If sleep mode is entered via WFE, the exit is: wakeup event.
Wakeup latency	not have

Table 6-4 Sleep-on-exit

Sleep-on-exit	Description
Mode entry	WFI, and: <ul style="list-style-type: none"> - SLEEPDEEP = 0 and - SLEEPONEXIT = 1
Mode exit	disruptions
Wakeup latency	not have

6.3. Stop mode

Stop mode is based on the Cortex-M0+'s DEEPSLEEP and Gating of the peripheral clock, and the VR can be configured to be powered by either MR or LPR. In this mode, HSI, PLL, and HSE are turned off, SRAM and register contents are held, LSI, LPTIMER, RTC, and IWDG can be configured by software to work or not, low-power wake-up and some RCC logic, etc. are held working, and clock inputs to the digital modules in the remaining VDDx domains are turned off.

In Stop mode, all IO pins remain in the same state as in Run mode.

6.3.1. Enter Stop mode

To further reduce the power consumption in Stop mode, configure PWR_CR1.LPR=1 so that the VR can enter LPR power supply.

If a Flash erase/write operation is in progress, entry into Stop mode is delayed until the memory access is complete (determined by software reading the BSY bit of the FLASH_SR register to determine if the erase/write operation is currently complete).

If an operation is in progress on the APB bus, entry of Stop mode is also delayed until the APB access is complete (controlled by software).

If the system clock source is a high-speed clock source (PLL/HSE) before entering low-power mode again, software switching of the system clock source to HSI is required to ensure successful system clock switching.

6.3.2. Exiting Stop Mode

The HSI is selected as the system clock when exiting Stop mode via an interrupt or Wakeup event.

In Stop mode, if the VR is in LPR, there is an additional delay to wake up from Stop mode.

In Stop mode, if the VR is in MR state, the current consumption will be high, but the wake-up time will be minimized.

Table 6-5 Stop Mode

Stop mode	descriptive
Access Mode	<p>WFI (Wait for interrupt) or WFE (Wait for event), and:</p> <ol style="list-style-type: none"> Configuration settings: <ul style="list-style-type: none"> Select the VR to operate under MR or LPR via the LPR bit of PWR_CR1 With the VOS bit of PWR_CR1, the LPR mode is selected to provide 1.2 V, 1.0 V, 0.9 V, 0.8 V Configure FLASH wake-up time via FLS_SLPTIME of PWR_CR1 Setting the SLEEPDEEP bit of the Cortex M0+ <p>Notes:</p> <ol style="list-style-type: none"> In order to enter Stop mode, the Pending bits of all EXTI lines (EXTI_PR register), the interrupt Pending bits of all peripherals, and the RTC alarm flag bit, must be reset. Otherwise, the process of entering STOP mode is ignored and the program continues to execute. If the application needs to turn off the HSE, PLL before entering Stop mode, the system clock source must be switched to the HSI and then the HSEON bit must be cleared. In order to equalize the power consumption variation of the chip as much as possible, the software needs to follow the principle of gradual shutdown: gradually turn off the clock of each module, select HSI as the system clock, and turn off HSE and PLL. To shorten the wake-up time, the system clock should be configured to select the HSI high-frequency clock before entering Stop mode, and the HPRE of the RCC_CFGR register should be set to 0. Otherwise, the hardware switching clock will consume extra clock after wake-up.
exit mode	<p>If using WFI to enter Stop mode:</p> <ul style="list-style-type: none"> Any EXTI line configured for interrupt mode (the corresponding EXTI interrupt vector must be enabled in the NVIC)

Stop mode	descriptive
	If using WFE to enter Stop mode: <ul style="list-style-type: none"> - Any EXTI line that is configured for event mode - CPU SEVONPEND position bit case interrupt pending bit
wake-up call delay	LPR to MR wakeup time + HSI wakeup time + Flash wakeup time

6.4. Reduced system clock frequency

In Run mode, the frequency of the system clock (SYSCLK, HCLK, PCLK) can be reduced by configuring the frequency division through the prescaler register. These prescalers can also be used to reduce the frequency of a peripheral before entering Sleep mode.

6.5. Peripheral clock gating

In Run mode, the AHB clock (HCLK) and APB clock (PCLK) of individual peripherals and memories can be stopped at any time to reduce power consumption.

To further reduce power consumption in Sleep mode, the peripheral's clock can be stopped before executing the WFI or WFE instruction.

6.6. Power Management Registers

The registers of this peripheral can be accessed by half-word or full-word.

6.6.1. Power Control Register 1 (PWR_CR1)

Address offset: 0x00

Reset value: 0x0000 0000 (reset by POR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res												HSION_CTRL	Res		
-												RW	-		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	LPR	FLS_SLPTIME[1:0]		Res	VOS [1:0]		DBP	Res				Res			
-	RW	RW	RW		RW	RW		-				-			

Bit	Name	R/W	Reset Value	Function
31: 20	Res	-	-	Res
19	HSION_CTRL	RW	0	When waking up from Stop mode, the HSI turns on the time control. 0: Wait for MR to stabilize and enable HSI; 1: Enable HSI immediately on wakeup.
18: 15	Res	-	-	Res
14	LPR	RW	0	Low power regulator 0: Main regulator working in stop mode 1: Low power regulator working in stop mode

13: 12	FLS_SLPTIME[1:0]	RW	2' b00	<p>The Stop mode wake-up timing requires a wait time after the HSI is stabilized and before FLASH operation.</p> <p>2'b00: 1 μs</p> <p>2'b01: 2 μs</p> <p>2'b10: 3 μs</p> <p>2'b11: 0 μs</p> <p>Note: When this register is set to 2'b11, it indicates that the program is executed from SRAM, not Flash, after wake-up. And the program guarantees that Flash will not be accessed within 3μs after waking up to execute the program.</p>
11	Res	-	-	Res
10: 9	VOS [1:0]	RW	0	<p>Voltage regulation range selection</p> <p>00: VDD = 1.2 V after entering Stop mode</p> <p>01: VDD = 1.0 V after entering Stop mode</p> <p>10: VDD = 0.9 V after entering Stop mode</p> <p>11: VDD = 0.8 V after entering Stop mode</p>
8	DBP	RW	0	<p>RTC Write Protect Disable</p> <p>After a reset, the RTC is in a write-protected state to prevent accidental writes. To access the RTC this bit must be set to 1.</p> <p>0: Access to the RTC is disabled</p> <p>1: RTC can be accessed</p>
7: 0	Res	-	-	Res

6.6.2. Power Control Register 2 (PWR_CR2)

Address offset: 0x04

Reset value: 0x0000 0500 (reset by POR)

Note: This register is a PVD function related register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res				FLT_TIME[2:0]			FLTEN	Res	PVDT [2:0]			Res	SRCSEL	Res	PVDE
-				RW			RW	-	RW			-	RW	-	RW

Bit	Name	R/W	Reset Value	Function
31: 12	Res	-	-	Res
11: 9	FLT_TIME[2:0]	RW	3' b010	Digital Filter Time Configuration

Bit	Name	R/W	Reset Value	Function
				110: filtering time of about 30.7 ms (1024 LSI/LSE clocks) 101: Filtering time of about 3.8 ms (128 LSI/LSE clocks) 100: filtering time of about 1.92 ms (64 LSI/LSE clocks) 011: Filtering time approx. 480 μ s (16 LSI/LSE clocks) 010: Filtering time approx. 120 μ s (4 LSI/LSE clocks) 001: Filtering time approx. 60 μ s (2 LSI/LSE clocks) 000: Filtering time approx. 30 μ s (1 LSI/LSE clock)
8	FLTEN	RW	1	Digital Filter Function Enable Control 0: Prohibited 1: Enabling
7	Res	-	-	Res
6: 4	PVDT [2:0]	RW	000	Voltage rising edge detection threshold (falling edge detection threshold is reduced by 0.1 V accordingly) and PVDIN detection control. 000: VPVD0 (around 1.8 V) 001: VPVD1 (around 2.0 V) 010: VPVD2 (around 2.2 V) 011: VPVD3 (around 2.4 V) 100: VPVD4 (around 2.6 V) 101: VPVD5 (around 2.8 V) 110: VPVD6 (around 3.0 V) 111: VPVD7 (around 3.2 V)
3	Res	-	-	Res
2	SRCSEL	RW	0	PVD detection power supply selection. 0: VCC 1: Detect PB7 pin If this position is 1, the voltage on PB7 is internally compared to VREFINT (both rising and falling thresholds). The PVDT register setting is invalid in this case.
1	Res	-	-	Res
0	PVDE	RW	0	Voltage Detect Enable Bit 0: Voltage detection not enabled 1: Voltage detection enable PVDE write-protected if SYSCFG_C, G2.PVD_LOCK=1. Write protection is reset only when the system is reset.

6.6.3. Power Supply Status Register (PWR_SR)

Address offset: 0x14

Reset value: 0x0000 0000 (reset by POR)

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	PVDO	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	R	-	-	-	-	-	-	-	-	-	-	-

Bit	Name	R/W	Reset Value	Function
31: 12	Res	-	-	Res
11	PVDO	R	0	PVD test result output. 0: the detected VCC or PB7 exceeds the comparison threshold selected by the PVD 1: The detected VCC or PB7 is lower than the comparison threshold selected by the PVD
10: 0	Res	-	-	Res

7. Reset

Two types of reset are designed within the chip, namely: power reset and system reset.

7.1. Reset source

7.1.1. Power Reset

A power reset resets all registers out and is generated under the following conditions:

The POR/ BOR generated by the analog circuit realizes the detection of VCC.

- The reset is released when the VCC voltage rises to the trigger value;
- A reset is generated when the VCC voltage drops to a certain trigger value.

7.1.2. System reset

A system reset sets most registers to their reset values; some special registers, such as the reset identification bit register, are not reset by the system.

A system reset is generated when the following events are generated:

- Reset of the NRST pin
- Window watchdog (WWDG) reset
- Independent Watchdog Dog (IWDG) Reset
- Cortex-M0+ SYSRESETREQ Software Reset
- Option byte load (OBL) reset

7.1.3. NRST pin (External reset)

With the loading of the option byte (NRST_MODE bit), the NRST pin can be configured in the following modes (see the option byte description for specific configuration):

- Reset Input

In this mode, any valid reset signal on the NRST pin is passed to the internal logic, but resets generated internally by the chip are not output on the NRST pin.

In this configuration mode, the PF2 function of the GPIO is disabled.

The NRST pin input is then passed through the deburring circuit (deburring can be configured to be disabled) to generate an external reset of the chip.

- GPIO

In this mode, this PIN can be used as a standard GPIO, i.e. PF2. The reset function on Pin is invalid. Chip reset will only be generated internally by the chip and cannot be passed on to the pin.

Note: After power-on reset, the NRST pin is configured in reset input mode by default.

7.1.4. Watchdog reset

See Independent Watchdog (IWDG) and Window Watchdog (WWDG).

7.1.5. Software reset

A software reset can be achieved by setting the SYSRESETREQ bit of the ARM M0+'s interrupt and reset control register.

7.1.6. Option byte loader reset

Software generates an option byte loader reset by configuring FLASH_CR.OBL_LAUNCH=1, which initiates option byte reloading.

8. Clocks

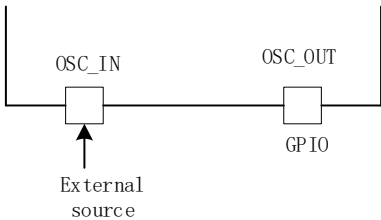
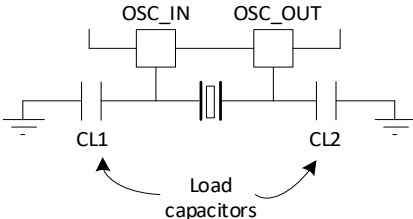
8.1. Clock source

8.1.1. External high-speed clock HSE

The external high-speed clock (HSE) comes from two sources:

- External XTAL OSC + internal oscillator circuitry
- External clock input via OSC_IN structure (HSEBYP=1)

Table 8-1 HSE Clock Sources

Clock source	Hardware configuration
external clock	
External Crystal	

External high-frequency OSC with a frequency range of 4 ~ 32 MHz.

The stabilization time of the HSE clock is determined by RCC_ECSCR. HSE_STARTUP register configuration. When the HSE goes from OFF to ON, it needs to wait for the stabilization time, and after stabilization, the hardware sets the RCC_CR.HSERDY register. When HSEBYP=1, the stabilization time is halved compared to non-bypass mode.

HSE clock related registers refer to RCC_ECSCR.

8.1.2. Internal high-speed clock HSI

Internal RC oscillator with reference frequencies of 4 MHz, 8 MHz, 16 MHz, 22.12 MHz, and 24 MHz is available. Compared to XTAL OSC, RC OSC has low power consumption and short stabilization time, but low accuracy.

After power-on reset, the HSI calibration value needs to be software loaded into the RCC_ICSCR.HSI_TRIM register. When the system is reset, this register is reset with it.

After waking up from Stop mode, only the HSI can be used as the system clock source.

8.1.3. Internal low-speed clock LSI

Internal low frequency 32.768 kHz clock.

8.1.4. HSI10M Clock

This clock is used as a low-precision clock for filter counting on the NRST pin and for low-power handling during Flash low-speed Run.

8.1.5. PLL

PLL module reference clock is HSI or HSE, PLL input clock frequency range is required to be 16 ~ 24 MHz, not in this range can not guarantee the output clock frequency and stability.

The PLL supports 2x or 3x frequency. When the USB module is in operation, it is necessary to select the HSI reference frequency as 16 MHz and the PLL multiplication factor as 3, thus generating a frequency of 48 MHz. At this point, if PLL is selected for the system clock, the maximum frequency will also be limited to 48 MHz.

8.1.6. LSE Clock

External 32.768 kHz OSC for use as a low-power clock.

A balance between stabilization time and power consumption can be made by configuring LSE_DRV. LSE stabilization time by RCC_ECSCR. LSE_STARTUP register configuration.

Similar to the HSE source, the LSE has two sources:

- 32.768 kHz XTAL + internal oscillator circuitry
- External clock input via OSC_IN (LSEBYP=1)

In the case of LSE bypass, the stabilization time is halved compared to non-bypass mode.

8.2. The Clock Tree

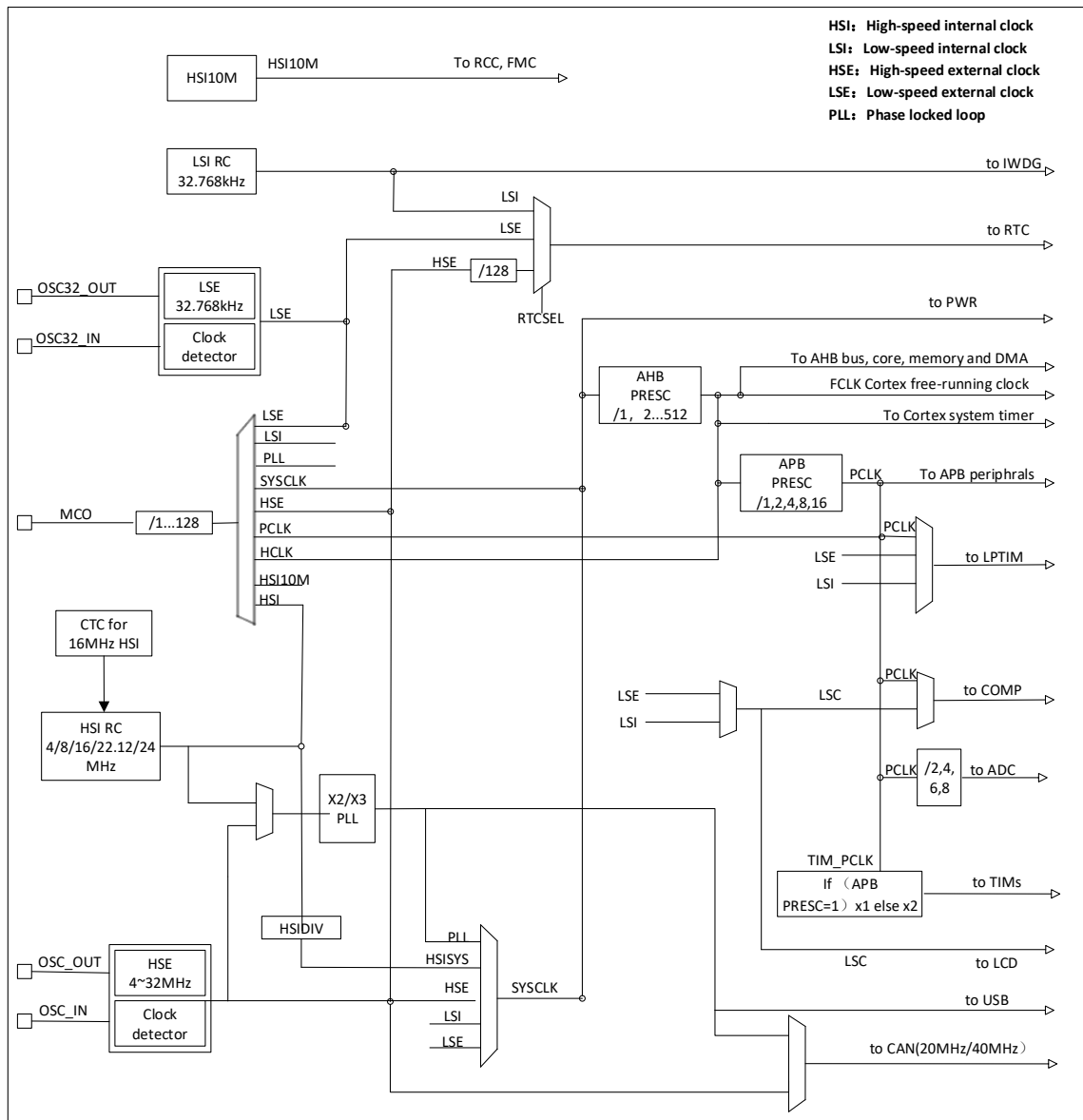


Figure 8-1 System clock structure

8.3. Clock Safety System (CSS)

Clock security consists mainly of the following aspects:

- Clock Configuration and Stateful Security
- Clock Source HSE Safety
- Clock Source LSE Security
- IWDG-based clock security
- Timer-based clock security

8.3.1. Clock Configuration and Stateful Security

The software periodically reads back the clock configuration and status registers to obtain information about the system's current clock and determine whether it is as expected.

8.3.2. Clock source HSE monitoring

The HSE clock security system can be activated by software by configuring `RCC_CR.CSSON`. In this case, the clock detection function is turned on after the HSE is started. When HSE is turned off, the clock detection function is turned off.

If a clock error is detected on the HSE, the HSE is automatically shut down and the clock error event is sent to the brake inputs of TIM1 (Advanced Timer) and TIM15/TIM16/TIM17 (General Purpose Timer) and an interrupt is generated to notify the software of the error (Clock Security System Interrupt, CSSI). CSSI, which in turn allows the MCU to perform a rescue operation. CSSI is linked to the Cortex-M0+'s NMI (non-maskable interrupt, Non-maskable interrupt) Exception vector.

Note: Once CSS is enabled, and if the HSE clocks Failure, a CSS interrupt is generated and an NMI is automatically generated. This NMI will continue to execute until the CSS interrupt pending bit is cleared. Therefore, the CSS interrupt must be cleared in the NMI's handler by setting the CSSC bit in the Clock Interrupt Register (`RCC_CICR`).

If the HSE is used directly or indirectly as the system clock (indirectly meaning: it is used as an input to the PLL and the PLL is used as the system clock), Clock Failure will cause the system clock to automatically switch to the HSI while turning off the HSE. If the HSE is the input clock to the PLL when the clock is Failure, the PLL will also be turned off.

8.3.3. Clock source LSE monitoring

The LSE clock security system can be activated by software by configuring `RCC_BDCR.LSECSSON`. In this case, the clock detection function is turned on after the LSE is started. When the LSE is turned off, the clock detection function is turned off.

If a clock error is detected on the LSE, the LSE is automatically turned off and the clock error event is sent to the brake inputs of TIM1 (Advanced Timer) and TIM15/TIM16/TIM17 (General Purpose Timer) and an interrupt is generated to notify the software of the error (CSSI), which in turn allows the MCU to perform a rescue operation. CSSI is linked to the NMI (Non-maskable interrupt) exception vector of the Cortex-M0+.

Note: Once LSECSS is enabled and if the LSE clock is incorrect, a CSS interrupt is generated and an NMI is automatically generated. This NMI will continue to execute until the CSS interrupt pending bit is cleared. Therefore, the CSS interrupt must be cleared in the NMI's handler by setting the CSSC bit in the Clock Interrupt Register (`RCC_CICR`).

If the LSE is used as the system clock, Clock Failure will cause the system clock to automatically switch to the LSI while turning off the LSE. Also, if LSE is selected for the LPTIM and RTC count clock, it will automatically switch to LSI.

8.4. Output Clock Capability

In order to facilitate board-level applications, save BOM costs, and for Debug, etc., the chip is required to provide a clock output function. In other words, the MCO signal of the table below (and divided frequency) is used to realize the clock output function through the multiplexing function of GPIO.

Table 8-2 Output Clock Selection

clock source	MCO exportable clock source
HSI	√
SYSCLK	√
HSE	√
LSI	√
PLL	√
LSE	√

Note: When making a switch to the MCO clock source and selecting the GPIO AF function as the start of the MCO, the MCO may generate a burr and needs to be avoided for that period of time.

8.5. Reset/Clock Register

The module's registers can be accessed in full words (32 bits), half words (16 bits), and bytes (8 bits).

8.5.1. Clock Control Register (RCC_CR)

Address offset: 0x00

Reset value: 0x0000 0100

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	PLLRDY	PLLON	Res	ADC_DIV		Res	CSS ON	HSE BYP	HSE RDY	HSE ON
-	-	-	-	-	-	R	RW	-	RW		-	RS	RW	R	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	HSIDIV [2:0]			HSI RDY	Res	HSION	Res	Res	Res	Res	Res	Res	Res	Res
-	-	RW			R	-	RW	-	-	-	-	-	-	-	-

Bit	Name	R/W	Reset Value	Function
31: 26	Res	-	-	Res
25	PLLRDY	R	0	PLL clock ready flag. Hardware set to indicate that the PLL clock is latched. 0: PLL unlocked 1: PLL locked
24	PLLON	RW	0	PLL Enable.

Bit	Name	R/W	Reset Value	Function
				The hardware clears this bit when the system enters Stop mode. This bit cannot be cleared when the PLL clock is used as the system clock. 0: PLL off 1: PLL Open
23	Res	-	-	Res
22: 21	ADC_DIV	RW	0	ADC dividing factor 00: 2 crossover frequency 01: 4-way frequency 10: 6 crossover 11: 8-way
20	Res	-	-	Res
19	CSSON	RS	0	HSE Clock Safety System Enable. When this bit is 1, the hardware enables the clock detection module if the HSE OSC is ready, and disables the clock detection module if the HSE detection fails. 0: Clock safety system off (clock detection off) 1: Clock safety system on (clock detection on if HSE clock is stable, otherwise clock detection off)
18	HSEBYP	RW	0	The HSE shields the crystal and selects the pin to input the clock. This bit can only be written when HSEON=0. 0: HSE crystal not shielded, external high-speed clock selects external crystal 1: HSE crystal shielding, external high-speed clock selection external pin input clock source
17	HSERDY	R	0	HSE crystal clock ready flag. This bit is set by hardware to indicate that the HSE crystal is stable. 0: HSE crystal not ready 1: HSE crystals ready to go Note: When HSEON is cleared, HSERDY is cleared after 6 HSE clock cycles.
16	HSEON	RW	0	HSE crystal enable. When the system enters Stop mode, the hardware clears this bit and turns off the HSE crystal. This bit cannot be set to 0 when the HSE is used as the system clock source. 0: HSE crystal off

Bit	Name	R/W	Reset Value	Function
				1: HSE crystal turn on
15: 14	Res	-	-	Res
13: 11	HSIDIV	RW	0	The HSI generates the crossover coefficients for the HSISYS clock. 000: 1 001: 2 010: 4 011: 8 100: 16 101: 32 110: 64 111: 128
10	HSIRDY	R	0	HSI clock ready flag. A hardware set indicates that the HSI OSC is stable. This bit is only valid when HSION=1. 0: HSI OSC not ready 1: HSI OSC ready
9	Res	-	-	Res
8	HSION	RW	1	HSI Clock Enable. The hardware will clear this register as necessary to stop the HSI when it enters stop mode. 0: HSI OSC off 1: HSI OSC On
7: 0	Res	-	-	Res

8.5.2. Internal clock source calibration register (RCC_ICSCR)

Address offset: 0x04

Reset value: 0x00FF 1080

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res		Res	LSI_TRIM[8:0]								
-	-	-	-	-		-	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
HSI_FS[2:0]			HSI_TRIM[12:0]												
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 25	Res	-	-	Res

Bit	Name	R/W	Reset Value	Function
24: 16	LSI_TRIM	RW	9'h0FF	The internal low-speed clock frequency is adjusted so that the internal low-speed clock can output 32.768 kHz by calibration. The calibration value is saved in the Flash at the following address: 32.768 kHz calibration value address: 0x1FFF 0FA4
15: 13	HSI_FS	RW	3'b000	HSI frequency selection: 000: 4 MHz 001: 8 MHz 010: 16 MHz 011: 22.12 MHz 100: 24 MHz ≥101: 4 MHz Upon power-up, 4 MHz is selected by default and hardware switches to 8 MHz after the reload option byte is completed.
12: 0	HSI_TRIM	RW	13'h1080	Clock Frequency Adjustment, changing the value of this register adjusts the output frequency of the HSI. Each increase of 1 in the register value increases the output frequency of the HSI by about 0.2%, with a total adjustment range of 4 to 24 MHz. The calibration values corresponding to 24 MHz/22.12 MHz/16 MHz/8 MHz/4 MHz are stored in the following addresses in Flash: 24 MHz calibration value address: 0x1FFF 3220 22.12 MHz calibration value address: 0x1FFF 3218 16 MHz calibration value address: 0x1FFF 3210 8 MHz calibration value address: 0x1FFF 3208 4 MHz calibration value address: 0x1FFF 3200

8.5.3. Clock Configuration Register (RCC_CFGR)

Address offset: 0x08

Reset value: 0x0000 0000

When the clock source is switched, there is a wait period of 1 or 2 clocks for accessing this register.

When the APB or AHB divider value is updated, there may be a wait period of 0 ~ 15 clocks to access this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	MCOPRE [2:0]			MCOSEL [3:0]				Res	Res	Res	Res	Res	Res	Res	Res
-	RW	RW	RW	RW	RW	RW	RW	-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	PPRE [2:0]			HPRE [3:0]				Res	Res	SWS[2:0]			SW[2:0]		
-	RW	RW	RW	RW	RW	RW	RW	-	-	R	R	R	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31	Res	-	-	Res
30: 28	MCOPRE [2:0]	RW	0	<p>MCO (microcontroller clock output) crossover coefficient. Software controls these bits to set the crossover coefficient of the MCO output:</p> <p>000: 1 001: 2 010: 4 011: 8 100: 16 101: 32 110: 64 111: 128</p> <p>This crossover factor is to be configured before the MCO output is enabled.</p>
27: 24	MCOSEL [3:0]	RW	0	<p>MCO Selection</p> <p>0000: no clock, MCO output disabled 0001: SYSCLK 0010: HSI 0011: HSI0 0100: HSE 0101: PLL CLK 0110: LSI 0111: LSE 1000: HCLK 1001: PCLK Other: no clock</p> <p>Note: Incomplete output clocks may occur during the clock startup or switching phase.</p>
23: 15	Res	-	-	Res
14: 12	PPRE [2:0]	RW	0	<p>This bit is controlled by software. To generate the PCLK clock, it sets the HCLK division factor as follows:</p> <p>0xx: 1 100: 2 101: 4 110: 8 111: 16</p>
11: 8	HPRE [3:0]	RW	0	<p>AHB clock division factor.</p> <p>The software controls this bit. To generate the HCLK clock, it sets the dividing factor of SYSCLK as follows:</p>

Bit	Name	R/W	Reset Value	Function
				0xxx: 11000: 21001: 41010: 81011: 161100: 641101: 1281110: 2561111: 512 In order to ensure that the system works properly, it is necessary to configure the appropriate frequency according to the VR power supply. Note: It is recommended to switch the crossover coefficients step by step.
7: 6	Res	-	-	Res
5: 3	SWS[2:0]	R	0	System clock toggle status bit These bits are controlled by hardware and indicate which clock source is currently being used as the system clock: 000: HSI SYS001: HSE010: PLL CLK011: LSI100: LSE Other: Reserved
2: 0	SW[2:0]	RW	0	System clock source selection bit. These bits are controlled by software and hardware and are used to select the system clock: 000: HSI SYS001: HSE010: PLL CLK011: LSI100: LSE remaining: reserved Hardware configurations for HSI SYS include: 1) System exit from Stop mode 2) Software configuration 001 (HSE) with HSE failure (HSE is the system clock source)

8.5.4. PLL configuration register (RCC_PLLCFGR)

Address offset: 0x0C

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 4	Res	-	-	Res
3: 2	PLLMUL [1:0]	RW	2'b0	PLL multiplication factor 00: x2 01: x3

Bit	Name	R/W	Reset Value	Function
				11: Res
1: 0	PLLSRC [1:0]	RW	0	PLL clock source selection. 00: No clock01: Reserved10: HSI11: HSE

8.5.5. External Clock Source Control Register (RCC_ECSCR)

Address offset: 0x10

Reset value: 0x0003 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	LSE_STARTUP		Res	Res	LSE_DRV	
-	-	-	-	-	-	-	-	-	-		RW	-	-		RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	HSE_STARTUP		Res	HSE_DRV	
-	-	-	-	-	-	-	-	-	-	-		RW			RW

Bit	Name	R/W	Reset Value	Function
31: 22	Res	-	-	Res
21: 20	LSE_STARTUP	RW	0	LSE crystal stabilization time selection. LSEBYP=0: 00: 4096 LSE clock cycles 01: 2048 LSE clock cycles 10: 8192 LSE clock cycles 11: Direct output regardless of stabilization time LSEBYP=1: 00: 2048 LSE clock cycles 01: 1024 LSE clock cycles 10: 4096 LSE clock cycles 11: Direct output regardless of stabilization time
19: 18	Res	-	-	Res
17: 16	LSE_DRV	RW	0x3	Low-speed crystal drive capability selection. 00: Reserved; 01: Weak drive capability 10: Strong drive capability (recommended) 11: Strongest drive capability Note: Appropriate drive capability needs to be selected based on crystal characteristics, load capacitance, and board parasitic parameters. The higher the drive capability the higher the

Bit	Name	R/W	Reset Value	Function
				power consumption, the weaker the drive capability the lower the power consumption.
15: 5	Res	-	-	Res
4: 3	HSE_STARTUP	RW	0	<p>HSE stabilization time selection.</p> <p>HSEBYP=0:</p> <p>00: 4096 HSE clocks</p> <p>01: 2048 HSE clocks</p> <p>10: 8192 HSE clocks</p> <p>11: Direct output regardless of stabilization time</p> <p>HSEBYP=1:</p> <p>00: 2048 HSE clocks</p> <p>01: 1024 HSE clocks</p> <p>10: 4096 HSE clocks</p> <p>11: Direct output regardless of stabilization time</p>
2	Res	-	-	Res
1: 0	HSE_DRV	RW	0x3	<p>High-speed crystal drive capability selection.</p> <p>00: Reserved;</p> <p>01: Weak drive capability</p> <p>10: Strong drive capability (recommended)</p> <p>11: Strongest drive capability</p> <p>Note: Appropriate drive capability needs to be selected based on crystal characteristics, load capacitance, and board parasitic parameters. The higher the drive capability the higher the power consumption, the weaker the drive capability the lower the power consumption.</p>

8.5.6. Clock interrupt enable register (RCC_CIER)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res										PLL RDYIE	HSE RDYIE	HSI RDYIE	Res	LSE RDYIE	LSI RDYIE
-										RW	RW	RW	-	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 6	Res	-	-	Res
5	PLLRDYIE	RW	0	PLL ready interrupt enable. 0: Prohibited 1: Enabling
4	HSE RDYIE	RW	0	HSE clock ready interrupt enable. 0: Prohibited 1: Enabling
3	HSIRDYIE	RW	0	HSI clock ready interrupt enable. 0: Prohibited 1: Enabling
2	Res	-	-	Res
1	LSE RDYIE	RW	0	LSE clock ready interrupt enable. 0: Prohibited 1: Enabling
0	LSIRDYIE	RW	0	LSI clock ready interrupt enable. 0: Prohibited 1: Enabling

8.5.7. Clock Interrupt Flag Register (RCC_CIFR)

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res						LSE CSSF	CSSF	Res	Res	PLL RDYF	HSE RDYF	HSI RDYF	Res	LSE RDYF	LSI RDYF
-						R	R	-	-	R	R	R	-	R	R

Bit	Name	R/W	Reset Value	Function
31: 10	Res	-	-	Res
9	LSECSSF	R	0	LSE Clock Safety System (CSS) interrupt flag. This register is set when the hardware fails to detect the LSE OSC clock. 0: LSE clock detection failure interrupt not generated; 1: LSE clock detection failure interrupt generated; Write LSECSSC register 1 to clear this bit.
8	CSSF	R	0	HSE Clock Safety System Interrupt Identification Bit.

Bit	Name	R/W	Reset Value	Function
				<p>This register is set when the hardware fails to detect the HSE OSC clock.</p> <p>0: HSE clock detection failure interrupt not generated; 1: HSE clock detection failure interrupt generated; Write CSSC register 1 to clear this bit.</p>
7: 6	Res	-	-	Res
5	PLLRDYF	R	0	<p>PLL ready to interrupt flag.</p> <p>Hardware sets this register when PLL lock and PLLRDYIE=1.</p> <p>0: PLL lock interrupt not generated; 1: PLL lock interrupt generation; Write PLLRDYC register 1 to clear this bit.</p>
4	HSERDYF	R	0	<p>HSE ready interrupt identification bit</p> <p>This bit is set by hardware when HSE is stable and HSERDYIE is enabled. The software clears this bit by setting the HSERDYC bit.</p> <p>0: No clock ready interrupt caused by HSE 1: There is a clock ready interrupt caused by HSE Write HSERDYC register 1 to clear this bit</p>
3	HSIRDYF	R	0	<p>HSI clock ready interrupt flag.</p> <p>Hardware sets this register when the HSI clock is stable and HSIRDYIE=1.</p> <p>0: HSI clock ready interrupt not generated; 1: HSI clock ready interrupt generation; Write HSIRDYC register 1 to clear this bit.</p>
2	Res	-	-	Res
1	LSERDYF	R	0	<p>LSERDY clock ready interrupt flag.</p> <p>Hardware sets this register when the LSE clock is stable and LSERDYIE=1.</p> <p>0: LSERDY clock ready interrupt not generated; 1: LSERDY clock ready interrupt generated; Write LSERDYC register 1 to clear this bit.</p>
0	LSIRDYF	R	0	<p>LSI Ready Interrupt Identifier Bit</p> <p>This bit is set by hardware when the LSI is stable and LSIRDYIE is enabled. Software clears this bit by setting the LSIRDYC bit.</p> <p>0: No clock ready interrupts caused by LSIs 1: There is a clock ready interrupt caused by the LSI Write LSIRDYC register 1 to clear this bit.</p>

8.5.8. Clock Interrupt Clear Register (RCC_CICR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res						LSE CSSC	CSSC	Res		PLL RDYC	HSE RDYC	HSI RDYC	Res	LSE RDYC	LSI RDYC
-						W	W	-		W	W	W	-	W	W

Bit	Name	R/W	Reset Value	Function
31: 10	Res	-	-	Res
9	LSECSSC	W	0	LSE Clock Safety System (CSS) interrupt flag cleared. 0: no effect 1: Clear the LSECSSF flag
8	CSSC	W	0	Clock safety interrupt clear bit. 0: no effect 1: Clear the CSSF flag bit
7: 6	Res	-	-	Res
5	PLLRDYC	W	0	PLL ready to interrupt flag cleared. 0: no effect 1: Clear the PLLRDYF flag
4	HSE RDYC	W	0	The HSE ready flag is cleared. 0: no effect 1: Clear the HSE RDYF bit
3	HSI RDYC	W	0	The HSI ready flag is cleared. 0: no effect 1: Clear the HSI RDYF bit
2	Res	-	-	Res
1	LSE RDYC	W	0	The LSE ready to interrupt flag is cleared. 0: no effect 1: Zeroing the LSE RDYF flag
0	LSI RDYC	W	0	The LSI ready flag is cleared. 0: no effect 1: Clear the LSI RDYF bit

8.5.9. I/O Interface Reset Register (RCC_IOPRSTR)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	GPIOF RST	Res	Res	GPIO CRST	GPIOB RST	GPIOA RST
-	-	-	-	-	-	-	-	-	-	RW	-	-	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 6	Res	-	-	Res
5	GPIOFRST	RW	0	I/O Port F reset. 0: no effect 1: Port F I/O Reset
4: 3	Res	-	-	Res
2	GPIOCRST	RW	0	I/O Port C reset. 0: no effect 1: Port C I/O Reset
1	GPIOBRST	RW	0	I/O Port B reset. 0: no effect 1: Port B I/O Reset
0	GPIOARST	RW	0	I/O Port A reset. 0: no effect 1: Port A I/O Reset

8.5.10. AHB Peripheral Reset Register (RCC_AHBRSTR)

Address offset: 0x28

Reset value: 0x0000 0000

This register is set and cleared by software. After the software is set, the module maintains the reset until the software clears the reset to zero.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	DIV RST	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	RW	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	CRC	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	DMA

			RST												RST
-	-	-	RW	-	-	-	-	-	-	-	-	-	-	-	RW

Bit	Name	R/W	Reset Value	Function
31: 25	Res	-	-	Res
24	DIVRST	RW	0	Divider module reset. 0: no effect 1: Divider module reset
23: 13	Res			Res
12	CRCRST	RW	0	CRC module reset. 0: no effect 1: CRC module reset
11: 9	Res	-	-	Res
8: 1	Res	-	-	Res
0	DMARST	RW	0	DMA Reset. 0: no effect 1: DMA module reset

8.5.11. APB Peripheral Reset Register 1 (RCC_APBSTR1)

Address offset: 0x2C

Reset value: 0x0000 0000

This register is set and cleared by software. After the software is set, the module maintains the reset until the software clears the reset to zero.

31	30	29	28	27	26	25	24	23	22	21	20
LPTIM RST	OPA RST	DAC RST	PWR RST	CTC RST	Res	CAN RST	Res	USB RST	I2C2 RST	I2C1 RST	Res
RW	RW	RW	RW	RW	-	RW	-	RW	RW	RW	-
19	18	17	16	15	14	13	12	11	10	9	8
USART4 RST	USART3 RST	USART2 RST	Res	Res	SPI2 RST	Res	Res	WWDG RST	RTC APB RST	Res	Res
RW	RW	RW	-	-	RW	-	-	RW	RW	-	-
7	6	5	4	3	2	1	0				
Res	Res	TIM7 RST	TIM6 RST	Res	Res	TIM3 RST	TIM2 RST				
-	-	RW	RW	-	-	RW	RW				

Bit	Name	R/W	Reset Value	Function
31	LPTIMRST	RW	0	LPTIM module reset. 0: no effect 1: The module resets
30	OPARST	RW	0	OPA module reset. 0: no effect 1: The module resets
29	DACRST	RW	0	DAC module reset. 0: no effect 1: The module resets
28	PWRRST	RW	0	Power interface module reset. 0: no effect 1: The module resets
27	CTCRST	RW	0	CTC module reset. 0: no effect 1: The module resets
26	Res	-	-	Res
25	CANRST	RW	0	CAN module reset. 0: no effect 1: The module resets
24	Res	-	-	Res
23	USBRST	RW	0	USB module reset. 0: no effect 1: The module resets
22	I2C2RST	RW	0	I2C2 module reset. 0: no effect 1: The module resets
21	I2C1RST	RW	0	I2C1 module reset. 0: no effect 1: The module resets
20	Res	-	-	Res
19	USART4RST	RW	0	USART4 module reset. 0: No effect; 1: The module resets
18	USART3RST	RW	0	USART3 module reset. 0: no effect 1: The module resets
17	USART2RST	RW	0	USART2 module reset. 0: no effect

Bit	Name	R/W	Reset Value	Function
				1: The module resets
16: 15	Res	-	-	Res
14	SPI2RST	RW	0	SPI2 module reset. 0: no effect 1: The module resets
13: 12	Res	-	-	Res
11	WWDGRST	RW	0	WWDG module reset. 0: no effect 1: The module resets
10	RTCAPBRST	RW	0	RTC module APB reset. 0: no effect 1: The module resets
9: 6	Res	-	-	Res
5	TIM7RST	RW	0	TIM7 module reset. 0: no effect 1: The module resets
4	TIM6RST	RW	0	TIM6 module reset. 0: No effect; 1: The module resets
3: 2	Res	-	-	Res
1	TIM3RST	RW	0	TIM3 module reset. 0: no effect 1: The module resets
0	TIM2RST	RW	0	TIM2 module reset. 0: no effect 1: The module resets

8.5.12. APB Peripheral Reset Register 2 (RCC_APBSTR2)

Address offset: 0x30

Reset value: 0x0000 0000

This register is set and cleared by software. After the software is set, the module maintains the reset until the software clears the reset to zero.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res								LCD RST	COMP3 RST	COMP2 RST	COMP1 RST	Res	TIM17 RST	TIM16 RST	TIM15 RST
-								RW	RW	RW	RW	-	RW	RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIM14	USART1	Res	SPI1	TIM1	MCUDBG	ADC	Res	Res							SYS

RST	RST		RST	RST	RST	RST			CFG
									RST
RW	RW	-	RW	RW	RW	RW		-	RW

Bit	Name	R/W	Reset Value	Function
31: 24	Res	-	-	Res
23	LCDRST	RW	0	LCD module reset. 0: no effect 1: The module resets
22	COMP3RST	RW	0	COMP3 module reset. 0: no effect 1: The module resets
21	COMP2RST	RW	0	COMP2 module reset. 0: no effect 1: The module resets
20	COMP1RST	RW	0	COMP1 module reset. 0: no effect 1: The module resets
19	Res	-	-	Res
18	TIM17RST	RW	0	TIM17 module reset. 0: no effect 1: The module resets
17	TIM16RST	RW	0	TIM16 module reset. 0: no effect 1: The module resets
16	TIM15RST	RW	0	TIM15 module reset. 0: no effect 1: The module resets
15	TIM14RST	RW	0	TIM14 module reset. 0: no effect 1: The module resets
14	USART1RST	RW	0	USART1 module reset. 0: no effect 1: The module resets
13	Res	-	-	Res
12	SPI1RST	RW	0	SPI1 module reset. 0: no effect 1: The module resets
11	TIM1RST	RW	0	TIM1 module reset.

Bit	Name	R/W	Reset Value	Function
				0: no effect 1: The module resets
10	MCUDBG_RST	RW	0	MCU Debug module reset. 0: no effect 1: The module resets
9	ADCRST	RW	0	ADC module reset. 0: no effect 1: The module resets
8: 1	Res	-	-	Res
0	SYSCFG_RST	RW	0	SYSCFG module reset. 0: no effect 1: The module resets

8.5.13. I/O interface clock enable register (RCC_IOPENR)

Address offset: 0x34

Reset value: 0x0000 0000

This register is set and cleared by software.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	GPIOF EN	Res	Res	GPIOC EN	GPIOB EN	GPIOA EN
-	-	-	-	-	-	-	-	-	-	RW	-	-	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 6	Res	-	-	Res
5	GPIOFEN	RW	0	I/O Port F Clock Enable. 0: Clock disabled 1: Clock Enable
4: 3	Res	-	-	Res
2	GPIOCEN	RW	0	I/O Port C Clock Enable. 0: Clock disabled 1: Clock Enable
1	GPIOBEN	RW	0	I/O Port B clock enable. 0: Clock disabled 1: Clock Enable
0	GPIOAEN	RW	0	I/O Port A clock enable.

				0: Clock disabled 1: Clock Enable
--	--	--	--	--------------------------------------

8.5.14. AHB Peripheral Clock Enable Register (RCC_AHBENR)

Address offset: 0x38

Reset value: 0x0000 0300

This register is set and cleared by software.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	DIVEN	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	RW	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	CRC EN	Res	Res	SRAM EN	flash EN	Res	Res	Res	Res	Res	Res	Res	DMA EN
-	-	-	RW	-	-	RW	RW	-	-	-	-	-	-	-	RW

Bit	Name	R/W	Reset Value	Function
31: 25	Res	-	-	Res
24	DIVEN	RW	0	Divider module clock enable. 0: Prohibited 1: Enabling
23: 13	Res	-	-	Res
12	CRCEN	RW	0	CRC module clock enable. 0: Prohibited 1: Enabling
11: 10	Res	-	-	Res
9	SRAMEN	RW	1	In Sleep mode, the SRAM clock enable control 0: In Sleep mode the module is clocked off 1: In Sleep mode, the module clock is enabled. Note: This bit only affects the clock enable of the module in Sleep mode; in Run mode, the module clock is not turned off.
8	FLASHEN	RW	1	Clock enable control for Flash in Sleep mode 0: In Sleep mode the module is clocked off 1: In Sleep mode, the module clock is enabled. Note: This bit only affects the clock enable of the module in Sleep mode; in Run mode, the module clock is not turned off.
7: 1	Res	-	-	Res
0	DMAEN	RW	0	DMA module clock enable.

Bit	Name	R/W	Reset Value	Function
				0: Prohibited 1: Enabling

8.5.15. APB Peripheral Clock Enable Register 1 (RCC_APBENR1)

Address offset: 0x3C

Reset value: 0x0000 0000

This register is set and cleared by software.

31	30	29	28	27	26	25	24	23	22	21
LPTIM EN	OPAEN	DACEN	PWR EN	CTCEN	Res	CANEN	Res	USBEN	I2C2EN	I2C1 EN
RW	RW	RW	RW	RW	-	RW	-	RW	RW	RW
20	19	18	17	16	15	14	13	12	11	10
Res	USART4E N	USART3E N	USART2E N	Res	Res	SPI2EN	Res	Res	WWDG EN	TIM- DIV_EN
-	RW	RW	RW	-	-	-	RW	-	-	RW
9	8	7	6	5	4	3	2	1	0	
Res	Res	Res	Res	TIM7EN	TIM6EN	Res	Res	TIM3 EN	TIM2EN	
-	RW	-	-	RW	RW	-	-	RW	RW	

Bit	Name	R/W	Reset Value	Function
31	LPTIMEN	RW	0	LP Timer1 module clock enable. 0: Prohibited 1: Enabling
30	OPAEN	RW	0	OPA module clock enable. 0: Prohibited 1: Enabling
29	DACEN	RW	0	DAC module clock enable. 0: Prohibited 1: Enabling
28	PWREN	RW	0	Low power control module clock enable. 0: Prohibited 1: Enabling
27	CTCEN	RW	0	CTC module clock enable. 0: Prohibited 1: Enabling
26	Res	-	-	Res
25	CANEN	RW	0	CAN module clock enable.

Bit	Name	R/W	Reset Value	Function
				0: Prohibited 1: Enabling
24	Res	-	-	Res
23	USBEN	RW	0	USB module clock enable. 0: Prohibited 1: Enabling
22	I2C2EN	RW	0	I2C2 module clock enable. 0: Prohibited 1: Enabling
21	I2C1EN	RW	0	I2C1 module clock enable. 0: Prohibited 1: Enabling
20	Res	-	-	Res
19	USART4EN	RW	0	USART4 module clock enable. 0: Prohibited 1: Enabling
18	USART3EN	RW	0	USART3 module clock enable. 0: Prohibited 1: Enabling
17	USART2EN	RW	0	USART2 module clock enable. 0: Prohibited 1: Enabling
16: 15	Res	-	-	Res
14	SPI2EN	RW	0	SPI2 module clock enable. 0: Prohibited 1: Enabling This register is cleared by a hardware system reset.
13: 12	Res	-	-	Res
11	WWDGEN	RW	0	Window WDG module clock enable. 0: Prohibited 1: Enabling This register is cleared by a hardware system reset.
10	TIMDIV_EN	RW	0	TIMER PCLK frequency control. 0: TIMER PCLK is system PCLK*2, but the frequency will not exceed HCLK 1: TIMER PCLK for system PCLK*1
9: 6	Res	-	-	Res
5	TIM7EN	RW	0	TIM7 module clock enable.

Bit	Name	R/W	Reset Value	Function
				0: Prohibited 1: Enabling
4	TIM6EN	RW	0	TIM6 module clock enable. 0: Prohibited 1: Enabling
3: 2	Res	-	-	Res
1	TIM3EN	RW	0	TIM3 module clock enable. 0: Prohibited 1: Enabling
0	TIM2EN	RW	0	TIM2 module clock enable. 0: Prohibited 1: Enabling

8.5.16. APB Peripheral Clock Enable Register 2 (RCC_APBENR2)

Address offset: 0x40

Reset value: 0x0000 0001

This register is set and cleared by software.

31	30	29	28	27	26	25	24	23	22	21
Res	Res	Res	Res	Res	Res	Res	Res	LCDEN	COMP3EN	COMP2EN
-	-	-	-	-	-	-	-	RW	RW	RW
20	19	18	17	16	15	14	13	12	11	10
COMP1EN	Res	TIM17 EN	TIM16 EN	TIM15 EN	TIM14 EN	USART1E N	Res	SPI1EN	TIM1EN	MCUD- BGEN
RW	-	RW	RW	RW	RW	RW	-	RW	RW	RW
9	8	7	6	5	4	3	2	1	0	
ADCEN	Res	Res	Res	Res	Res	Res	Res	Res	SYSCFGEN	
RW	-	-	-	-	-	-	-		RW	

Bit	Name	R/W	Reset Value	Function
31: 24	Res	-	-	Res
23	LCDEN	RW	0	LCD module clock enable. 0: Prohibited 1: Enabling
22	COMP3EN	RW	0	COMP3 module clock enable. 0: Prohibited 1: Enabling
21	COMP2EN	RW	0	COMP2 module clock enable. 0: Prohibited

Bit	Name	R/W	Reset Value	Function
				1: Enabling
20	COMP1EN	RW	0	COMP1 module clock enable. 0: Prohibited 1: Enabling
19	Res	-	-	Res
18	TIM17EN	RW	0	TIM17 module clock enable. 0: Prohibited 1: Enabling
17	TIM16EN	RW	0	TIM16 module clock enable. 0: Prohibited 1: Enabling
16	TIM15EN	RW	0	TIM15 module clock enable. 0: Prohibited 1: Enabling
15	TIM14EN	RW	0	TIM14 module clock enable. 0: Prohibited 1: Enabling
14	USART1EN	RW	0	USART1 module clock enable. 0: Prohibited 1: Enabling
13	Res	-	-	Res
12	SPIEN	RW	0	SPI1 module clock enable. 0: Prohibited 1: Enabling
11	TIM1EN	RW	0	TIM1 module clock enable. 0: Prohibited 1: Enabling
10	MCUDBGEN	RW	0	MCUDBG module clock enable. 0: Prohibited 1: Enabling
9	ADCEN	RW	0	ADC module clock enable. 0: Prohibited 1: Enabling
8: 1	Res	-	-	Res
0	SYSCFGEN	RW	1	SYSCFG module clock enable. 0: Prohibited 1: Enabling

8.5.17. Peripheral Independent Clock Configuration Register (RCC_CCIPR)

Address offset: 0x54

Reset value: 0x0000 0000

This register is set and cleared by software.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	LPTIM1SEL [1: 0]		Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	RW	RW	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res		Res	COMP3 SEL	COMP2 SEL	COMP1 SEL	PVD SEL	CAN SEL	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	RW	RW	RW	RW	RW	-	-	-	-	-	-	-

Bit	Name	R/W	Reset Value	Function
31: 20	Res	-	-	Res
19: 18	LPTIMSEL[1:0]	RW	2'b00	LPTIM1 internal clock source selection. 00: PCLK01: LSI10: No clock11: LSE
17: 11	Res	-	-	Res
10	COMP3SEL	RW	0	COMP3 module clock source selection. 0: PCLK 1: LSC (clock after RCC_BDCR.LSCOSEL selection) Note: Configure the selection of the LSC clock before enabling COMP3_FR.FLTEN.
9	COMP2SEL	RW	0	COMP2 module clock source selection. 0: PCLK 1: LSC (clock after RCC_BDCR.LSCOSEL selection) Note: Configure the selection of the LSC clock before enabling COMP2_FR2.FLTEN.
8	COMP1SEL	RW	0	COMP1 module clock source selection. 0: PCLK 1: LSC (clock after RCC_BDCR.LSCOSEL selection) Note: Configure this register to select the clock before enabling COMP1_FR1.FLTEN.
7	PVDSEL	RW	0	PVD detect clock source selection. 0: PCLK 1: LSC (clock after RCC_BDCR.LSCOSEL selection) Note: When PCLK is selected for the clock source, PWR_CR1.FLTEN needs to be configured to 0. If

Bit	Name	R/W	Reset Value	Function
				FLTEN is enabled, the LSC clock must be selected and the LSC clock must be configured to be selected before FLTEN is enabled.
6	CANSEL	RW	0	CAN module clock source selection. 0: PLL 1: HSE
5: 0	Res	-	-	Res

8.5.18. RTC Domain Control Register (RCC_BDCR)

Address offset: 0x5C

Reset value: 0x0000 0000, reset by POR/BOR

When this register is accessed consecutively, $0 \leq \text{wait state} \leq 3$.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res						LSC SEL	LSC OEN	Res	Res	Res	Res	Res	Res	Res	BDRST
-						RW	RW	RW	-	-	-	-	-	-	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTC EN	Res					RTCSEL [1:0]		Res	LSE CSS D	LSEC SSON	Res	Res	LSE- BYP	LSE- DY	LSEON
RW	-					RW	RW	-	R	RW	-	-	RW	R	RW

Bit	Name	R/W	Reset Value	Function
31: 26	Res	-	-	Res
25	LSCOSEL	RW	0	Low-speed clock selection. 0: LSI 1: LSE
24	LSCOEN	RW	0	Low-speed clock enable. 0: Prohibited 1: Enabling
23: 17	Res	-	-	Res
16	BDRST	RW	0	RTC domain soft reset. 0: no effect 1: Reset
15	RTCEN	RW	0	RTC Clock Enable. Software setting or clearing. 0: Prohibited 1: Enabling
14: 10	Res	-	-	Res

Bit	Name	R/W	Reset Value	Function
9: 8	RTCSEL [1:0]	RW	0	<p>RTC clock source selection.</p> <p>00: No clock01: LSE10: LSI11: HSE 128 divisions</p> <p>Once the RTC clock source is selected it cannot be changed except in the following cases:</p> <ol style="list-style-type: none"> 1. RTC domain is reset to 00 2. Select LSE (LSECSSD=1) but no LSE
7	Res	-	-	Res
6	LSECSSD	R	0	<p>CSS failed to detect LSE.</p> <p>This bit is set by hardware to indicate that the CSS has failed to detect the 32.768 kHz OSC (LSE).</p> <p>0: LSE failure not detected</p> <p>1: Failure to detect LSE</p>
5	LSECSSON	RW	0	<p>The CSS enables the LSE clock.</p> <p>0: Prohibited;</p> <p>1: Enable;</p> <p>Note: LSEON=1 and LSE RDY=1 must be present to enable LSECSSON.</p> <p>Once this bit is enabled, it cannot be disabled unless LSECSSD=1.</p>
4: 3	Res	-	-	Res
2	LSEBYP	RW	0	<p>LSE OSC bypass</p> <p>0: no bypass, low-speed external clock selects crystal; 1: bypass, low-speed external clock selects external interface input clock;</p> <p>Note: This bit can only be written if the external 32.768 kHz OSC is disabled (LSEON=0 and LSE RDY=0).</p>
1	LSE RDY	R	0	<p>LSE OSC ready.</p> <p>Hardware configuration of this bit to 1 indicates that the LSE clock is ready.</p>
0	LSEON	RW	0	<p>LSE OSC Enable.</p> <p>0: Prohibited</p> <p>1: Enabling</p>

8.5.19. Control/Status Register (RCC_CSR)

Address offset: 0x60

Reset value: 0x0800 0000

The reset flag bit in this register can only be reset by power reset, the others are reset by the system.

When this register is accessed consecutively, $0 \leq \text{wait state} \leq 3$.

31	30	29	28	27	26	25	24	23	22	21
Res	WWDG RSTF	IWDG RSTF	SFT RSTF	PWR RSTF	PIN RSTF	OBL RSTF	Res	RMVF	Res	Res
-	R	R	R	R	R	R	-	RW	-	-
20	19	18	17	16	15	14	13	12	11	
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	
-	-	-	-	-	-	-	-	-	-	
10	9	8	7	6	5	4	3	2	1	0
Res	Res	PINRST _FLTDIS	Res	Res	Res	Res	Res	Res	LSI RDY	LSION
-	-	RW	-	-	-	-	-	-	R	RW

Bit	Name	R/W	Reset Value	Function
31	Res	-	-	Res
30	WWDGRSTF	R	0	Window WDG reset flag. RMVF set to 1 clears this bit.
29	IWDGRSTF	R	0	IWDG reset flag. RMVF set to 1 clears this bit.
28	SFTRSTF	R	0	Soft reset flag. RMVF set to 1 clears this bit.
27	PWRRSTF	R	1	BOR/POR/PDR reset flag. RMVF set to 1 clears this bit.
26	PINRSTF	R	0	External NRST pin reset flag. RMVF set to 1 clears this bit.
25	OBLRSTF	R	0	Option byte loader reset flag. RMVF set to 1 clears this bit.
24	Res	-	-	Res
23	RMVF	RW	0	A software set 1 is required to clear the [30:25] reset flag.
22: 9	Res	-	-	Res
8	PINRST_FLTDIS	RW	0	NRST filter width 40 μ s prohibited 0: HSI_10M is enabled and the filter 40 μ s width function is enabled. 1: Filter function disabled, synchronized to system clock to generate system reset
7: 2	Res	-	-	Res
1	LSIRDY	R	0	LSI OSC stabilization flags. 0: LSI not stabilized 1: LSI has been stabilized
0	LSION	RW	0	LSI OSC Enable.

Bit	Name	R/W	Reset Value	Function
				0: Prohibited 1: Enabling Hardware turns on the analog LSI case: 1. Hardware IWDG Enable 2. LSECSS Enable

9. Clock Calibration Controller (CTC)

9.1. Introduction

The Clock Calibration Controller (CTC) uses hardware to automatically calibrate the RC crystal (HSI) when the internal configuration is 16 MHz, and the PLL (48 MHz) after 3x frequency is used as the USB module clock source, and the PLL generated by configuring the HSI to 16 MHz and after 3x frequency is abbreviated as PLL48M below. For the clock source of the USB module, the PLL48M clock frequency must be required to be in the range of $48 \text{ MHz} \pm 500 \text{ ppm}$, but an internal crystal that is not calibrated is not capable of meeting such a high level of accuracy. The CTC module calibrates the HSI's clock frequency based on an external high-precision reference signal source, and adjusts the calibration value automatically or manually to obtain an accurate PLL48M clock.

9.2. CTC Main Characteristics

- Three external reference signal sources: GPIO, LSE clock, USBD_SOF
- Provides software reference synchronization pulses;
- Hardware auto-calibration, no software operation required;
- 16 bits calibrated counter with reference source capture and reload functions;
- 8 bits clock calibration base for frequency evaluation and auto-calibration;
- Flag bits and interrupts to indicate the status of the clock calibration: calibration success status (CKOKIF), warning status (CKWARNIF) and error status (ERRIF).

9.3. CTC Functional Description

9.3.1. CTC Block Diagram

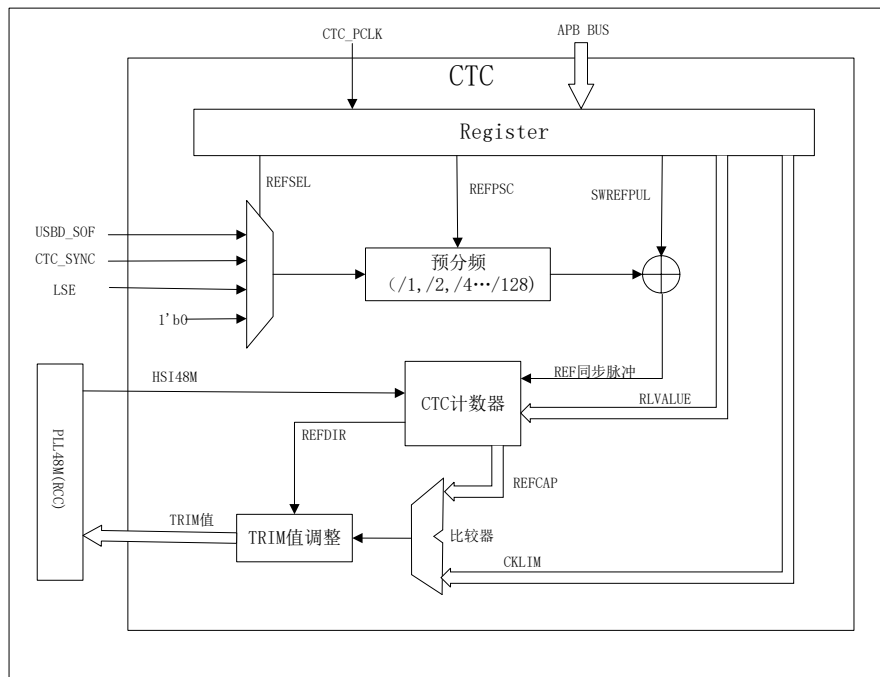


Figure 9-1 CTC architecture block diagram

9.3.2. REF Synchronized Pulse Generator

First, the reference signal source is selected by setting the REFSEL bit in the CTC_CTL1 register: the GPIO, the LSE clock output or USBD_SOF.

The signal polarity during synchronization of the reference source can then be configured by setting the REFPOL bit in the CTC_CTL1 register, and an appropriate synchronization clock frequency signal (no greater than 48 kHz) can be generated by setting the REFPSC bit in the CTC_CTL1 register.

If a software reference pulse signal is required, the SWREFPUL bit in the CTC_CTL0 register needs to be set to one. The software reference pulse signal and the external reference pulse signal are finally subjected to a logical 'or' operation.

9.3.3. CTC Calibration Counter

The CTC clock calibration counter is clocked by the PLL48M. After setting the CNTEN bit in the CTC_CTL0 register, when the first REF synchronization pulse signal is detected, the counter starts counting down from the RLVALUE value (RLVALUE is defined in the CTC_CTL1 register). Each time the REF synchronization pulse signal is detected, the counter reloads the RLVALUE value and also restarts counting down. If the REF sync pulse signal is never detected, the counter counts down to zero, then up to $128 \times \text{CKLIM}$ (CKLIM is defined in CTC_CTL1), and finally stops until the next REF sync pulse signal is detected. Once the REF synchronization pulse signal is detected, the count value of the current CTC calibration counter is captured and deposited into the REFCAP bit in CTC_SR, while the count direction of the current counter is deposited into the REFDIR bit in CTC_SR. Details are shown below.

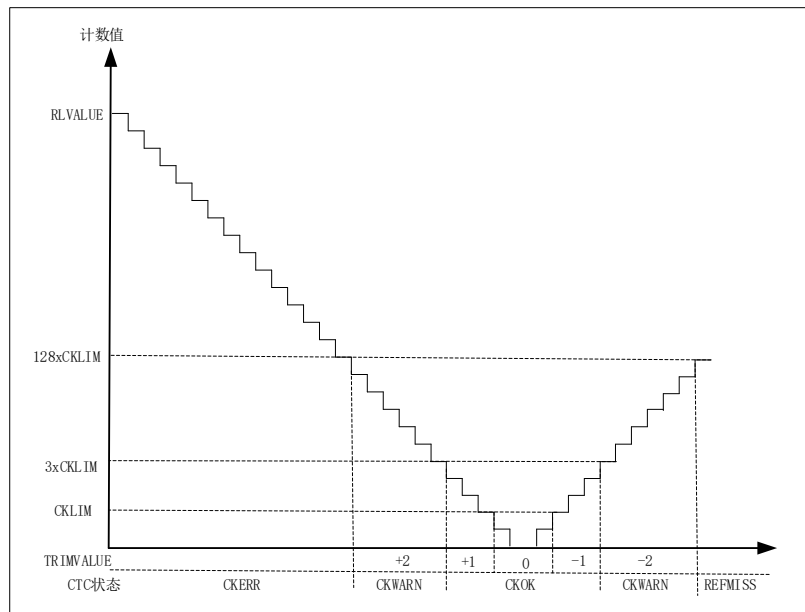


Figure 9-2 CTC Calibration Counter

9.3.4. Frequency evaluation and automatic calibration process

The clock frequency evaluation function starts to execute when the REF synchronization pulse signal appears. If the REF synchronization pulse signal appears while the counter is counting down, it means that the current clock frequency is slower than the desired clock frequency (frequency of 48MHz), and the TRIMVALUE value (clock calibration value) in CTC_CTL0 needs to be increased. If the REF synchronization pulse signal occurs while the counter is counting up, it means that the current clock frequency is faster than the desired clock frequency and the TRIMVALUE value needs to be reduced. The CKOKIF bit, the CKWARNIF bit, the CKERR bit and the REFMIS bit in CTC_SR reflect the status of the frequency evaluation.

If AUTOTRIM (Hardware Auto Calibration Mode) position 1 in CTC_CTL0, Hardware Auto Calibration Mode is enabled. In this mode, if the REF synchronization pulse signal appears during the downward count of the counter, indicating that the current clock frequency is slower than the desired clock frequency, the TRIMVALUE value in CTC_CTL0 is automatically increased to increase the current clock frequency. Conversely, if the REF synchronization pulse signal occurs while the counter is counting up, indicating that the current clock frequency is faster than the desired clock frequency, the TRIMVALUE value is automatically reduced, thus decreasing the current clock frequency.

- The REF synchronization pulse signal is detected when $\text{Counter} < \text{CKLIM}$:
The CKOKIF bit (Clock Calibration Success Flag Bit) in CTC_SR is set, and at the same time, if the CKOKIE bit (Clock Calibration Completed Interrupt Enable Bit) in CTC_CTL0 is set to 1, an interrupt will be generated. If AUTOTRIM in CTC_CTL0 is set to 1, the TRIMVALUE value in CTC_CTL0 remains unchanged.
- The REF synchronization pulse signal is detected when $\text{CKLIM} \leq \text{Counter} < 3 \times \text{CKLIM}$:
The CKOKIF bit in CTC_SR is set, and at the same time, an interrupt will be generated if the CKOKIE bit in CTC_CTL0 is 1. If the AUTOTRIM position in CTC_CTL0 is 1, the TRIMVALUE value in CTC_CTL0 will be increased by 1 during the counter down count and decreased by 1 during the count up count.
- The REF synchronization pulse signal is detected when $3 \times \text{CKLIM} \leq \text{Counter} < 128 \times \text{CKLIM}$:
The CKWARNIF bit (Clock Calibration Warning Interrupt Bit) in CTC_SR is set, and at the same time, if the CKWARNIE bit (Clock Calibration Warning Interrupt Enable Bit) in CTC_CTL0 is set to 1, an interrupt will be generated. If the AUTOTRIM position in CTC_CTL0 is 1, the TRIMVALUE value in CTC_CTL0 will be increased by 2 during the downward counter count and decreased by 2 during the upward count.
- $\text{Counter} \geq 128 \times \text{CKLIM}$, the counter detects the REF synchronization pulse signal during the downward count:
The CKERR bit (clock calibration error bit) in CTC_SR is set, and an interrupt will be generated if the ERRIE bit (error interrupt enable bit) in CTC_CTL0 is set to 1. The TRIMVALUE value in CTC_CTL0 is unchanged.
- $\text{Counter} = 128 \times \text{CKLIM}$, the counter is in the process of counting up:

The REFMISS bit (REF Synchronization Pulse Loss Bit) in CTC_SR is set, and at the same time, an interrupt will be generated if the ERRIE bit in CTC_CTL0 is 1. The TRIMVALUE value in CTC_CTL0 is unchanged.

If the calibrated value of TRIMVALUE in CTC_CTL0 is greater than 127, an overflow event will occur, and at the same time, if the calibrated value of TRIMVALUE is less than 0, an underflow event will occur. The value of TRIMVALUE ranges from 0 to 127 (when an overflow event occurs, the TRIMVALUE value is 127; when an underflow event occurs, the TRIMVALUE value is 0). The TRIMERR bit (Calibration Value Error bit) in CTC_SR will then be set, and an interrupt will be generated if the ERRIE bit in CTC_CTL0 is set to one.

9.3.5. Software Programming Guide

The RLVALUE bit and the CKLIM bit in CTC_CTL1 are key to clock frequency evaluation and hardware auto-calibration. Their values are calculated from the frequency of the desired clock (PLL: 48 MHz) and the frequency of the REF synchronization pulse signal. The ideal state is that the REF synchronization pulse signal occurs when the CTC counter counts to zero, so the value of RLVALUE is:

$$RLVALUE = (f_{clock} \div f_{REF}) - 1$$

The value of CKLIM is set by the user according to the accuracy of the clock, and it is generally recommended to set it to half of the step size, so the value of CKLIM is:

$$CKLIM = (f_{clock} \div f_{REF}) \times 0.12\% \div 2$$

Typical step values are 0.12%, f_{clock} is the frequency of the desired clock (48 MHz), and f_{REF} is the frequency of the REF synchronization pulse signal.

TRIMVALUE in CTC_CTL0 can be written by software when the AUTOTRIM bit is 0. However, modification of TRIMVALUE directly affects the frequency of the HSI clock, and therefore, TRIMVALUE should not be arbitrarily modified by software. It is recommended that the user modifies it in the middle of two reference signals according to the flag bit judgment (see Frequency Evaluation and Auto-calibration Procedure for details); or if a reliable value already exists, the user can directly modify the value of HSI_TRIM in RCC_ICSCR.

9.4. CTC Register

9.4.1. CTC control register 0 (CTC_CTL0)

Address offset: 0x00

Reset value: 0x0000 4000

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

Res	TRIMVALUE[6:0]							SWR EFP UL	AU- TO- TRIM	CNT EN	Res	ERE- FIE	ERRI E	CKW AR- NIE	CKO KIE
-	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	-	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 15	Res	-	-	Res
14: 8	TRIMVALUE[6:0]	RW	7'b1000000	<p>PLL48M Calibration Values.</p> <p>This bit is set and cleared by software when the AUTO-TRIM value in CTC_CTL0 is 0. This mode is used for the software calibration process.</p> <p>When the AUTOTRIM value in CTC_CTL0 is 1, this bit is read-only and is automatically modified by the hardware; this mode is used for the hardware calibration process.</p> <p>The middle value of TRIMVALUE is 64, and when the TRIMVALUE value is increased by 1, the PLL48M clock frequency is increased by approximately 48 kHz. When the TRIMVALUE value is decremented by 1, the PLL48M clock frequency is reduced by approximately 48 kHz.</p>
7	SWREFPUL	RW	0	<p>Software generates synchronized reference signal pulses.</p> <p>This bit is set by software and provides a synchronized reference pulse signal for the CTC counter. This bit is automatically cleared by hardware and returns 0 for read operations.</p> <p>0: No effect; 1: The software generates a synchronized reference pulse signal;</p>
6	AUTOTRIM	RW	0	<p>Hardware auto-calibration mode.</p> <p>This bit is set or cleared by software. When this position is 1, the hardware auto-calibration mode is enabled, and the TRIMVALUE value in CTC_CTL0 is continuously and automatically modified by hardware until the clock frequency of the PLL48M reaches 48 MHz.</p> <p>0: Disable hardware auto-calibration mode 1: Enable hardware auto-calibration mode</p>
5	CNTEN	RW	0	<p>CTC Counter Enable.</p> <p>This bit is set or cleared by software to enable or disable the CTC counter. When this position is 1, the value of CTC_CTL1 cannot be modified.</p>

				0: Disable CTC counter 1: Enable CTC counter
4	Res	-	-	Res
3	EREFIE	RW	0	Expected reference signal interrupt enable. 0: Disable interrupt generation from the desired reference signal 1: Enable the desired reference signal to generate an interrupt
2	ERRIE	RW	0	Error interrupt enable. 0: Disable error interrupt 1: Enable error interrupt
1	CKWARNIE	RW	0	Clock calibration warning interrupt enable. 0: Disable clock calibration warning interrupt 1: Enable clock calibration warning interrupt
0	CKOKIE	RW	0	Clock calibration completion interrupt enable. 0: Disable clock calibration completion interrupt 1: Enable clock calibration completion interrupt

9.4.2. CTC control register 1 (CTC_CTL1)

Address offset: 0x04

Reset value: 0x2022 BB7F

This register can only be accessed by word (32-bit). When CNTEN is 1, the value of this register cannot be modified.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REF POL	Res	REFSEL [1:0]		Res	REFPSC [2:0]			CKLIM [7:0]							
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
RLVALUE [15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31	REFPOL	RW	0	Reference signal source polarity. This bit is set or cleared by software to select the synchronization polarity of the reference signal source. 0: Select rising edge 1: Select falling edge
30	Res	-	-	Res
29: 28	REFSEL [1:0]	RW	2'b10	Reference signal source selection.

				<p>This bit is set or cleared by software to select the reference signal source.</p> <p>00: Select GPIO input signal</p> <p>01: Select LSE Clock</p> <p>10: Select USB_D_SOF signal</p> <p>11: Reservation, option 0</p>
27	Res	-	-	Res
26: 24	REFPSC [2:0]	RW	3'b000	<p>Reference signal source prescaler.</p> <p>This bit is set or cleared by software.</p> <p>000: Reference signal not crossover</p> <p>001: Reference signal 2 divisions</p> <p>010: Reference signal 4 divisions</p> <p>011: Reference signal 8 divisions</p> <p>100: Reference signal 16 divisions</p> <p>101: Reference signal 32 crossover frequency</p> <p>110: Reference signal 64 crossover frequency</p> <p>111: Reference signal 128 crossover frequency</p>
23: 16	CKLIM [7:0]	RW	0x22	<p>Clock calibration time base limits.</p> <p>This bit is set or cleared by software to define the clock calibration time base limit. This bit is used for frequency evaluation and automatic calibration processes.</p>
15: 0	RLVALUE [15:0]	RW	0xBB7F	<p>CTC Counter reload value.</p> <p>This bit is set or cleared by software to define the CTC counter reload value that will be reloaded into the CTC calibration counter when a synchronization reference pulse is detected.</p>

9.4.3. CTC Status Register (CTC_SR)

Address offset: 0x08

Reset value: 0x0000 8000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REFCAP [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
REF DIR	Res	Res	Res	Res	TRIM ERR	REF MISS	CKE RR	Res	Res	Res	Res	ERE-FIF	ERRIF	CKW AR-NIF	CKO KIF
R	-	-	-	-	R	R	R	-	-	-	-	R	R	R	R

Bit	Name	R/W	Reset Value	Function
31: 16	REFCAP [15:0]	R	0	CTC Counter Capture Value. When a synchronized reference pulse signal is detected, the count value in the CTC calibration counter is deposited into the REFCAP bit.
15	REFDIR	R	1	CTC Calibrates the clock count direction. When a synchronized reference pulse signal is detected, the count direction of the CTC calibration counter is stored in the REFDIR bit. 0: count up 1: Counting down
14: 11	Res	-	-	Res
10	TRIMERR	R	0	Calibration value error bit. This bit is set by hardware when an overflow or underflow of the TRIMVALUE value in CTC_CTL0 occurs. If ERRIE in CTC_CTL0 is position 1, an interrupt is generated. The TRIMERR bit can be cleared to zero by writing a 1 to the ERRIC bit in CTC_INTC. 0: No calibration value error occurs 1: Calibration value error occurred
9	REFMISS	R	0	Synchronization reference pulse signal is lost. This bit is set by hardware when the synchronization reference pulse signal is lost. REFMISS position bit when the CTC calibration counter does not detect a synchronization reference pulse signal for 128 x CKLIM counted during incremental counting. Indicates that the current clock is too fast to calibrate to the desired frequency value, or some other error has been generated. The REFMISS bit can be cleared to zero by writing a 1 to the ERRIC bit in CTC_INTC. 0: No synchronization reference pulse signal loss 1: Loss of synchronization reference pulse signal Note: To prevent the flag bit from being set again after clearing REFMISS and the concurrent ERRIF, the CNTEN bit can be cleared and the ERRIC bit can be written repeatedly until the flag bit is no longer set.
8	CKERR	R	0	Clock calibration error bit. This bit is set by hardware when a clock calibration error is generated. When the CTC Calibration Counter count value is greater than or equal to 128 x CKLIM during a

Bit	Name	R/W	Reset Value	Function
				<p>decrement count and a synchronization reference pulse signal is detected, CKERR is set, indicating that the current clock is too slow to calibrate to the desired frequency value. An interrupt is generated when ERRIE in CTC_CTL0 is set to 1. The CKERR bit can be cleared to zero by writing a 1 to the ERRIC bit in CTC_INTC.</p> <p>0: No clock calibration error occurs 1: Clock calibration error occurred</p>
7: 4	Res	-	-	Res
3	EREFIF	R	0	<p>Expected reference interrupt flag bit.</p> <p>When the CTC calibration clock counter counts to 0, a reference signal is detected and this bit is set by hardware. An interrupt is generated when EREFIE in CTC_CTL0 is set to 1. The EREFIF bit can be cleared to zero by writing a 1 to the EREFIC bit in CTC_INTC.</p> <p>0: No desired reference signal generation 1: Expected reference signal generation</p>
2	ERRIF	R	0	<p>Error interrupt flag bit.</p> <p>This bit is set by hardware when an error occurs. This position bit whenever a TRIMERR, REFMIS or CKERR error occurs. An interrupt is generated when ERRIE in CTC_CTL0 is set. The ERRIF bit can be cleared to zero by writing a 1 to the ERRIC bit in CTC_INTC.</p> <p>0: No error occurred 1: An error occurred</p>
1	CKWARNIF	R	0	<p>Clock calibration warning interrupt flag bit.</p> <p>This bit is set by hardware when a clock calibration warning is generated. CKWARNIF is set when the CTC Calibration Counter count value is greater than or equal to 3xCKLIM and less than 128xCKLIM and a synchronized reference pulse signal is detected. This indicates that the current clock frequency is too slow or too fast, but can be calibrated to achieve the desired frequency value. When a clock calibration warning is generated, the TRIMVALUE value is increased or decreased by 2.</p> <p>An interrupt is generated when CKWARNIE in CTC_CTL0 is set to 1. The CKWARNIF bit can be cleared to zero by writing a 1 to the CKWARNIC bit in CTC_INTC.</p>

Bit	Name	R/W	Reset Value	Function
				0: No clock calibration warning occurs 1: A clock calibration warning occurs
0	CKOKIF	R	0	<p>Clock calibration success interrupt flag bit.</p> <p>This bit is set by hardware when the clock calibration is successful. If a synchronization reference pulse signal is detected when the CTC calibration counter count value is less than 3 x CKLIM, CKOKIF is set. Indicates that the current clock frequency is normal and can be used without clock calibration by the TRIMVALUE value. An interrupt is generated when CKOKIE in CTC_CTL0 is set to 1. The CKOKIF bit can be cleared to zero by writing a 1 to the CKOKIC bit in CTC_INTC.</p> <p>0: Clock calibration unsuccessful 1: Clock calibration successful</p>

9.4.4. CTC interrupt clear register (CTC_INTC)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	ERE-FIC	ERRIC	CKWARNIC	CKOKIC
-	-	-	-	-	-	-	-	-	-	-	-	W	W	W	W

Bit	Name	R/W	Reset Value	Function
31: 4	Res	-	-	Res
3	EREFIC	W	0	<p>EREFIF Interrupt clear bit.</p> <p>This bit can only be written by software; read operations return 0. A write of 1 clears the EREFIF bit in CTC_SR; a write of 0 has no effect.</p>
2	ERRIC	W	0	<p>ERRIF Interrupt clear bit.</p> <p>This bit can only be written by software; read operations return 0. A write of 1 clears the ERRIF bit, the TRIMERR bit, the REFMIS bit, and the CKERR bit in CTC_SR; a write of 0 has no effect.</p>
1	CKWARNIC	W	0	CKWARNIF interrupt clear bit.

				This bit can only be written by software; read operations return 0. A write of 1 clears the CKWARNIF bit in CTC_SR; a write of 0 has no effect.
0	CKOKIC	W	0	CKOKIF Interrupt clear bit. This bit can only be written by software; read operations return 0. Writing 1 clears the CKOKIF bit in CTC_SR, writing 0 has no effect.

10. General Purpose I/O (GPIO)

10.1. Introduction

The GPIO contains PA[15:0], PB[15:0], PC[15:0] and PF[9:0] for each GPIO port:

- Four 32-bit configuration registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR)
- Two 32-bit data registers (GPIOx_IDR and GPIOx_ODR)
- 1 32-bit bit/reset register (GPIOx_BSRR)
- 1 32-bit latched register (GPIOx_LCKR)
- 2 multiplexing function selection registers (GPIOx_AFRH and GPIOx_AFRL)
- 1 32-bit reset register (GPIOx_BRR)

10.2. General Purpose IO Functional Description

- Registers support Fast IO / AHB bus reads and writes
- Output state: push-pull or open-drain + pull-up/pull-down
- Data register (GPIOx_ODR) or peripheral (multiplexed function output) data outputs
- Speed selectable per I/O
- Input states: float, pull-up/down, analog
- Data input to input data register (GPIOx_IDR) or peripheral (multiplexed function input)
- Position Bit/Reset Register (GPIOx_BSRR), allows bit write access to GPIOx_ODR
- The locking mechanism (GPIOx_LCKR) freezes the I/O port configuration function
- analog function
- Multiplexing function selection register (up to 16 multiplexing functions per IO port)
- Ability to flip quickly in a single cycle
- Highly flexible I/O multiplexing enables I/O ports to function as GPIOs or as interfaces to various peripherals.

10.3. General Purpose IO Functional Description

Each bit of each GPIO can be configured in several modes through software programming:

- Input Float
- Input pull-up
- Input Dropdown
- analog input
- Open-drain output with pull-up or pull-down
- Push-pull output with pull-up or pull-down
- Push-pull with pull-up or pull-down multiplexing.
- Open drain with pull-down or pull-up multiplexing

Each I/O port can be freely programmed; however, the I/O port registers must be accessed as full words, half words, or bytes. The GPIOx_BSRR and GPIOx_BRR registers allow independent access

for read/change of any GPIOx_ODR register. This way, there is no danger of generating an IRQ between read and change accesses.

The following figure gives the basic structure of an I/O port (1bit)

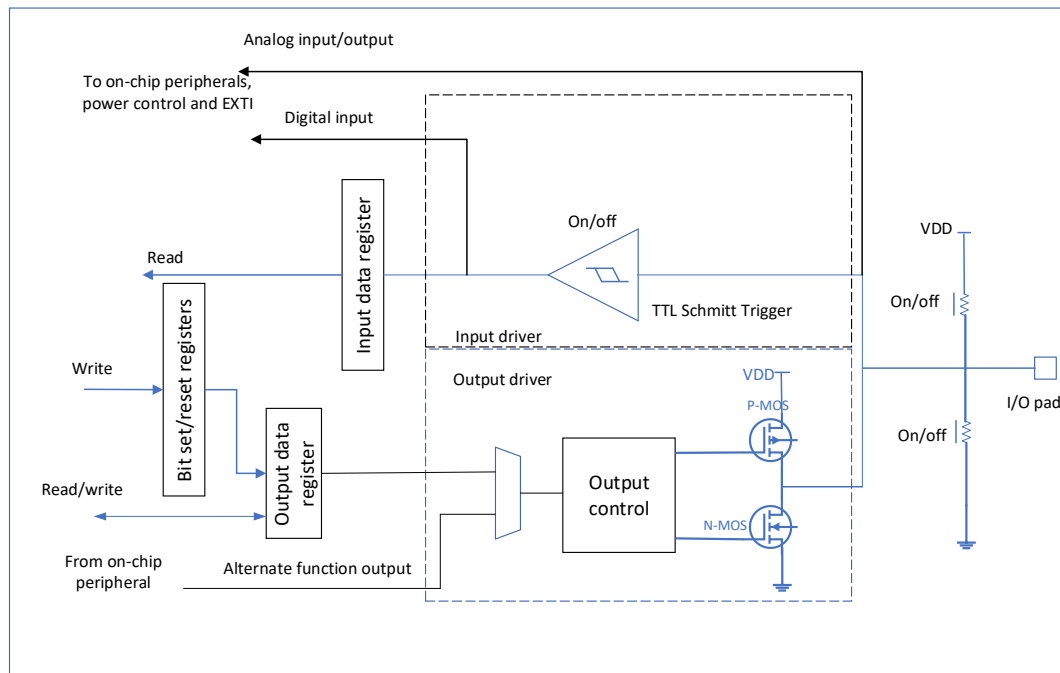


Figure 10-1 Basic structure of IO port bits

10.3.1. General Purpose I/O (GPIO)

During and after reset, the multiplexing function is not activated and most of the IOs are configured in analog mode.

The Debug pin is by default placed in multiplexed function pull-up or pull-down mode:

- PA14-SWCLK: placed in pull-down mode
- PA13-SWDIO: placed in pull-up mode

Boot pin is placed in input mode by default, pull-down mode

- PF8-Boot: placed in drop-down mode

When the pin is configured as an output, the value written to the output data register (GPIOx_ODR) will be output on the I/O pin. The output driver can be used in push-pull mode or open-drain mode (only low levels can be driven, high levels are highly resistive).

The input data register (GPIOx_IDR) will acquire the level on the I/O pin at each AHB clock.

All GPIO pins have internal weak pull-up and weak pull-down resistors, which can be used to enable or disable the function via the GPIOx_PUPDR register.

10.3.2. I/O pin-multiplexing function multiplexing and mapping

Device I/O ports are connected to board-level peripherals/modules via multiplexers, and a peripheral can be connected to one IO port at a time via the multiplexing feature. This prevents available peripherals on the same IO port from conflicting.

Up to 16 multiplexed function inputs (AF0 to AF15) for the multiplexer on each I/O port can be configured through registers GPIOx_AFRL (for pins 0 to 7) and GPIOx_AFRH (for pins 8 to 15).

- Just after reset, the multiplexer defaults to AF0. The multiplexing function mode of the I/O port is configured through register GPIOx_MODER
- Refer to the datasheet for the distribution of the multiplexing function for each pin.

In addition to this flexible multiplexer architecture, each peripheral has multiplexing functions that can be distributed across different I/O ports to optimize the number of peripherals used in a smaller package.

The user follows the instructions below to configure the IO:

- **Debug Functions:** After each reset, these debug function pins are the reusable function pins that are immediately available to the debugger by default.
- **GPIO:** Configure the corresponding I/O port as output, input or analog mode at GPIOx_MODER
- **Peripheral multiplexing function:**
 - Register GPIOx_AFRL or GPIOx_AFRH configures the corresponding I/O for multiplexing function x (x=0..15)
 - Registers GPIOx_OTYPER, GPIOx_PUPDR and GPIOx_OSPEEDER configure the type, pull-up/down and output speed, respectively.
 - Configure the desired I/O as multiplexed in the GPIOx_MODER registers
 - **Bus read/write function:** GPIO module not only supports Fast IO to read/write GPIOx registers, but also supports to read/write registers through AHB bus, the register access method is selected by GPIO_AHB_SEL bit of SYSCFG module. When GPIO_AHB_SEL bit is 0, the GPIOx registers can only be accessed via Fast IO; when GPIO_AHB_SEL is 1, the GPIOx registers can only be accessed via the AHB bus.
 - The AHB bus access to GPIO registers supports DMA in addition to CPU, i.e., DMA can directly access GPIO registers via the AHB bus.
- **Additional features**
 - The ADC and COMP functions are enabled in the registers of the ADC and COMP modules, regardless of the mode in which the IO port is configured. When the IO port is used as ADC or COMP, the port needs to be configured to analog mode via register GPIOx_MODER
 - For additional crystal functions, configure the respective functions in the corresponding PWR and RCC module registers. These configurations have a higher priority than the standard GPIO configurations.

10.3.3. I/O Control Registers

Each GPIO port has four 32-bit memory-mapped control registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR and GPIOx_PUPDR) to configure up to 16 I/O ports. Register GPIOx_MODER is used to select the I/O mode (input, output, multiplexing, analog). Registers GPIOx_OTYPER and GPIOx_OSPEEDR are used to select the output type (push-pull or open-drain) and speed. Regardless of which I/O direction is used, register GPIOx_PUPDR is used to select pull-up/down.

10.3.4. I/O Data Register

Each GPIO has two 16-bit memory-mapped data registers: input and output data registers (GPIOx_IDR and GPIOx_ODR). Register GPIOx_ODR holds the data to be output and is readable and writable. The input data register (GPIOx_IDR) is used to hold the level status on the I/O port and is read-only.

10.3.5. I/O data processing by bit

The Set/Reset Register (GPIOx_BSRR) is a 32-bit register that individually sets and resets each bit of the Output Data Register (GPIOx_ODR). The number of set/reset register bits is twice the number of output registers (GPIOx_ODR).

Each bit of GPIOx_ODR corresponds to two control bits of GPIOx_BSRR: BS(i) and BR(i). Bit BS(i) set to one sets the corresponding bit of GPIOx_ODR to one, and bit BR(i) set to one clears the corresponding bit of GPIOx_ODR to zero.

Writing 0 to any bit of register GPIOx_BSRR does not affect the corresponding bit of register GPIOx_ODR. If GPIOx_BSRR clears 0 and sets 1 for a bit at the same time, the set 1 operation has priority.

Using register GPIOx_BSRR to change the corresponding bit of register GPIOx_ODR has a one-time effect and does not lock the bit of register GPIOx_ODR. Register GPIOx_ODR can also be accessed directly. Register GPIOx_BSRR simply provides a way to handle atomic bit manipulation.

There is no need for software to disable interrupts during bit manipulation of GPIOx_ODR: one or more bits can be modified in a single atomic AHB write access.

Register GPIOx_LCKR allows freezing of the IO's control registers through a series of special write timings, including GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_AFRL, and GPIOx_AFRH.

A special write/read timing can operate the register GPIOx_LCKR. When Bit16 of this register is written with the correct timing, the LCKR[15:0] write value can lock the I/O (the LCKR[15:0] write value remains unchanged during the write timing). When a lock (LOCK) program is performed on a port bit, the configuration of the port bit cannot be changed again until the next MCU or peripheral reset. Each bit of GPIOx_LCKR freezes the bit corresponding to the control registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_AFRL and GPIOx_AFRH).

The LOCK timing can only access the GPIOx_LCKR register by word (32 bits long) because GPIOx_LCKR bit 16 is set along with the [15:0] bits.

10.3.6. I/O Multiplexing Function Input/Output Mode Configuration

Two registers per I/O can be used to configure the multiplexing function input/output modes. The user multiplexes the multiplexing function to the IO port according to the application requirements.

Many possible peripheral functions can be multiplexed per GPIO port using registers GPIOx_AFRL and GPIOx_AFRH, so the application lets each I/O select one of these functions. The AF select signal is the same for both multiplex function inputs and multiplex function outputs, and separate channels can be selected for multiplex function inputs/outputs for a given I/O.

10.3.7. External interrupt/wakeup line

All ports have external interrupt capability. In order to use an external interrupt line, the port must be disabled from being configured in analog mode or as a crystal pin, and input triggering needs to be enabled.

10.3.8. I/O Input Configuration

When the I/O port is configured as an input:

- Output buffer not enabled
- Schmitt trigger input enable
- Enable/disable pull-up and pull-down resistors according to GPIOx_PUPDR registers.
- Data appearing on the I/O pins is sampled at each AHB clock into the input data registers
- Read accesses to the input data registers yield I/O states

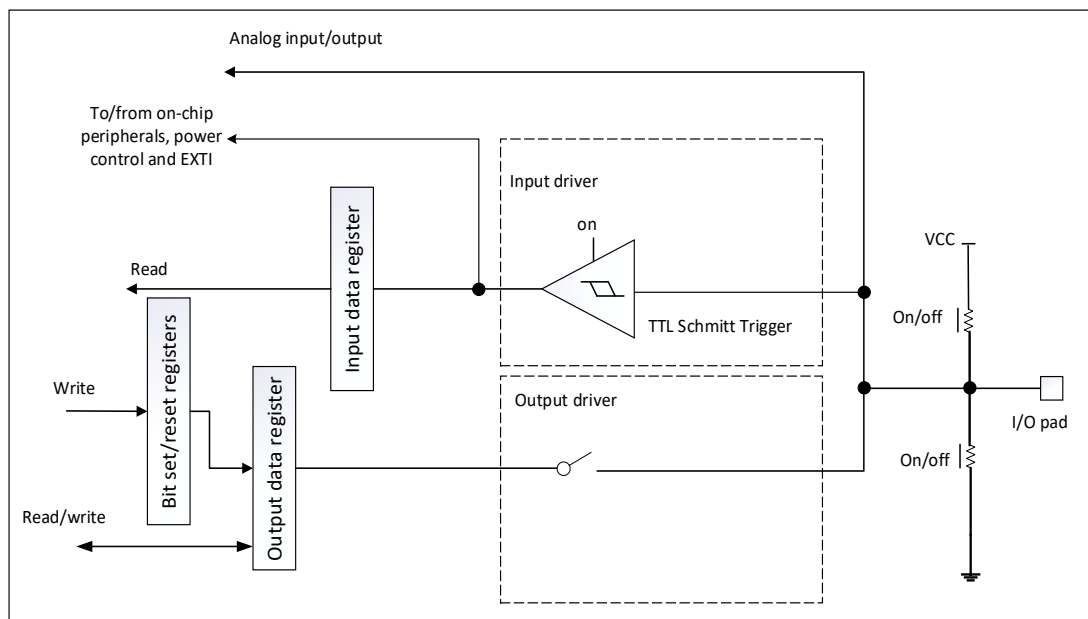


Figure 10-2 Input float/pull-up/pull-down configurations

10.3.9. I/O Output Configuration

When an I/O port is configured as an output:

- Output buffer activated
 - Open-drain mode: '0' on the output register activates the N-MOS, while '1' on the output register places the port in a high-resistance state (the P-MOS is never activated).
 - Push-Pull Mode: '0' on the output register activates N-MOS, while '1' on the output register will activate P-MOS.

- Schmidt trigger input activated
- Enable/disable pull-up and pull-down resistors according to GPIOx_PUPDR registers.
- Data appearing on the I/O pins is sampled at each AHB clock into the input data registers
- Read accesses to the input data registers yield I/O states
- A read access to the output data register gets the value of the subsequent write

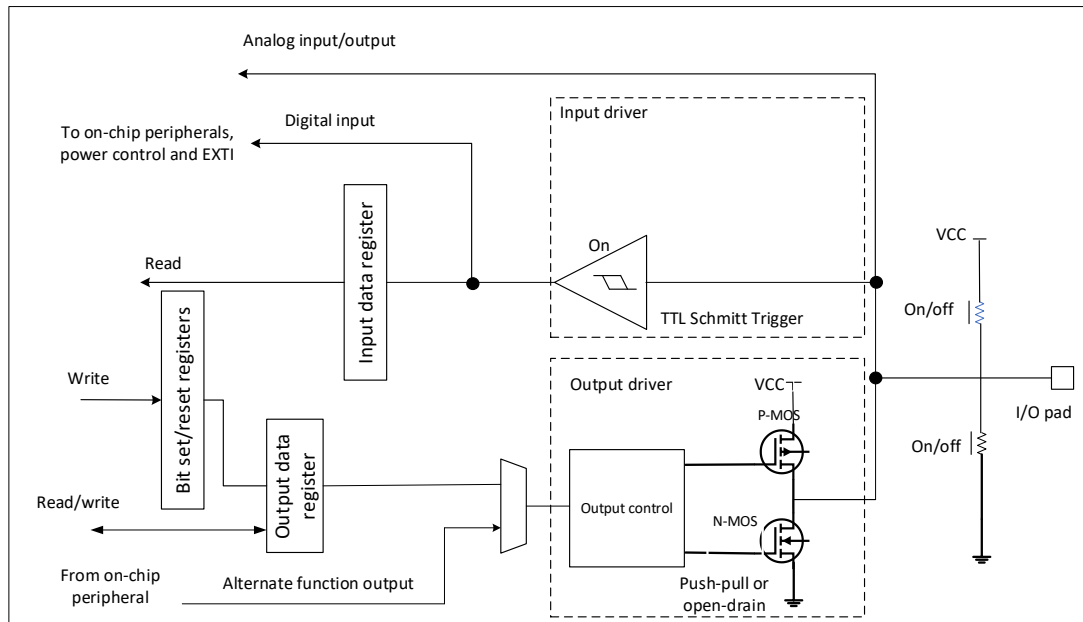


Figure 10-3 Output Configuration

10.3.10. Multiplexing Function Configuration

When an I/O port is configured as a multiplexing function:

- In open-drain or push-pull configurations, the output buffer is turned on
- Signal drive output buffer with built-in peripheral (multiplexed function output)
- Schmidt trigger input activated
- Enable/disable pull-up and pull-down resistors according to GPIOx_PUPDR registers.
- At each AHB clock cycle, data appearing on the I/O pins is sampled into the input data registers
- The I/O port status can be obtained when reading the input data register.

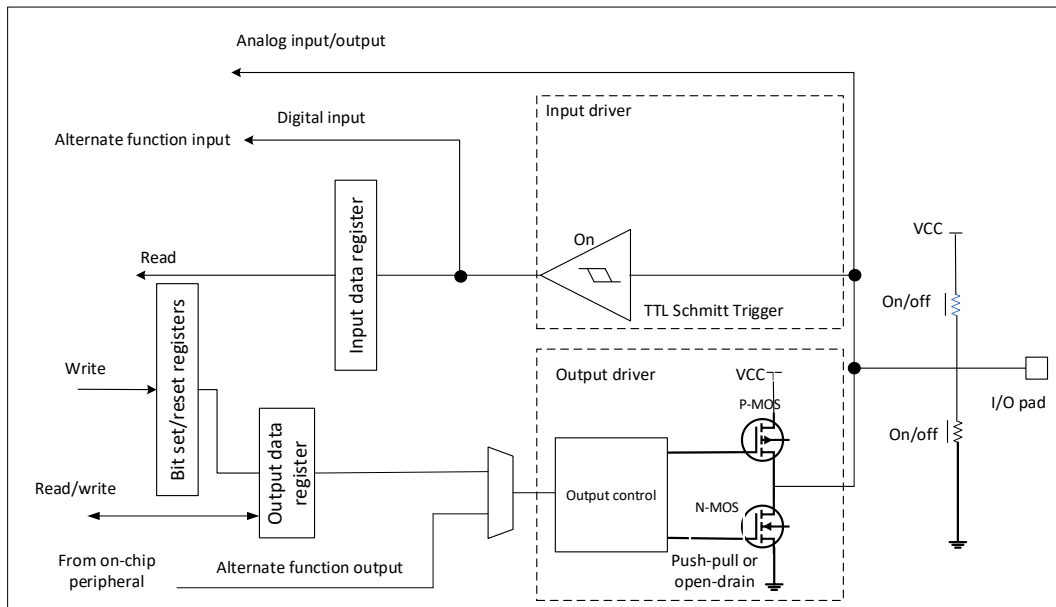


Figure 10-4 Multiplexing Function Configuration

10.3.11. Analog Configuration

When the I/O port is configured in an analog configuration:

- The output buffer is disabled;
- Disabling Schmitt trigger inputs achieves zero consumption on each analog I/O pin. The Schmitt trigger output value is forced to '0';
- Weak pull-up and pull-down resistors are disabled (requires software setting);
- The value is "0" when the input data register is read.

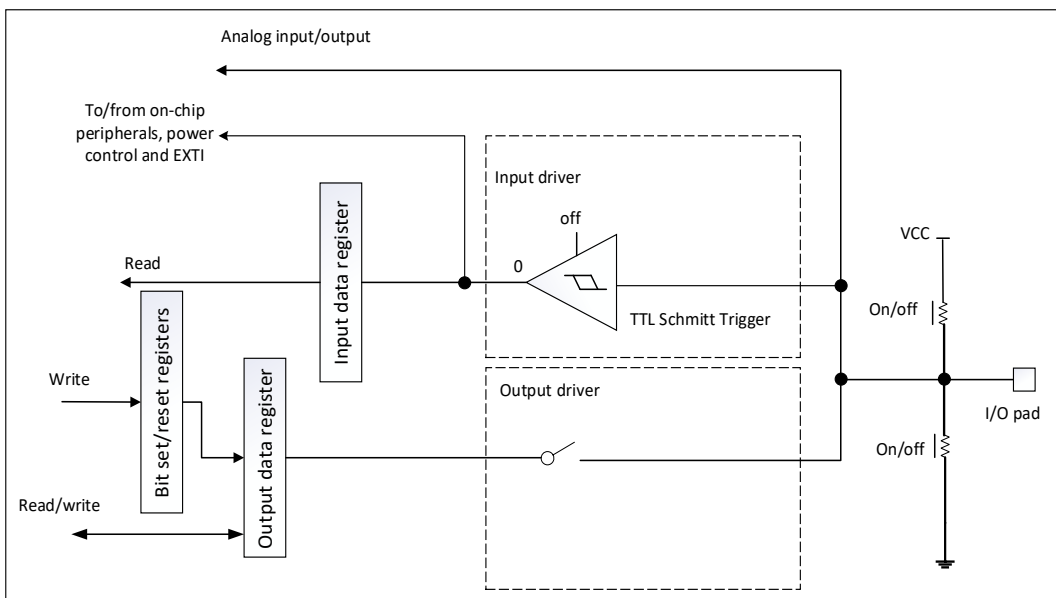


Figure 10-5 High Impedance Analog Configuration

10.3.12. Using the HSE/LSE pins as GPIOs

When the HSE or LSE oscillator is turned off (default after reset), the corresponding pin can be used as a normal GPIO.

When the HSE or LSE oscillator is turned on (HSEON or LSEON in the RCC_CSR register is set), software configuration of the corresponding port as an analog port is required.

When the oscillator is configured for user external clock mode, only OSC_IN or OSC32_IN is reserved for clock input, while the OSC_OUT or OSC32_OUT pin can still be used as a normal GPIO.

10.4. GPIO registers

Word, half-word and byte write operations are available for all GPIO-related registers.

10.4.1. GPIO port mode register (GPIOx_MODER) (x=A, B, C, F)

Address offset: 0x00

Reset value:

- 0xEBFF FFFF (port A)
- 0xFFFF FFFF (port B)
- 0xFFFF FFFF (port C)
- 0x000C FFFF (port F)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODE15[1: 0]		MODE14[1: 0]		MODE13[1: 0]		MODE12[1: 0]		MODE11[1: 0]		MODE10[1: 0]		MODE9[1: 0]		MODE8[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
MODE7[1: 0]		MODE6[1: 0]		MODE5[1: 0]		MODE4[1: 0]		MODE3[1: 0]		MODE2[1: 0]		MODE1[1: 0]		MODE0[1: 0]	
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 0	MODEy [1:0]	RW	0xEBFFFFFF (PA) 0xFFFFFFFF (pb) 0xFFFFFFFF (PC) 0x000CFFFF (PF)	y = 15...0 The software configures the corresponding I/O modes with these bits 00: Input mode 01: Generic Output Mode 10: Multiplexed Functional Modes 11: Simulation mode (reset state)

10.4.2. GPIO port output type register (GPIOx_OTYPER) (x = A, B, C, F)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 16	Res	-	-	Res
15: 0	OT [15:0]	RW	0	Software Configuration of I/O Output Types

				0: Push-pull output (reset state) 1: Open-drain output
--	--	--	--	---

10.4.3. GPIO port output speed register (GPIOx_OSPEEDR) (x = A, B, C, F)

Address offset: 0x08

Reset value: 0x0C00 0000 (Port A)

Reset value: 0x0000 0000 (other ports)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSPEED15		OSPEED14		OSPEED13		OSPEED12		OSPEED11		OSPEED10		OSPEED9		OSPEED8	
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
OSPEED7		OSPEED6		OSPEED5		OSPEED4		OSPEED3		OSPEED2		OSPEED1		OSPEED0	
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 0	OSPEEDy [1:0]	RW	0x0C000000 (PA) 0x0 (other)	Y = 15...0 Software configuration of IO port output speed 00: Low speed 01: Medium speed 10: High Speed 11: Very high

10.4.4. GPIO port pull-down register (GPIOx_PUPDR) (x = A, B, C, F)

Address offset: 0x0C

Reset value:

- 0x2400 0000 (Port A)
- 0x0000 0000 (ports B and C)
- 0x0002 0000 (port F)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPD15[1: 0]		PUPD14[1: 0]		PUPD13[1: 0]		PUPD12[1: 0]		PUPD11[1: 0]		PUPD10[1: 0]		PUPD9[1: 0]		PUPD8[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
PUPD7[1: 0]		PUPD6[1: 0]		PUPD5[1: 0]		PUPD4[1: 0]		PUPD3[1: 0]		PUPD2[1: 0]		PUPD1[1: 0]		PUPD0[1: 0]	
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 0	PUPDy [1:0]	RW	0x24000000 (PA)	Y = 15...0

			0x0 (PB) 0x0 (PC) 0x20000 (PF)	Software Configuration I/O Port Pull-Up or Pull-Down 00: No pull-down 01: Pull-up 10: Dropdown 11: Res
--	--	--	--------------------------------------	--

10.4.5. GPIO port input data register (GPIOx_IDR) (x = A, B, C, F)

Address offset: 0x10

Reset value: 0x0000 XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Bit	Name	R/W	Reset Value	Function
31: 16	Res	-	-	Res
15: 0	IDy	R	-	y = 15...0 This is read-only, and the readout value bit corresponds to the status of the I/O port.

10.4.6. GPIO port output data register (GPIOx_ODR) (x = A, B, C, F)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 16	Res	-	-	Res
15: 0	ODy	RW	0	y = 15...0 The software is readable and writable. Note: For GPIOx_BSRR or GPIOx_BRR (x=A,B,C,F), each ODR bit can be set/cleared independently.

10.4.7. GPIO port bit set/reset register (GPIOx_BSRR) (x = A, B, C, F)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
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
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W

Bit	Name	R/W	Reset Value	Function
31: 16	BRy	W	0	y = 15...0 The software is writable and reads out to return a value of 0 0: no effect on the corresponding ODRy bit 1: Clear the corresponding ODRy bit Note: If the corresponding bits of BSy and BRy are set at the same time, the BSy bit works
15: 0	BSy	W	0	y = 15...0 The software is writable and reads out to return a value of 0 0: no effect on the corresponding ODRy bit 1: Set the corresponding ODRy bit

10.4.8. GPIO Port Configuration Lock Register (GPIOx_LCKR) (x = A, B, C, F)

This register is used to lock the configuration of the port bits when the correct write sequence is executed with 16 bits (LCKK) set. LCKy[15:0] is used to lock the configuration of the GPIO port and cannot be changed during a defined write operation. After a LOCK sequence has been performed on the corresponding port, the configuration of the port bits will not be able to be changed again until the next system reset.

Note: Special write timing is used to write the GPIOx_LCKR register. Only word accesses can be performed in the locking timings.

Each lock bit freezes a specific configuration register (control and multiplexing function registers)

Address offset: 0x1C

Reset value: 0x0000 0000

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

-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LCK1	LCK1	LCK1	LCK1	LCK1	LCK1	LCK	LCK	LCK	LCK	LCK	LCK	LCK	LCK	LCK	LCK
5	4	3	2	1	0	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	Name	R/W	Reset Value	Function
31: 17	Res	-	-	Res
16	LCKK	RW	0	<p>This bit can be read out at any time; it can only be modified by a lock-key write sequence</p> <p>0: Port Configuration Lock keybindings not activated</p> <p>1: Port configuration lock key bit is activated and GPIOx_LCKR register is locked before next system reset</p> <p>Lock key write timing:</p> <p>WR LCKR[16] = '1' + LCKR[15:0]</p> <p>WR LCKR[16] = '0' + LCKR[15:0]</p> <p>WR LCKR[16] = '1' + LCKR[15:0]</p> <p>RD LCKR</p> <p>RD LCKR[16] = '1' (this read operation is optional, but it confirms that the lock is active)</p> <p>Note: The value of LCK[15:0] cannot be changed while operating the write timing of the lock key. Any error in the lock key timing will terminate the lock key being activated. After the first key-lock timing for any bit of the port, reading the LCKK bit is a return of 1, a direct MCU reset or a peripheral reset.</p>
15: 0	LCKy	RW	0	<p>y = 15...0</p> <p>These bits are readable and writable but can only be written when the LCKK bit is zero.</p> <p>0: Configuration without locking the port</p> <p>1: Locked Port Configuration</p>

10.4.9. GPIO multiplexing function register (low) (GPIOx_AFRL) (x = A, B, C, F)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFSEL7[3: 0]				AFSEL6[3: 0]				AFSEL5[3: 0]				AFSEL4[3: 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
AFSEL3[3: 0]				AFSEL2[3: 0]				AFSEL1[3: 0]				AFSEL0[3: 0]			
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 0	AFSELY [3:0] (y= 7 ~ 0)	RW	0	Software can write these bits to configure the multiplexed function I/Os AFSELY choice: 0000: AF01000: AF80001: AF11001: AF9 0010: AF21010: AF10 0011: AF31011: AF11 0100: AF41100: AF12 0101: AF51101: AF13 0110: AF61110: AF14 0111: AF71111: AF15

10.4.10. GPIO multiplexing function register (high) (GPIOx_AFRH) (x = A, B, C, F)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFSEL15[3: 0]				AFSEL14[3: 0]				AFSEL13[3: 0]				AFSEL12[3: 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
AFSEL11[3: 0]				AFSEL10[3: 0]				AFSEL9[3: 0]				AFSEL8[3: 0]			
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 0	AFSELY [3:0] (y= 8 ~ 15)	RW	0	Software can write these bits to configure the multiplexed function I/Os AFSELY choice: 0000: AF01000: AF80001: AF11001: AF9 0010: AF21010: AF10 0011: AF31011: AF11 0100: AF41100: AF12 0101: AF51101: AF13 0110: AF61110: AF14 0111: AF71111: AF15

10.4.11. GPIO Port Bit Reset Register (GPIOx_BRR) (x = A, B, C, F)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W

Bit	Name	R/W	Reset Value	Function
31: 16	Res	-	-	Res
15: 0	BRy	W	0	<p>y = 15...0</p> <p>These bits are software-writable and read out to return a value of 0</p> <p>0: no effect on the corresponding ODRy bit</p> <p>1: Clear the corresponding ODRy bit</p>

11. System Configuration Controller

(SYSCFG)

There is a set of configuration registers within the chip and the main purpose of the system configuration controller:

- I2C type IO filter enable and disable
- Mapping the initial program area according to different boot modes
- DMA peripheral channel selection control
- TIMx cascade control
- PVD Lock Enable and Disable
- Cortex-M0+ LOCKUP Enable and Disable
- Noise filter enable and disable for all GPIOs

11.1. System Configuration Register

11.1.1. SYSCFG Configuration Register 1 (SYSCFG_CFGR1)

The MEM_MODE bit of this register is used to configure the kind of memory address 0x0000 0000 accesses. These two registers are used to select the physical remap for software and to bypass the results of the hardware BOOT selection. After a power-up or PIN reset, the value of these two bits is determined by the actual boot mode configuration sampled by the hardware.

Address offset: 0x00

Reset value: 0x0000 000x (x is the memory mode selected by the actual boot mode configuration)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res							GPIO_AHB_SEL	Res					ETR_SRC_TIM3[2: 0]		
-							RW	-					RW		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	ETR_SRC_TIM2[2: 0]			Res	ETR_SRC_TIM1[2: 0]			TIM3_IC1_S	TIM2_IC4_S		TIM1_IC1_SR		MEM_MOD		
-	RW			-	RW			RC	RC		C		E		
-	RW			-	RW			RW	RW		RW		RW		

Bit	Name	R/W	Reset Value	Function
31: 25	Res	-	-	Res
24	GPIO_AHB_SEL	RW	0	CPU FASTIO or AHB bus access to GPIO register control. 0: FASTIO bus access; 1: AHB bus access;
23: 19	Res	-	-	Res
18: 16	ETR_SRC_TIM3[2: 0]	RW	0	TIMER3 ETR input source selection. 3'b000: ETR originates from GPIO

Bit	Name	R/W	Reset Value	Function
				3'b001: ETR derived from COMP1 3'b010: ETR derived from COMP2 3'b011: ETR derived from ADC 3'b100: ETR derived from COMP3.
15	Res	-	-	Res
14: 12	ETR_SRC_TIM2[2: 0]	RW	0	TIMER2 ETR input source selection. 3'b000: ETR originates from GPIO 3'b001: ETR derived from COMP1 3'b010: ETR derived from COMP2 3'b011: ETR derived from ADC 3'b100: ETR from COMP3
11	Res	-	-	Res
10: 8	ETR_SRC_TIM1[2: 0]	RW	0	TIMER1 ETR input source selection. 3'b000: ETR originates from GPIO 3'b001: ETR derived from COMP1 3'b010: ETR derived from COMP2 3'b011: ETR derived from ADC 3'b100: ETR from COMP3
7: 6	TIM3_IC1_SRC	RW	0	TIM13 CH1 input source 00: from TIM3_CH1 IO 01: from COMP1 10: from COMP2 11: from COMP3
5: 4	TIM2_IC4_SRC	RW	0	TIM2 CH4 input source 00: from TIM2_CH4 IO; 01: from COMP1 10: from COMP2 11: from COMP3
3: 2	TIM1_IC1_SRC	RW	0	TIM1 CH1 input source 00: from TIM1_CH1 IO; 01: from COMP1 10: from COMP2 11: from COMP3
1: 0	MEM_MODE	RW	x	Storage area mapping selection. x0: Main Flash memory mapped at 0x0000 000001: System Flash memory mapped at 0x0000 000011: Embedded SRAM mapped at 0x0000 0000

11.1.2. SYSCFG Configuration Register 2 (SYSCFG_CFGR2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21
Res	Res	Res	Res	Res	Res	Res	Res	COMP3_O cref_CLR_ TIM3	COMP3_O cref_CLR_ TIM2	COMP3_O cref_CLR_ TIM1
-	-	-	-	-	-	-	-	-RW	RW	RW
20	19	18	17	16	15	14	13	12	11	10
COMP2_O cref_CLR_ TIM3	COMP2_O cref_CLR_ TIM2	COMP2_O cref_CLR_ TIM1	COMP1_O cref_CLR_ TIM3	COMP1_O cref_CLR_ TIM2	COMP1_O cref_CLR_ TIM1	COMP3 BRK_TIM1 7	COMP2_ BRK_TIM1 7	COMP1_B RK_TIM17	COMP3 BRK_TIM1 6	COMP2_ BRK_TIM1 6
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
9	8	7	6	5	4	3	2	1	0	
COMP1_B RK_TIM16	COMP3 BRK_TIM1 5	COMP2_ BRK_TIM1 5	COMP1_B RK_TIM15	COMP3 BRK_TIM1	COMP2_ BRK_TIM1	COMP1_B RK_TIM1	PVD_LOC K	Res	LOCKUP_LOCK	
RW	RW	RW	RW	RW	RW	RW	RW	-	RW	

Bit	Name	R/W	Reset Value	Function
31: 24	Res	-	-	Res
23	COMP3_Ocref_CLR_TIM3	RW	0	1: COMP3 output as TIM3 ocref_clr input 0: COMP3 output not used as TIM3 ocref_clr input
22	COMP3_Ocref_CLR_TIM2	RW	0	1: COMP3 output as TIM2 ocref_clr input 0: COMP3 output not used as TIM2 ocref_clr input
21	COMP3_Ocref_CLR_TIM1	RW	0	1: COMP3 output as TIM1 ocref_clr input 0: COMP3 output not used as TIM1 ocref_clr input
20	COMP2_Ocref_CLR_TIM3	RW	0	1: COMP2 output as TIM3 ocref_clr input 0: COMP2 output not used as TIM3 ocref_clr input
19	COMP2_Ocref_CLR_TIM2	RW	0	1: COMP2 outputs as TIM2 ocref_clr inputs 0: COMP2 output not used as TIM2 ocref_clr input
18	COMP2_Ocref_CLR_TIM1	RW	0	1: COMP2 output as TIM1 ocref_clr input

Bit	Name	R/W	Reset Value	Function
				0: COMP2 output not used as TIM1 ocref_clr input
17	COMP1_Ocref_CLR_TIM3	RW	0	1: COMP1 output as TIM3 ocref_clr input 0: COMP1 output not used as TIM3 ocref_clr input
16	COMP1_Ocref_CLR_TIM2	RW	0	1: COMP1 output as TIM2 ocref_clr input 0: COMP1 output not used as TIM2 ocref_clr input
15	COMP1_Ocref_CLR_TIM1	RW	0	1: COMP1 output as TIM1 ocref_clr input 0: COMP1 output not used as TIM1 ocref_clr input
14	COMP3_BRK_TIM17	RW	0	COMP3 as TIM17 break input enable 0: COMP3 output not connected to TIM17 break input 1: COMP3 output as TIM17 break input
13	COMP2_BRK_TIM17	RW	0	COMP2 as TIM17 break input enable 0: COMP2 output not connected to TIM17 break input 1: COMP2 output as TIM17 break input
12	COMP1_BRK_TIM17	RW	0	COMP1 as TIM17 break input enable 0: COMP1 output not connected to TIM17 break input 1: COMP1 output as TIM17 break input
11	COMP3_BRK_TIM16	RW	0	COMP3 is used as the TIM16 break input enable. 0: COMP3 output is not connected to the TIM16 break input; 1: COMP3 output as TIM16 break input;
10	COMP2_BRK_TIM16	RW	0	COMP2 is used as the TIM16 break input enable. 0: COMP2 output is not connected to the TIM16 break input; 1: COMP2 output is used as TIM16 break input;
9	COMP1_BRK_TIM16	RW	0	COMP1 is used as the TIM16 break input enable. 0: COMP1 output is not connected to the TIM16 break input;

Bit	Name	R/W	Reset Value	Function
				1: COMP1 output is used as TIM16 break input;
8	COMP3_BRK_TIM15	RW	0	COMP3 is used as the TIM15 break input enable. 0: COMP3 output is not connected to the TIM15 break input; 1: COMP3 output is used as TIM15 break input;
7	COMP2_BRK_TIM15	RW	0	COMP2 is used as the TIM15 break input enable. 0: The COMP2 output is not connected to the TIM15 break input; 1: COMP2 output is used as TIM15 break input;
6	COMP1_BRK_TIM15	RW	0	COMP1 is used as the TIM15 break input enable. 0: COMP1 output is not connected to the TIM15 break input; 1: COMP1 output is used as TIM15 break input;
5	COMP3_BRK_TIM1	RW	0	COMP3 is used as a TIM1 break input enable. 0: COMP3 output is not connected to TIM1 break input; 1: COMP3 output as TIM1 break input;
4	COMP2_BRK_TIM1	RW	0	COMP2 is used as a TIM1 break input enable. 0: COMP2 output is not connected to the TIM1 break input; 1: COMP2 output is used as TIM1 break input;
3	COMP1_BRK_TIM1	RW	0	COMP1 is used as a TIM1 break input enable. 0: COMP1 output is not connected to the TIM1 break input; 1: COMP1 output is used as TIM1 break input;
2	PVD_LOCK	RW	0	PVD lock enable bit. 0: The PVD interrupt is not connected to the TIM1/15/16/17 break input, and the PVD function is configured by the PVDE and PLS; 1: PVD interrupt is connected to TIM1/15/16/17 break input, PVDE and PLS are read only;

Bit	Name	R/W	Reset Value	Function
1	Res	-	-	Res
0	LOCKUP_LOCK	RW	0	Cortex-M0+ LOCKUP bit lock enable bit. 0: Cortex-M0+ LOCKUP bit not connected to TIM1/15/16/17 break inputs 1: Cortex-M0+ LOCKUP bit connected to TIM1/15/16/17 break input

11.1.3. SYSCFG Configuration Register 3 (SYSCFG_CFGR3)

Address offset: 0x08

Reset value: 0x3F3F 3F3F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res		DMA4_MAP						Res		DMA3_MAP					
-		RW						-		RW					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res		DMA2_MAP						Res		DMA1_MAP					
-		RW						-		RW					

Bit	Name	R/W	Reset Value	Function
31: 30	Res	-	-	Res
29: 24	DMA4_MAP	RW	0x3F	DMA1 channel 4 mapping. See DMA1_MAP description.
23: 22	Res	Res	-	Res
21: 16	DMA3_MAP	RW	0x3F	DMA1 channel 3 mapping. See DMA1_MAP description.
15: 14	Res	Res	-	Res
13: 8	DMA2_MAP	RW	0x3F	DMA1 channel 2 mapping. See DMA1_MAP description.
7: 6	Res	Res	-	Res
5: 0	DMA1_MAP	RW	0x3F	DMA1 Channel 1 Mapping. 000000: ADC1 000001: DAC1 000010: DAC2 000011: SPI1_RD 000100: SPI1_WR 000101: SPI2_RD 000110: SPI2_WR 000111: USART1_RD 001000: USART1_WR

Bit	Name	R/W	Reset Value	Function
				001001: USART2_RD
				001010: USART2_WR
				001011: USART3_RD
				001100: USART3_WR
				001101: USART4_RD
				001110: USART4_WR
				001111: I2C1_RD
				010000: I2C1_WR
				010001: I2C2_RD
				010010: I2C2_WR
				010011: TIM1_CH1
				010100: TIM1_CH2
				010101: TIM1_CH3
				010110: TIM1_CH4
				010111: TIM1_COM
				011000: TIM1_TRG
				011001: TIM1_UP
				011010: TIM2_CH1
				011011: TIM2_CH2
				011100: TIM2_CH3
				011101: TIM2_CH4
				011110: TIM2_UP
				011111: TIM2_TRG
				100000: TIM3_CH1
				100001: TIM3_CH2
				100010: TIM3_CH3
				100011: TIM3_CH4
				100100: TIM3_UP
				100101: TIM3_TRG
				100110: TIM6_UP
				100111: TIM7_UP
				101000: TIM15_CH1
				101001: TIM15_CH2
				101010: TIM15_UP
				101011: TIM15_TRG
				101100: TIM15_COM
				101101: TIM16_CH1
				101110: TIM16_UP
				101111: TIM17_CH1

Bit	Name	R/W	Reset Value	Function
				110000: TIM17_UP 110001: USB 110010: LCD Other: Reserved

11.1.4. SYSCFG Configuration Register 4 (SYSCFG_CFGR4)

Address offset: 0x0C

Reset value: 0x003F 3F3F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res										DMA7_MAP					
-										RW					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res		DMA6_MAP						Res		DMA5_MAP					
-		RW						-		RW					

Bit	Name	R/W	Reset Value	Function
31: 22	Res	-	-	Res
21: 16	DMA7_MAP	RW	0x3F	DMA1 channel 7 mapping. See DMA1_MAP description.
15: 14	Res	-	-	Res
13: 8	DMA6_MAP	RW	0x3F	DMA1 channel 6 mapping. See DMA1_MAP description.
7: 6	Res	-	-	Res
5: 0	DMA5_MAP	RW	0x3F	DMA1 channel 5 mapping. See DMA1_MAP description.

11.1.5. GPIOA filter enable (PA_ENS)

Address offset: 0x10

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 16	Res	-	-	Res
15: 0	PA_ENS[x]	RW	0x0000	PORTA IO Filter Enable. 0: Filtering disabled 1: Filter enable

11.1.6. GPIOB filter enable (PB_ENS)

Address offset: 0x14

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 16	Res	-	-	Res
15: 0	PB_ENS[x]	RW	0x0000	PORTB IO filter enable. 0: Filtering disabled 1: Filter enable

11.1.7. GPIOC filter enable (PC_ENS)

Address offset: 0x18

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 16	Res	-	-	Res
15: 0	PC_ENS[x]	RW	0x0000	PORTC IO filter enable. 0: Filtering disabled 1: Filter enable

11.1.8. GPIOF filter enable (PF_ENS)

Address offset: 0x1C

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 10	Res	-	-	Res
9: 0	PF_ENS[x]	RW	0x000	PORTF IO filter enable. 0: Filtering disabled 1: Filter enable

11.1.9. I2C configuration register (SYSCFG_EIIC)

Address offset: 0x20

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 26	Res	-	-	Res
25: 24	PF_EIIC[1:0]	RW	0	EIIC control signal for PF. Used as I2C type IO Bit0: Control PF3 Bit1: Control PF4
23: 16	Res	-	-	Res
15: 8	PB_EIIC[7:0]	RW	0	EIIC control signal for PB. Used as I2C type IO Bit0: Control PB6 Bit1: Control PB7 Bit2: Control PB8 Bit3: Control PB9

				Bit4: Control PB10 Bit5: Control PB11 Bit6: Control PB13 Bit7: Control PB14
7: 2	Res	-	-	Res
1: 0	PA_EIIC[1:0]	RW	0	EIIC control signal of the PA. Used as I2C type IO Bit0: Control PA9 Bit1: Control PA10

12. Direct Memory Access (DMA)

12.1. Introduction

Direct memory access (DMA) is used to provide high-speed data transfers between peripherals and memory or between memory and memory. Moving data does not require CPU intervention and data can be transferred quickly via DMA, which saves CPU resources for other operations. The DMA controller has seven channels, each dedicated to managing requests for memory access from one or more peripherals. There is also an arbiter to coordinate the priority of individual DMA requests.

12.2. DMA main features

- Single AHB Master
- Supports peripheral to memory, memory to peripheral, memory to memory and peripheral to peripheral data transfers
- On-chip memory devices such as Flash, SRAM, AHB and APB peripherals as source and target
- All DMA channels are independently configurable:
 - Each channel is either associated with a DMA request signal from a peripheral or a software trigger in a memory-to-memory transfer. This configuration is done by software.
 - Priority between requests can be programmed by software (4 levels per channel: very high, high, medium, low), and in equal cases by hardware (e.g. requests for channel 1 have priority over requests for channel 2).
 - Source and destination transfer sizes are independent (byte, halfword, word), simulating packing and unpacking. The source and destination addresses must be aligned by data size.
 - Number of programmable transmission data: 0 ~ 65535
- Each channel generates an interrupt request. Each interrupt request is caused by any of three DMA events: transfer completion, half-transfer, or transfer error.

12.3. DMA Function Description

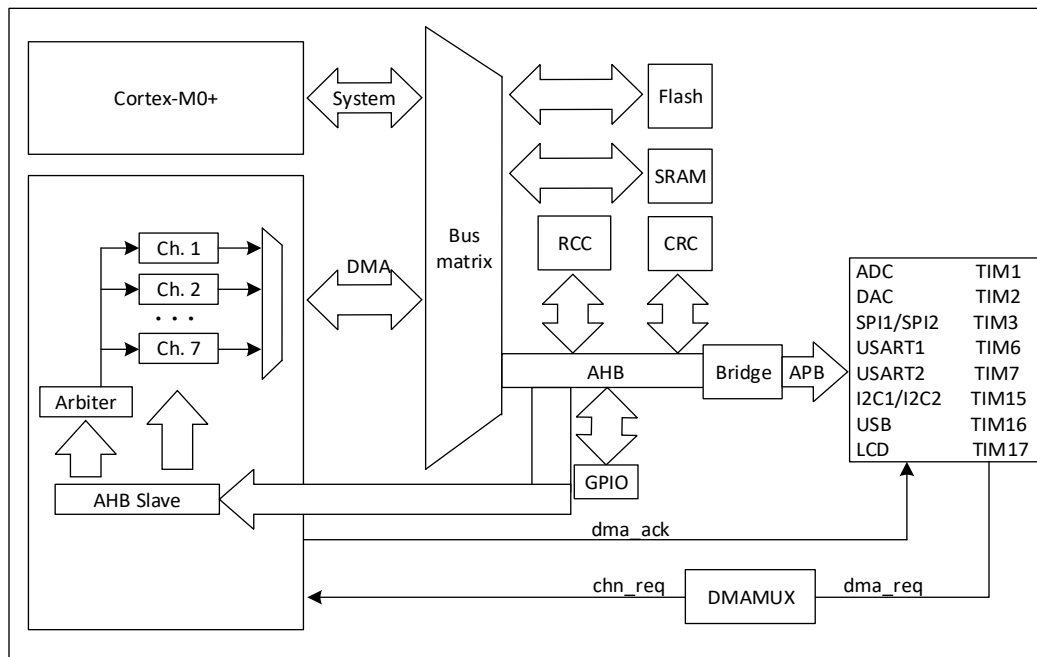


Figure 12-1 DMA Block Diagram

12.3.1. DMA processing

Upon completion of an event, the peripheral sends a request signal to the DMA controller. The DMA controller processes the request based on the priority of the channel. When the DMA controller starts accessing the peripheral that made the request, the DMA controller immediately sends it an answer signal. When an answer signal is received from the DMA controller, the peripheral releases its request immediately. Once the peripheral releases the request, the DMA controller revokes the answer signal. The peripheral can initiate the next transmission if there are more requests.

In summary, each DMA transfer consists of three operations:

- Fetch from the peripheral data register or from the current peripheral/memory address register, the start address on the first transfer is the peripheral base address or memory cell specified by the DMA_CPARx or DMA_CMARx registers.
- Store data to the peripheral register or the memory address indicated by the current peripheral/memory address register, and the start address for the first transfer is the peripheral and address or memory cell specified by the DMA_CPARx or DMA_CMARx registers.
- Performs a decrement operation of the DMA_CNDTRx register, which contains the number of outstanding operations.

12.3.2. Arbiter

The arbiter initiates peripheral/memory accesses based on the priority of the channel request.

Priority management is in 2 stages:

- Software: The priority of each channel can be set in the DMA_CCRx register with 4 levels
 - highest priority

- high priority
- medium priority
- low priority
- Hardware: If 2 requests have the same software priority, the lower numbered channel has higher priority than the higher numbered channel. For example, channel 2 takes precedence over channel 4.

12.3.3. DMA channel

Each channel can perform DMA transfers between peripheral registers with fixed addresses and memory addresses. The amount of data transferred by the DMA is programmable up to a maximum of 65535 bytes, and this register value is decremented after each data transfer.

Transmission data volume programmable

The amount of peripheral and memory transfers can be programmed with the PSIZE and MSIZE bits in the DMA_CCRx register.

Address Pointer Increment

By setting the PINC and MINC flag bits in the DMA_CCRx register, the pointers to the peripherals and memory can be selectively auto-incremented after each transfer.

When set to incremental mode, the next address to be transmitted will be the previous address plus the incremental value, which depends on whether the selected data width is 1, 2, or 4. The address of the first transmission is the address stored in the DMA_CPARx/DMA_CMARx register. During a transfer, these registers maintain their initial values, and software cannot change and read out the address that is currently being transferred (it is in the internal Current Peripheral/Memory Address Register).

When the channel is configured in acyclic mode, no further DMA operations will be generated after the transfer is completed (i.e., the transfer count becomes 0). To start a new DMA transfer, the number of transfers needs to be rewritten in the DMA_CNDTRx register with the DMA channel closed.

In cyclic mode, at the end of the last transfer, the contents of the DMA_CNDTRx register are automatically reloaded to their initial values, and the internal current peripheral/memory address registers are reloaded to the initial base address set by the DMA_CPARx/DMA_CMARx registers.

Channel Configuration Process

Configure the DMA channel on peripheral request as follows:

- Set the address of the peripheral register in the DMA_CPARx register. When a peripheral data transfer request occurs, this address will be the source or destination of the data transfer.
- Set the address of the data memory in the DMA_CMARx register. When a peripheral data transfer request occurs, the transferred data will be read from or written to this address.
- Set the amount of data to be transferred in the DMA_CNDTRx register. This value is decremented after each data transfer.
- Configure the DMA_CCRx register with the following parameters:
 - The priority of the channel.

- data transmission direction
- recurrent mode
- Peripheral and memory incremental modes
- Peripheral and memory data size
- interrupt enable

- Set the EN bit of the DMA_CCRx register to start the channel.

Once a DMA channel is activated, it can respond to DMA requests from peripherals connected to that channel.

When half of the data has been transferred, the Half Transfer Flag (HTIF) is set to 1. When the Half Transfer Interrupt Allowed bit (HTIE) is set, an interrupt request is generated. The transmission completion flag (TCIF) is set to 1 at the end of a data transmission, and an interrupt request is generated when the transmission completion interrupt allowed bit (TCIE) is set.

Channel Status and Disabled Channels

Channel x in the active state is an enabled channel (read DMA_CCRx.EN=1). An active channel x is a channel that must have been enabled by software (DMA_CCRx. EN = 1) and then no transmission error occurs (DMA_ISR.TEIFx = 0). If a transmission error occurs, the channel is automatically disabled by hardware (DMA_CCRx.EN = 0).

The following 3 scenarios may occur:

1. Suspension and resumption of access

This corresponds to the following two actions:

- 1) The active channel is disabled by software (write DMA_CCRx.EN = 0).
- 2) Software re-enabled the channel (DMA_CCRx.EN=1) but did not reconfigure the other channel registers (e.g., DMA_CNDTRx, DMA_CPARx, and DMA_CMARx); or incomplete transfers hung the bus when software disabled it.

This is not supported by the DMA hardware and therefore there is no guarantee that the remaining data transfers will be performed correctly.

2. Stopping and aborting a channel

If the application no longer needs the channel, the active channel can be disabled by software.

The channel is stopped and aborted, but the DMA_CNDTRx register contents may not correctly reflect the remaining data transfer.

3. Abort and restart channel

This corresponds to the software sequence: disable the active channel, then reconfigure the channel and enable it again.

Hardware support when the following conditions are met:

1. The application program guarantees that the DMA has no ongoing (not yet completed) transfers when the channel is disabled by software. For example, an application can first disable a peripheral in DMA mode to ensure that there are no pending hardware DMA requests for that peripheral.

2. Software must perform independent write accesses to the same DMA_CCRx register: first, to close the channel, and second, to reconfigure the channel for the next block transfer if a configuration change is required, including DMA_CCRx. Finally the channel is enabled again.

When a channel transfer error occurs, hardware clears the EN bit of the DMA_CCRx register. This EN bit cannot be set again by software to reactivate channel x until the TEIFx bit in the DMA_ISR register is set.

DMA cyclic mode

Cyclic mode is used to handle circular buffers and continuous data transfers (e.g., ADC's scan mode). The CIRC bit in the DMA_CCRx register is used to enable this feature. When cyclic mode is activated and the number of data transfers becomes 0, the initial value set when the channel was configured is automatically restored and DMA operation continues.

Note: Cyclic mode cannot be used in memory-to-memory mode. Before enabling the channel in cyclic mode (CIRC = 1), software must clear the MEM2MEM bit of the DMA_CCRx register. When cyclic mode is activated, the amount of data to be transferred will be automatically reloaded during the channel configuration phase using the programmed initial values and DMA requests will continue to be responded to.

In order to stop cyclic transfers, software needs to stop the peripheral from generating DMA requests (e.g., exit ADC scan mode) before disabling the DMA channel.

The DMA_CNDTRx value must be explicitly programmed by software before starting/enabling a transfer and after stopping a cyclic transfer.

Memory to memory mode

The operation of the DMA channel can be performed without a peripheral request; this operation is the memory-to-store mode.

After setting the MEM2MEM bit in the DMA_CCRx register, the DMA transfer will begin immediately when the DMA channel is initiated by software setting the EN bit in the DMA_CCRx register. The DMA transfer ends when the DMA_CNDTRx register becomes 0.

Memory-to-memory mode cannot be used in conjunction with cycle mode.

Peripheral to Peripheral Mode

Any DMA channel can be operated in peripheral-to-peripheral mode:

- When a hardware request from a peripheral is selected to trigger a DMA channel

This peripheral is a DMA initiator and transfers data between this peripheral and registers belonging to another memory-mapped peripheral (which is not configured for DMA mode).

- When no peripheral request is selected and connected to a DMA channel

The software configures register-to-register transfers by setting the MEM2MEM bit of the DMA_CCRx register.

Configure the direction of transmission, specifying source/destination

The value of the DIR bit of the DMA_CCRx register sets the direction of the transfer, and therefore identifies the source and target, regardless of the source/target type (peripheral or memory):

- DIR = 1 typically defines memory-to-peripheral transfers. More generally, if DIR = 1:
- The source attribute is defined by the DMA_MARx register, the MSIZE[1:0] field, and the MINC bit of the DMA_CCRx register.

Regardless of their common designation, these "memory" registers, fields, and bits are used to define the source peripheral in peripheral-to-peripheral mode.

- The target attribute is defined by the DMA_PARx register, the PSIZE[1:0] field of the DMA_CCRx register, and the PINC bit.

Regardless of their common designation, these "peripheral" registers, fields, and bits are used to define the target memory in memory-to-memory mode.

- DIR = 0 typically defines a peripheral to memory transfer. More generally, if DIR = 0:
- The source attribute is defined by the DMA_PARx register, the PSIZE[1:0] field of the DMA_CCRx register, and the PINC bit. Regardless of their common designation, these "peripheral" registers, fields, and bits are used to define the source memory in memory-to-memory mode.
- The target attribute is defined by the DMA_MARx register, the MSIZE[1:0] field of the DMA_CCRx register, and the MINC bit.

Regardless of their common designation, these "memory" registers, fields, and bits are used to define the target peripheral in peripheral-to-peripheral mode.

12.3.4. Data Transfer Width/Alignment/Size End

When the memory data width MSIZE and the peripheral data width PSIZE are different, the DMA performs data alignment according to the following table:

Table 12-1 Data width and size end (PINC=MINC=1)

root height	target width	transport amount	Source: Address/Data	transport operation	Target: Address/Data
8	8	4	0x0/B0	1: read B0[7:0] at 0x0, write B0[7:0] at 0x0	0x0/B0
			0x1/B1	2: Read B1[7:0] at 0x1, write B1[7:0] at 0x1	0x1/B1
			0x2/B2	3: read B2[7:0] at 0x2, write B2[7:0] at 0x2	0x2/B2
			0x3/B3	4: read B3[7:0] at 0x3, write B3[7:0] at 0x3	0x3/B3
8	16	4	0x0/B0	1: read B0[7:0] at 0x0, write 00B0[7:0] at 0x0	0x0/00B0
			0x1/B1	2: Read B1[7:0] at 0x1, write 00B1[7:0] at 0x2	0x2/00B1
			0x2/B2	3: read B2[7:0] at 0x2, write 00B2[7:0] at 0x4	0x4/00B2
			0x3/B3	4: Read B3[7:0] at 0x3, write 00B3[7:0] at 0x6	0x6/00B3
8	32	4	0x0/B0	1: read B0[7:0] at 0x0, write 000000B0[31:0] at 0x0	0x0/000000B0
			0x1/B1	2: read B1[7:0] at 0x1, write 000000B1[31:0] at 0x4	0x4/000000B1
			0x2/B2	3: read B2[7:0] at 0x2, write 000000B2[31:0] at 0x8	0x8/000000B2
			0x3/B3	4: Read B3[7:0] at 0x3, write 000000B3[31:0] at 0xC.	0xC/000000B3
16	8	4	0x0/B1B0	1: read B1B0[15:0] at 0x0, write B0[7:0] at 0x0	0x0/B0
			0x2/B3B2	2: Read B3B2[15:0] at 0x2, write B2[7:0] at 0x1	0x1/B2

			0x4/B5B4	3: Read B5B4[15:0] at 0x4, write B4[7:0] at 0x2	0x2/B4
			0x6/B7B6	4: Read B7B6[15:0] at 0x6, write B6[7:0] at 0x3	0x3/B6
16	16	4	0x0/B1B0	1: read B1B0[15:0] at 0x0, write B1B0[15:0] at 0x0	0x0/B1B0
			0x2/B3B2	2: read B3B2[15:0] at 0x2, write B3B2[15:0] at 0x2	0x2/B3B2
			0x4/B5B4	3: Read B5B4[15:0] at 0x4, write B5B4[15:0] at 0x4	0x4/B5B4
			0x6/B7B6	4: Read B7B6[15:0] at 0x6, write B7B6[15:0] at 0x6	0x6/B7B6
16	32	4	0x0/B1B0	1: read B1B0[7:0] at 0x0, write 0000B1B0[31:0] at 0x0	0x0/0000B1B0
			0x2/B3B2	2: Read B3B2[7:0] at 0x2, write 0000B3B2[31:0] at 0x4	0x4/000B3B2
			0x4/B5B4	3: Read B5B4[7:0] at 0x4, write 0000B5B4[31:0] at 0x8	0x8/0000B5B4
			0x6/B7B6	4: read B7B6[7:0] at 0x6, write 0000B7B6[31:0] at 0xC	0xC/0000B7B6
32	8	4	0x0/B3B2B1B0	1: Read B3B2B1B0 [31:0] at 0x0, write B0 [7:0] at 0x0	0x0/B0
			0x4/B7B6B5B4	2: Read B7B6B5B4 [31:0] at 0x4, write B4 [7:0] at 0x1	0x1/B4
			0x8/BBBAB9B8	3: Read BBBAB9B8 [31:0] at 0x8, write B8 [7:0] at 0x2	0x2/B8
			0xC/BFBEBDBC	4: Read BFBEBDBC [31:0] at 0xc, write BC [7:0] at 0x3	0x3/BC
32	16	4	0x0/B3B2B1B0	1: Read B3B2B1B0 [31:0] at 0x0,Write B1B0 [7:0] at 0x0	0x0/B1B0
			0x4/B7B6B5B4	2: Read B7B6B5B4 [31:0] at 0x4, write B5B4 [7:0] at 0x2	0x2/B5B4
			0x8/BBBAB9B8	3: Read BBBAB9B8 [31:0] at 0x8, write B9B8 [7:0] at 0x4	0x4/B9B8
			0xC/BFBEBDBC	4: Read BFBEBDBC [31:0] at 0xc, write BDBC [7:0] at 0x6	0x6/BDBC
32	32	4	0x0/B3B2B1B0	1: Read B3B2B1B0 [31:0] at 0x0,Write B3B2B1B0 [7:0] at 0x0	0x0/B3B2B1B0
			0x4/B7B6B5B4	2: Read B7B6B5B4 [31:0] at 0x4, write B7B6B5B4 [7:0] at 0x2	0x4/B7B6B5B4
			0x8/BBBAB9B8	3: Read BBBAB9B8 [31:0] at 0x8, write BBBAB9B8 [7:0] at 0x4	0x8/BBBAB9B8
			0xC/BFBEBDBC	4: Read BFBEBDBC [31:0] at 0xc, write BFBEBDBC [7:0] at 0x6	0xC/BFBEBDBC

Operate AHB devices that do not support byte or half-word writing

If an error does not occur when the DMA writes in bytes or halfwords to an AHB device that does not support byte or halfword write operations (i.e., HSIZE is not suitable for this module), the DMA will write 32-bit HWDATA data according to the following two examples:

- When HSIZE=Halfword, write the halfword '0xABCD' and the DMA will set the HWDATA bus to '0xABCDABCD'.
- When HSIZE=byte, write byte '0xAB' and the DMA will set the HWDATA bus to '0xABABABABAB'.

Assuming that the AHB/APB bridge is a 32-bit slave device to an AHB that does not handle the HSIZE parameter, it will transfer any byte or half-word on the AHB to the APB in 32-bits in the following manner:

- A write byte data '0xB0' operation on an AHB to address 0x0 (or 0x1, 0x2, or 0x3) will be converted to a write byte data '0xB0B0B0B0B0B0' operation on an APB to address 0x0.
- A write half-word data '0xB1B0' operation on an AHB to address 0x0 (or 0x2) will translate to a write word data '0xB1B0B1B0' operation on an APB to address 0x0.

12.3.5. Error management

Reading or writing a reserved address area will generate a DMA transfer error. When a DMA transfer error occurs during a DMA read or write operation, the hardware automatically clears the EN bit of the channel configuration register (DMA_CCRx) corresponding to the channel on which the error occurred, and that channel operation is stopped. At this time, the Transmission Error Interrupt Flag Bit (TEIF) corresponding to that channel in the DMA_ISR register will be set, and an interrupt will be generated if the Transmission Error Interrupt Allow bit is set in the DMA_CCRx register.

The EN bit of the DMA_CCRx register cannot be set again by software (channel x reactivated) until the TEIFx bit of the DMA_ISR register is cleared (by setting the CTEIFx bit of the DMA_IFCR register).

12.3.6. DMA interrupt

Each DMA channel can generate interrupts on DMA transfer halves, transfer completions, and transfer errors. For application flexibility, these interrupts are turned on by setting different bits in the register.

Table 12-2 DMA Interrupt Requests

isruption event	event marker	Enable Control Bit
more than halfway through transmission	HTIFx	HTIEx
Transmission complete	TCIFx	TCIEx
transmission error	TEIFx	TEIEx
Transmission halfway/transmission complete/transmission error	GIFx	-

Notes:

1. When the DMA_CNDTRx register is 1, the HTIFx bit is not set and the TCIFx bit is set when the transfer is complete.
2. Both the HTIF and TCIF flags are generated when the transmission length NDT is odd (greater than 1). The internal signal TCIF will be generated when $NDT = 1$; HTIF will be generated when $(NDT - (NDT/2 \text{ (rounded)} - 1))$. If $NDT = 5$, TCIF is generated when NDT is reduced to 1; HTIF is generated when NDT is reduced to 4.
3. Both the HTIF and TCIF flags are generated when the transmission length NDT is an even number (greater than 1). The internal signal TCIF will be generated when $NDT = 1$; HTIF will be generated when $(NDT - (NDT/2 \text{ (rounded)} - 1))$. If $NDT = 10$, TCIF is generated when NDT is reduced to 1; HTIF is generated when NDT is reduced to 6.

12.3.7. DMA Peripheral Request Mapping

The mapping of DMA peripheral requests to individual channels of the DMA is controlled by the DMAx_MAP register bits in two SYSCFG registers (SYSCFG_CFGR3 and SYSCFGR_CFGR4), and each peripheral request can be mapped to any of the seven channels by configuration.

Table 12-3 DMA requests per channel

Request MUX input serial number	root	Request MUX input serial number	root	Request MUX input serial number	root
0	ADC1	17	I2C2_RD	34	TIM3_CH3
1	DAC1	18	I2C2_WR	35	TIM3_CH4
2	DAC2	19	TIM1_CH1	36	TIM3_UP
3	SPI1_RD	20	TIM1_CH2	37	TIM3_TRG
4	SPI1_WR	21	TIM1_CH3	38	TIM6_UP
5	SPI2_RD	22	TIM1_CH4	39	TIM7_UP
6	SPI2_WR	23	TIM1_COM	40	TIM15_CH1
7	USART1_RD	24	TIM1_TRG	41	TIM15_CH2
8	USART1_WR	25	TIM1_UP	42	TIM15_UP
9	USART2_RD	26	TIM2_CH1	43	TIM15_TRG
10	USART2_WR	27	TIM2_CH2	44	TIM15_COM
11	USART3_RD	28	TIM2_CH3	45	TIM16_CH1
12	USART3_WR	29	TIM2_CH4	46	TIM16_UP
13	USART4_RD	30	TIM2_UP	47	TIM17_CH1
14	USART4_WR	31	TIM2_TRG	48	TIM17_UP
15	I2C1_RD	32	TIM3_CH1	49	USB
16	I2C1_WR	33	TIM3_CH2	50	LCD

12.4. DMA registers

12.4.1. DMA Interrupt Status Register (DMA_ISR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	TEIF7	HTIF7	TCIF7	GIF7	TEIF6	HTIF6	TCIF6	GIF6	TEIF5	HTIF5	TCIF5	GIF5
-	-	-	-	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
TEIF4	HTIF4	TCIF4	GIF4	TEIF3	HTIF3	TCIF3	GIF3	TEIF2	HTIF2	TCIF2	GIF2	TEIF1	HTIF1	TCIF1	GIF1
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Bit	Name	R/W	Reset Value	Function
31: 28	Res	-	-	Res
27	TEIF7	R	0	Channel 7 transmission error flag. Hardware set, software write DMA_IFCR=1 to clear. 0: No transmission error (TE) 1: Channel 7 transmission error (TE)
26	HTIF7	R	0	Channel 7 half-transmission flag. Hardware set, software write DMA_IFCR=1 to clear. 0: No half-transmission event (HT) 1: Half-transmission event (HT) occurs on channel 7
25	TCIF7	R	0	Channel 7 transmission completion flag. 0: No transmission completion (TC) 1: Channel 7 transmission complete (TC)
24	GIF7	R	0	Channel 7 global interrupt flag. Hardware set, software write DMA_IFCR=1 to clear. 0: no TE/HT/TC events 1: TE/HT/TC event on channel 7
23	TEIF6	R	0	Channel 6 transmission error flag. Hardware set, software write DMA_IFCR=1 to clear. 0: No transmission error (TE) 1: Channel 6 transmission error (TE)
22	HTIF6	R	0	Channel 6 half-transmission flag. Hardware set, software write DMA_IFCR=1 to clear. 0: No half-transmission event (HT) 1: Half-transmission event (HT) occurs on channel 6
21	TCIF6	R	0	Channel 6 transmits the completion flag. 0: No transmission completion (TC) 1: Channel 6 transmission complete (TC)
20	GIF6	R	0	Channel 6 global interrupt flag. Hardware set, software write DMA_IFCR=1 to clear. 0: no TE/HT/TC events 1: TE/HT/TC event occurs on channel 6
19	TEIF5	R	0	Channel 5 transmission error flag. Hardware set, software write DMA_IFCR=1 to clear. 0: No transmission error (TE) 1: Channel 5 transmission error (TE)
18	HTIF5	R	0	Channel 5 half-transmission flag. Hardware set, software write DMA_IFCR=1 to clear. 0: No half-transmission event (HT)

Bit	Name	R/W	Reset Value	Function
				1: Half-transmission event (HT) occurs on channel 5
17	TCIF5	R	0	Channel 5 transmission completion flag. 0: No transmission completion (TC) 1: Channel 5 transmission completion (TC)
16	GIF5	R	0	Channel 5 global interrupt flag. Hardware set, software write DMA_IFCR=1 to clear. 0: no TE/HT/TC events 1: TE/HT/TC event occurs on channel 5
15	TEIF4	R	0	Channel 4 transmission error flag. Hardware set, software write DMA_IFCR=1 to clear. 0: No transmission error (TE) 1: Channel 4 transmission error (TE)
14	HTIF4	R	0	Channel 4 half-transmission flag. Hardware set, software write DMA_IFCR=1 to clear. 0: No half-transmission event (HT) 1: Half-transmission event (HT) occurs on channel 4
13	TCIF4	R	0	Channel 4 transmission completion flag. 0: No transmission completion (TC) 1: Channel 4 transmission completion (TC)
12	GIF4	R	0	Channel 4 global interrupt flag. Hardware set, software write DMA_IFCR=1 to clear. 0: no TE/HT/TC events 1: TE/HT/TC event occurs on channel 4
11	TEIF3	R	0	Channel 3 transmission error flag. Hardware set, software write DMA_IFCR=1 to clear. 0: No transmission error (TE) 1: Channel 3 transmission error (TE)
10	HTIF3	R	0	Channel 3 half-transmission flag. Hardware set, software write DMA_IFCR=1 to clear. 0: No half-transmission event (HT) 1: Half-transmission event (HT) occurs on channel 3
9	TCIF3	R	0	Channel 3 transmission completion flag. 0: No transmission completion (TC) 1: Channel 3 transmission complete (TC)
8	GIF3	R	0	Channel 3 global interrupt flag. Hardware set, software write DMA_IFCR=1 to clear. 0: no TE/HT/TC events 1: TE/HT/TC event occurs on channel 3

Bit	Name	R/W	Reset Value	Function
7	TEIF2	R	0	Channel 2 transmission error flag. Hardware set, software write DMA_IFCR=1 to clear. 0: No transmission error (TE) 1: Channel 2 transmission error (TE)
6	HTIF2	R	0	Channel 2 half-transmission flag. Hardware set, software write DMA_IFCR=1 to clear. 0: No half-transmission event (HT) 1: Half-transmission event (HT) occurs on channel 2
5	TCIF2	R	0	Channel 2 transmission completion flag. Hardware set, software write DMA_IFCR=1 to clear. 0: No transmission completion (TC) 1: Channel 2 transmission complete (TC)
4	GIF2	R	0	Channel 2 global interrupt flag. Hardware set, software write DMA_IFCR=1 to clear. 0: no TE/HT/TC events 1: TE/HT/TC event occurs on channel 2
3	TEIF1	R	0	Channel 1 transmission error flag. Hardware set, software write DMA_IFCR=1 to clear. 0: No transmission error (TE) 1: Channel 1 transmission error (TE)
2	HTIF1	R	0	Channel 1 half-transmission flag. Hardware set, software write DMA_IFCR=1 to clear. 0: No half-transmission event (HT) 1: Half-transmission event (HT) occurs on channel 1
1	TCIF1	R	0	Channel 1 transmission completion flag. Hardware set, software write DMA_IFCR=1 to clear. 0: No transmission completion (TC) 1: Channel 1 transmission complete (TC)
0	GIF1	R	0	Channel 1 global interrupt flag. Hardware set, software write DMA_IFCR=1 to clear. 0: no TE/HT/TC events 1: TE/HT/TC event occurs on channel 1

12.4.2. DMA Interrupt Flag Bit Clear Register (DMA_IFCR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	CTEIF7	CHTIF7	CTCIF7	CGIF7	CTEIF6	CHTIF6	CTCIF6	CGIF6	CTEIF5	CHTIF5	CTCIF5	CGIF5

-	-	-	-	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
CTEIF4	CHTIF4	CTCIF4	CGIF4	CTEIF3	CHTIF3	CTCIF3	CGIF3	CTEIF2	CHTIF2	CTCIF2	CGIF2	CTEIF1	CHTIF1	CTCIF1	CGIF1
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W

Bit	Name	R/W	Reset Value	Function
31: 28	Res	-	-	Res
27	CTEIF7	W	0	Channel 7 transmission error flag cleared. 0: no effect 1: Clear TEIF7
26	CHTIF7	W	0	Channel 7 half-transmission flag cleared. 0: no effect 1: Clear HTIF7
25	CTCIF7	W	0	The channel 7 transmission completion flag is cleared. 0: no effect 1: Clear TCIF7
24	CGIF7	W	0	Channel 7 global interrupt flag cleared. 0: no effect 1: Zero channel 7 GIF/TEIF/HTIF/TCIF
23	CTEIF6	W	0	Channel 6 transmission error flag cleared. 0: no effect 1: Clear TEIF6
22	CHTIF6	W	0	Channel 6 half-transmission flag cleared. 0: no effect 1: Clear HTIF6
21	CTCIF6	W	0	Channel 6 transmission completion flag cleared. 0: no effect 1: Clear TCIF6
20	CGIF6	W	0	Channel 6 global interrupt flag cleared. 0: no effect 1: Zero channel 6 GIF/TEIF/HTIF/TCIF
19	CTEIF5	W	0	Channel 5 transmission error flag cleared. 0: no effect 1: Clear TEIF5
18	CHTIF5	W	0	Channel 5 half-transmission flag cleared. 0: no effect 1: Clear HTIF5
17	CTCIF5	W	0	Channel 5 transmission completion flag cleared. 0: no effect

Bit	Name	R/W	Reset Value	Function
				1: Clear TCIF5
16	CGIF5	W	0	Channel 5 global interrupt flag cleared. 0: no effect 1: Zero channel 5 GIF/TEIF/HTIF/TCIF
15	CTEIF4	W	0	Channel 4 transmission error flag cleared. 0: no effect 1: Clear TEIF4
14	CHTIF4	W	0	Channel 4 half-transmission flag cleared. 0: no effect 1: Clear HTIF4
13	CTCIF4	W	0	Channel 4 transmission completion flag cleared. 0: No effect; 1: Clear TCIF4;
12	CGIF4	W	0	Channel 4 global interrupt flag cleared. 0: no effect 1: Zero channel 4 GIF/TEIF/HTIF/TCIF
11	CTEIF3	W	0	Channel 3 transmission error flag cleared. 0: no effect 1: Clear TEIF3
10	CHTIF3	W	0	Channel 3 half-transmission flag cleared. 0: no effect 1: Clear HTIF3
9	CTCIF3	W	0	Channel 3 transmission completion flag cleared. 0: no effect 1: Clear TCIF3
8	CGIF3	W	0	Channel 3 global interrupt flag cleared. 0: no effect 1: Zero channel 3 GIF/TEIF/HTIF/TCIF
7	CTEIF2	W	0	Channel 2 transmission error flag cleared. 0: no effect 1: Clear TEIF2
6	CHTIF2	W	0	Channel 2 half-transmission flag cleared. 0: no effect 1: Clear HTIF2
5	CTCIF2	W	0	Channel 2 transmission completion flag cleared. 0: no effect 1: Clear TCIF2
4	CGIF2	W	0	Channel 2 global interrupt flag cleared.

Bit	Name	R/W	Reset Value	Function
				0: no effect 1: Zero channel 2 GIF/TEIF/HTIF/TCIF
3	CTEIF1	W	0	Channel 1 transmission error flag cleared. 0: no effect 1: Clear TEIF1
2	CHTIF1	W	0	Channel 1 half-transmission flag cleared. 0: no effect 1: Clear HTIF1
1	CTCIF1	W	0	Channel 1 transmission completion flag cleared. 0: no effect 1: Clear TCIF1
0	CGIF1	W	0	Channel 1 global interrupt flag cleared. 0: no effect 1: Zero channel 1 GIF/TEIF/HTIF/TCIF

12.4.3. DMA Channel 1 Configuration Register (DMA_CCR1)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	MEM2MEM	PL[1:0]		MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
-	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 15	Res	-	-	Res
14	MEM2MEM	RW	0	Channel 1 memory to memory mode. 0: Prohibited 1: Memory to memory mode enable
13: 12	PL[1:0]	RW	0	Channel 1 priority configuration. 00: Low 01: Medium 10: High 11: Very high
11: 10	MSIZE[1:0]	RW	0	Channel 1 memory data width. 00: 8 bits 01: 16-bit

Bit	Name	R/W	Reset Value	Function
				10: 32-bit 11: Res
9: 8	PSIZE[1:0]	RW	0	Channel 1 peripheral data width. 00: 8 bits 01: 16-bit 10: 32-bit 11: Res
7	MINC	RW	0	Channel 1 memory address increment mode. 0: Prohibited 1: Memory address increment mode enable
6	PINC	RW	0	Channel 1 peripheral address incremental mode. 0: Prohibited 1: Peripheral address increment mode enable
5	CIRC	RW	0	Channel 1 cyclic mode. 0: Prohibited 1: Cyclic mode enable
4	DIR	RW	0	Channel 1 data transmission direction. 0: Read from peripheral 1: Read from memory
3	TEIE	RW	0	Channel 1 transmission error interrupt (TE) enable. 0: Prohibited 1: TE interrupt enable
2	HTIE	RW	0	Channel 1 half-transfer interrupt (HT) enable. 0: Prohibited 1: HT interrupt enable
1	TCIE	RW	0	Channel 1 transmission completion interrupt (TC) enable. 0: Prohibited 1: TC interrupt enable
0	EN	RW	0	Channel 1 Enable. 0: Prohibited 1: Channel 1 enable

12.4.4. DMA channel 1 data transfer count register (DMA_CNDTR1)

Address offset: 0x0C

Reset value: 0x0000 0000

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

-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NDT [15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 16	Res	-	-	Res
15: 0	NDT [15:0]	RW	0	<p>Number of channel 1 data transfers.</p> <p>The number of data transfers is 0 to 65535. This register is written only when the channel is not operational (DMA_CCR1.EN=0). This register is read-only when the channel is enabled, indicating the number of bytes remaining to be transferred. This register value is decremented after each DMA transfer.</p> <p>At the end of the data transfer, the contents of the registers either change to 0, or when the channel is configured for cyclic mode, the contents of the registers are automatically reloaded with the values they had when they were previously configured.</p> <p>When the value of this register is 0, no data is transmitted even if the DMA channel starts.</p>

12.4.5. DMA Channel 1 Peripheral Address Register (DMA_CPAR1)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PA [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
PA [15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 0	PA [31:0]	RW	0	<p>Channel 1 Peripheral Address.</p> <p>The base address of the channel 1 peripheral data register, which is used as the source or target of the data transfer.</p>

				<p>When PSIZE=2'b01, the PA[0] bit is not used.</p> <p>The operation is automatically aligned with the half-word address.</p> <p>When PSIZE=2'b10, the PA[1:0] bits are not used. The operation is automatically aligned to the word address.</p>
--	--	--	--	---

12.4.6. DMA channel 1 memory address register (DMA_CMAR1)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MA [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
MA [15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 0	MA [31:0]	RW	0	<p>Channel 1 memory address.</p> <p>Channel 1 memory address as source or destination for data transfer.</p> <p>When MSIZE=2'b01, the MA[0] bit is not used.</p> <p>The operation is automatically aligned with the half-word address.</p> <p>When MSIZE=2'b10, the MA[1:0] bits are not used. The operation is automatically aligned to the word address.</p>

12.4.7. DMA Channel 2 Configuration Register (DMA_CCR2)

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	MEM2MEM	PL[1:0]		MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
-	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 15	Res	-	-	Res
14	MEM2MEM	RW	0	Channel 2 memory to memory mode. 0: Prohibited 1: Memory to memory mode enable
13: 12	PL[1:0]	RW	0	Channel 2 Priority Configuration. 00: Low 01: Medium 10: High 11: Very high
11: 10	MSIZE[1:0]	RW	0	Channel 2 memory data width. 00: 8 bits 01: 16-bit 10: 32-bit 11: Res
9: 8	PSIZE[1:0]	RW	0	Channel 2 peripheral data width. 00: 8 bits 01: 16-bit 10: 32-bit 11: Res
7	MINC	RW	0	Channel 2 memory address increment mode. 0: Prohibited 1: Memory address increment mode enable
6	PINC	RW	0	Channel 2 Peripheral Address Increment Mode. 0: Prohibited 1: Peripheral address increment mode enable
5	CIRC	RW	0	Channel 2 loop mode. 0: Prohibited 1: Cyclic mode enable
4	DIR	RW	0	Channel 2 data transmission direction. 0: Read from peripheral 1: Read from memory
3	TEIE	RW	0	Channel 2 transmission error interrupt (TE) enable. 0: Prohibited 1: TE interrupt enable
2	HTIE	RW	0	Channel 2 half-transfer interrupt (HT) enable. 0: Prohibited 1: HT interrupt enable

Bit	Name	R/W	Reset Value	Function
1	TCIE	RW	0	Channel 2 transmission completion interrupt (TC) enable. 0: Prohibited 1: TC interrupt enable
0	EN	RW	0	Channel 2 Enable. 0: Prohibited 1: Channel 1 enable

12.4.8. DMA Channel 2 Data Transfer Count Register (DMA_CNDTR2)

Address offset: 0x20

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 16	Res	-	-	Res
15: 0	NDT [15:0]	RW	0	<p>Number of channel 2 data transfers.</p> <p>The number of data transfers is 0 to 65535. This register is written only when the channel is not working (DMA_CCR2.EN=0). This register is read-only when the channel is enabled, indicating the number of bytes remaining to be transferred. This register value is decremented after each DMA transfer.</p> <p>At the end of the data transfer, the contents of the registers either change to 0, or when the channel is configured for cyclic mode, the contents of the registers are automatically reloaded with the values they had when they were previously configured.</p> <p>When the value of this register is 0, no data is transmitted even if the DMA channel starts.</p>

12.4.9. DMA Channel 2 Peripheral Address Register (DMA_CPAR2)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PA [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
PA [15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 0	PA [31:0]	RW	0	<p>Channel 2 Peripheral Address.</p> <p>The base address of the channel 2 peripheral data register, which serves as the source or target of the data transfer.</p> <p>When PSIZE=2'b01, the PA[0] bit is not used. The operation is automatically aligned with the half-word address.</p> <p>When PSIZE=2'b10, the PA[1:0] bits are not used. The operation is automatically aligned to the word address.</p>

12.4.10. DMA channel 2 memory address register (DMA_CMAR2)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MA [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
MA [15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 0	MA [31:0]	RW	0	<p>Channel 2 memory address.</p> <p>Channel 2 memory address as source or target for data transfer.</p>

				<p>When MSIZE=2'b01, the MA[0] bit is not used. The operation is automatically aligned with the half-word address.</p> <p>When MSIZE=2'b10, the MA[1:0] bits are not used. The operation is automatically aligned to the word address.</p>
--	--	--	--	--

12.4.11. DMA Channel 3 Configuration Register (DMA_CCR3)

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	MEM2MEM	PL[1:0]		MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
-	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 15	Res	-	-	Res
14	MEM2MEM	RW	0	Channel 3 memory to memory mode. 0: Prohibited 1: Memory to memory mode enable
13: 12	PL[1:0]	RW	0	Channel 3 Priority Configuration. 00: Low 01: Medium 10: High 11: Very high
11: 10	MSIZE[1:0]	RW	0	Channel 3 memory data width. 00: 8 bits 01: 16-bit 10: 32-bit 11: Res
9: 8	PSIZE[1:0]	RW	0	Channel 3 Peripheral Data Width. 00: 8 bits 01: 16-bit 10: 32-bit 11: Res
7	MINC	RW	0	Channel 3 memory address increment mode.

Bit	Name	R/W	Reset Value	Function
				0: Prohibited 1: Memory address increment mode enable
6	PINC	RW	0	Channel 3 Peripheral Address Increment Mode. 0: Prohibited 1: Peripheral address increment mode enable
5	CIRC	RW	0	Channel 3 loop mode. 0: Prohibited 1: Cyclic mode enable
4	DIR	RW	0	Channel 3 data transmission direction. 0: Read from peripheral 1: Read from memory
3	TEIE	RW	0	Channel 3 transmission error interrupt (TE) enable. 0: Prohibited 1: TE interrupt enable
2	HTIE	RW	0	Channel 3 half-transfer interrupt (HT) enable. 0: Prohibited 1: HT interrupt enable
1	TCIE	RW	0	Channel 3 transmission completion interrupt (TC) enable. 0: Prohibited 1: TC interrupt enable
0	EN	RW	0	Channel 3 enable. 0: Prohibited 1: Channel 1 enable

12.4.12. DMA Channel 3 Data Transfer Count Register (DMA_CNDTR3)

Address offset: 0x34

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 16	Res	-	-	Res

15: 0	NDT [15:0]	RW	0	<p>Number of channel 3 data transfers.</p> <p>The number of data transfers is 0 to 65535. This register is written only when the channel is not working (DMA_CCR3.EN=0). This register is read-only when the channel is enabled, indicating the number of bytes remaining to be transferred. This register value is decremented after each DMA transfer.</p> <p>At the end of the data transfer, the contents of the registers either change to 0, or when the channel is configured for cyclic mode, the contents of the registers are automatically reloaded with the values they had when they were previously configured.</p> <p>When the value of this register is 0, no data is transmitted even if the DMA channel starts.</p>
-------	------------	----	---	---

12.4.13. DMA Channel 3 Peripheral Address Register (DMA_CPAR3)

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PA [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
PA [15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 0	PA [31:0]	RW	0	<p>Channel 3 Peripheral Address.</p> <p>The base address of the channel 3 peripheral data register, which is used as the source or target of the data transfer.</p> <p>When PSIZE=2'b01, the PA[0] bit is not used. The operation is automatically aligned with the half-word address.</p> <p>When PSIZE=2'b10, the PA[1:0] bits are not used. The operation is automatically aligned to the word address.</p>

12.4.14. DMA Channel 3 Memory Address Register (DMA_CMAR3)

Address offset: 0x3C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MA [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
MA [15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 0	MA [31:0]	RW	0	<p>Channel 3 memory address.</p> <p>Channel 3 memory address as source or target for data transfer.</p> <p>When MSIZE=2'b01, the MA[0] bit is not used.</p> <p>The operation is automatically aligned with the half-word address.</p> <p>When MSIZE=2'b10, the MA[1:0] bits are not used. The operation is automatically aligned to the word address.</p>

12.4.15. DMA Channel 4 Configuration Register (DMA_CCR4)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	MEM2MEM	PL[1:0]		MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
-	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 15	Res	-	-	Res
14	MEM2MEM	RW	0	<p>Channel memory to memory mode.</p> <p>0: Prohibited</p> <p>1: Memory to memory mode enable</p>
13: 12	PL[1:0]	RW	0	Channel Priority Configuration.

Bit	Name	R/W	Reset Value	Function
				00: Low 01: Medium 10: High 11: Very high
11: 10	MSIZE[1:0]	RW	0	Channel memory data width. 00: 8 bits 01: 16-bit 10: 32-bit 11: Res
9: 8	PSIZE[1:0]	RW	0	Channel peripheral data width. 00: 8 bits 01: 16-bit 10: 32-bit 11: Res
7	MINC	RW	0	Channel memory address increment mode. 0: Prohibited 1: Memory address increment mode enable
6	PINC	RW	0	Channel Peripheral Address Increment Mode. 0: Prohibited 1: Peripheral address increment mode enable
5	CIRC	RW	0	Channel Cycle Mode. 0: Prohibited 1: Cyclic mode enable;
4	DIR	RW	0	Direction of channel data transmission. 0: Read from peripheral 1: Read from memory
3	TEIE	RW	0	Channel transmission error interrupt (TE) enable. 0: Prohibited 1: TE interrupt enable
2	HTIE	RW	0	Channel half-transfer interrupt (HT) enable. 0: Prohibited 1: HT interrupt enable
1	TCIE	RW	0	Channel transmission completion interrupt (TC) enable. 0: Prohibited 1: TC interrupt enable
0	EN	RW	0	channel enable (computing) 0: Prohibited

Bit	Name	R/W	Reset Value	Function
				1: Channel Enable

12.4.16. DMA channel 4 data transfer count register (DMA_CNDTR4)

Address offset: 0x48

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 16	Res	-	-	Res
15: 0	NDT [15:0]	RW	0	<p>Number of channel data transfers.</p> <p>The number of data transfers is 0 to 65535. This register is written only when the channel is not working (DMA_CCR4.EN=0). This register is read-only when the channel is enabled, indicating the number of bytes remaining to be transferred. This register value is decremented after each DMA transfer.</p> <p>At the end of the data transfer, the contents of the registers either change to 0, or when the channel is configured for cyclic mode, the contents of the registers are automatically reloaded with the values they had when they were previously configured.</p> <p>When the value of this register is 0, no data is transmitted even if the DMA channel starts.</p>

12.4.17. DMA Channel 4 Peripheral Address Register (DMA_CPAR4)

Address offset: 0x4C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PA [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
PA [15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 0	PA [31:0]	RW	0	<p>Channel Peripheral Address.</p> <p>The base address of the channel's peripheral data register, which serves as the source or destination of the data transfer.</p> <p>When PSIZE=2'b01, the PA[0] bit is not used. The operation is automatically aligned with the half-word address.</p> <p>When PSIZE=2'b10, the PA[1:0] bits are not used. The operation is automatically aligned to the word address.</p>

12.4.18. DMA Channel 4 Memory Address Register (DMA_CMAR4)

Address offset: 0x50

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MA [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
MA [15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 0	MA [31:0]	RW	0	<p>Channel memory address.</p> <p>Channel memory address as source or destination for data transfer.</p> <p>When MSIZE=2'b01, the MA[0] bit is not used. The operation is automatically aligned with the half-word address.</p> <p>When MSIZE=2'b10, the MA[1:0] bits are not used. The operation is automatically aligned to the word address.</p>

12.4.19. DMA Channel 5 Configuration Register (DMA_CCR5)

Address offset: 0x58

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	MEM2MEM	PL[1:0]		MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN

-	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
---	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

Bit	Name	R/W	Reset Value	Function
31: 15	Res	-	-	Res
14	MEM2MEM	RW	0	Channel memory to memory mode. 0: Prohibited 1: Memory to memory mode enable
13: 12	PL[1:0]	RW	0	Channel Priority Configuration. 00: Low 01: Medium 10: High 11: Very high
11: 10	MSIZE[1:0]	RW	0	Channel memory data width. 00: 8 bits 01: 16-bit 10: 32-bit 11: Res
9: 8	PSIZE[1:0]	RW	0	Channel peripheral data width. 00: 8 bits; 01: 16-bit 10: 32-bit 11: Res
7	MINC	RW	0	Channel memory address increment mode. 0: Prohibited 1: Memory address increment mode enable
6	PINC	RW	0	Channel Peripheral Address Increment Mode. 0: Prohibited 1: Peripheral address increment mode enable
5	CIRC	RW	0	Channel Cycle Mode. 0: Prohibited 1: Cyclic mode enable
4	DIR	RW	0	Direction of channel data transmission. 0: Read from peripheral 1: Read from memory
3	TEIE	RW	0	Channel transmission error interrupt (TE) enable. 0: Prohibited 1: TE interrupt enable
2	HTIE	RW	0	Channel half-transfer interrupt (HT) enable.

Bit	Name	R/W	Reset Value	Function
				0: Prohibited 1: HT interrupt enable
1	TCIE	RW	0	Channel transmission completion interrupt (TC) enable. 0: Prohibited 1: TC interrupt enable
0	EN	RW	0	Channel Enable. 0: Prohibited 1: Channel Enable

12.4.20. DMA channel 5 transfer count register (DMA_CNDTR5)

Address offset: 0x5C

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 16	Res	-	-	Res
15: 0	NDT [15:0]	RW	0	<p>Number of channel data transfers.</p> <p>The number of data transfers is 0 to 65535. This register is written only when the channel is not working (DMA_CCR5.EN=0). This register is read-only when the channel is enabled, indicating the number of bytes remaining to be transferred. This register value is decremented after each DMA transfer.</p> <p>At the end of the data transfer, the contents of the registers either change to 0, or when the channel is configured for cyclic mode, the contents of the registers are automatically reloaded with the values they had when they were previously configured.</p> <p>When the value of this register is 0, no data is transmitted even if the DMA channel starts.</p>

12.4.21. DMA Channel 5 Peripheral Address Register (DMA_CPAR5)

Address offset: 0x60

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PA [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
PA [15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 0	PA [31:0]	RW	0	<p>Channel Peripheral Address.</p> <p>The base address of the channel's peripheral data register, which serves as the source or destination of the data transfer.</p> <p>When PSIZE=2'b01, the PA[0] bit is not used. The operation is automatically aligned with the half-word address.</p> <p>When PSIZE=2'b10, the PA[1:0] bits are not used. The operation is automatically aligned to the word address.</p>

12.4.22. DMA Channel 5 Memory Address Register (DMA_CMAR5)

Address offset: 0x64

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MA [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
MA [15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 0	MA [31:0]	RW	0	<p>Channel memory address.</p> <p>Channel memory address as source or destination for data transfer.</p> <p>When MSIZE=2'b01, the MA[0] bit is not used. The operation is automatically aligned with the half-word address.</p> <p>When MSIZE=2'b10, the MA[1:0] bits are not used. The operation is automatically aligned to the word address.</p>

12.4.23. DMA Channel 6 Configuration Register (DMA_CCR6)

Address offset: 0x6C

Reset value: 0x0000 0000

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

Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	MEM2MEM	PL[1:0]		MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
-	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 15	Res	-	-	Res
14	MEM2MEM	RW	0	Channel memory to memory mode. 0: Prohibited 1: Memory to memory mode enable
13: 12	PL[1:0]	RW	0	Channel Priority Configuration. 00: Low 01: Medium 10: High 11: Very high
11: 10	MSIZE[1:0]	RW	0	Channel memory data width. 00: 8 bits; 01: 16-bit 10: 32-bit 11: Res
9: 8	PSIZE[1:0]	RW	0	Channel peripheral data width. 00: 8 bits; 01: 16-bit 10: 32-bit 11: Res
7	MINC	RW	0	Channel memory address increment mode. 0: Prohibited 1: Memory address increment mode enable
6	PINC	RW	0	Channel Peripheral Address Increment Mode. 0: Prohibited 1: Peripheral address increment mode enable
5	CIRC	RW	0	Channel Cycle Mode. 0: Prohibited 1: Cyclic mode enable
4	DIR	RW	0	Direction of channel data transmission. 0: Read from peripheral 1: Read from memory

Bit	Name	R/W	Reset Value	Function
3	TEIE	RW	0	Channel transmission error interrupt (TE) enable. 0: Prohibited 1: TE interrupt enable
2	HTIE	RW	0	Channel half-transfer interrupt (HT) enable. 0: Prohibited 1: HT interrupt enable
1	TCIE	RW	0	Channel transmission completion interrupt (TC) enable. 0: Prohibited 1: TC interrupt enable
0	EN	RW	0	Channel Enable. 0: Prohibited 1: Channel Enable

12.4.24. DMA Channel 6 Transmit Count Register (DMA_CNDTR6)

Address offset: 0x70

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 16	Res	-	-	Res
15: 0	NDT [15:0]	RW	0	<p>Number of channel data transfers.</p> <p>The number of data transfers is 0 to 65535. This register is written only when the channel is not working (DMA_CCR6.EN=0). This register is read-only when the channel is enabled, indicating the number of bytes remaining to be transferred. This register value is decremented after each DMA transfer.</p> <p>At the end of the data transfer, the contents of the registers either change to 0, or when the channel is configured for cyclic mode, the contents of the registers are automatically reloaded with the values they had when they were previously configured.</p>

Bit	Name	R/W	Reset Value	Function
				When the value of this register is 0, no data is transmitted even if the DMA channel starts.

12.4.25. DMA Channel 6 Peripheral Address Register (DMA_CPAR6)

Address offset: 0x74

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PA [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
PA [15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 0	PA [31:0]	RW	0	<p>Channel Peripheral Address.</p> <p>The base address of the channel's peripheral data register, which serves as the source or destination of the data transfer.</p> <p>When PSIZE=2'b01, the PA[0] bit is not used. The operation is automatically aligned with the half-word address.</p> <p>When PSIZE=2'b10, the PA[1:0] bits are not used. The operation is automatically aligned to the word address.</p>

12.4.26. DMA Channel 6 Memory Address Register (DMA_CMAR6)

Address offset: 0x78

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MA [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
MA [15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 0	MA [31:0]	RW	0	<p>Channel memory address.</p> <p>Channel memory address as source or destination for data transfer.</p> <p>When MSIZE=2'b01, the MA[0] bit is not used. The operation is automatically aligned with the half-word address.</p>

				When MSIZE=2'b10, the MA[1:0] bits are not used. The operation is automatically aligned to the word address.
--	--	--	--	--

12.4.27. DMA Channel 7 Configuration Register (DMA_CCR7)

Address offset: 0x80

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	MEM2MEM	PL[1:0]		MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN
-	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 15	Res	-	-	Res
14	MEM2MEM	RW	0	Channel memory to memory mode. 0: Prohibited 1: Memory to memory mode enable
13: 12	PL[1:0]	RW	0	Channel Priority Configuration. 00: Low 01: Medium 10: High 11: Very high
11: 10	MSIZE[1:0]	RW	0	Channel memory data width. 00: 8 bits 01: 16-bit 10: 32-bit 11: Res
9: 8	PSIZE[1:0]	RW	0	Channel peripheral data width. 00: 8 bits 01: 16-bit 10: 32-bit 11: Res
7	MINC	RW	0	Channel memory address increment mode. 0: Prohibited 1: Memory address increment mode enable
6	PINC	RW	0	Channel Peripheral Address Increment Mode. 0: Prohibited 1: Peripheral address increment mode enable;

Bit	Name	R/W	Reset Value	Function
5	CIRC	RW	0	Channel Cycle Mode. 0: Prohibited 1: Cyclic mode enable
4	DIR	RW	0	Direction of channel data transmission. 0: Read from peripheral 1: Read from memory
3	TEIE	RW	0	Channel transmission error interrupt (TE) enable. 0: Prohibited 1: TE interrupt enable
2	HTIE	RW	0	Channel half-transfer interrupt (HT) enable. 0: Prohibited 1: HT interrupt enable
1	TCIE	RW	0	Channel transmission completion interrupt (TC) enable. 0: Prohibited 1: TC interrupt enable
0	EN	RW	0	Channel Enable. 0: Prohibited 1: Channel Enable

12.4.28. DMA channel 7 transfer count register (DMA_CNDTR7)

Address offset: 0x84

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 16	Res	-	-	Res
15: 0	NDT [15:0]	RW	0	Number of channel data transfers. The number of data transfers is 0 to 65535. This register is written only when the channel is not working (DMA_CCR7.EN=0). This register is read-only when the channel is enabled, indicating

				<p>the number of bytes remaining to be transferred. This register value is decremented after each DMA transfer.</p> <p>At the end of the data transfer, the contents of the registers either change to 0, or when the channel is configured for cyclic mode, the contents of the registers are automatically reloaded with the values they had when they were previously configured.</p> <p>When the value of this register is 0, no data is transmitted even if the DMA channel starts.</p>
--	--	--	--	--

12.4.29. DMA Channel 7 Peripheral Address Register (DMA_CPAR7)

Address offset: 0x88

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PA [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
PA [15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 0	PA [31:0]	RW	0	<p>Channel Peripheral Address.</p> <p>The base address of the channel's peripheral data register, which serves as the source or destination of the data transfer.</p> <p>When PSIZE=2'b01, the PA[0] bit is not used. The operation is automatically aligned with the half-word address.</p> <p>When PSIZE=2'b10, the PA[1:0] bits are not used. The operation is automatically aligned to the word address.</p>

12.4.30. DMA Channel 7 Memory Address Register (DMA_CMAR7)

Address offset: 0x8C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MA [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
MA [15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 0	MA [31:0]	RW	0	<p>Channel memory address.</p> <p>Channel memory address as source or destination for data transfer.</p> <p>When MSIZE=2'b01, the MA[0] bit is not used. The operation is automatically aligned with the half-word address.</p> <p>When MSIZE=2'b10, the MA[1:0] bits are not used. The operation is automatically aligned to the word address.</p>

13. Interrupts and Events

13.1. Nested Vector Interrupt Controller (NVIC)

13.1.1. Main characteristics

- 32 maskable interrupt channels (excluding CPU internal interrupts)
- 4 programmable priorities (2-bit interrupt priority)
- Low latency exception and interrupt handling
- Power Management Control

The NVIC and CPU interfaces are tightly coupled, which enables low-latency interrupt processing and efficient handling of late-arriving interrupts. Including CPU exceptions, all interrupts are managed by the NVIC.

13.1.2. SysTick Calibration Value Registers

The system tick calibration value is set to 9000, which is set to 9MHz (Max fHCLK/8) by the SysTick clock, giving a reference time base of 1ms.

13.1.3. Interrupt and Exception Vector

place ment	prioriti- zation	Priority type	name (of a thing)	clarification	address
-	-	-	-	Res	0x0000_0000
-	-3	set rigidly in place	reset (a dislocated joint, an electronic device etc)	reset (a dislocated joint, an electronic device etc)	0x0000_0004
-	-2	set rigidly in place	NMI_Handler	The RCC Clock Security System (CSS) is linked to the NMI vector	0x0000_0008
-	-1	set rigidly in place	HardFualt_Handler	All types of failures	0x0000_000C
-	3	configurable	SVCall	System service calls via SWI instructions	0x0000_002C
-	5	configurable	PendSV	Suspendable system services	0x0000_0038
	6	configurable	SysTick	System Tick Timer	0x0000_003C
0	7	configurable	WWDG	Window Timer Interrupt	0x0000_0040
1	8	configurable	PVD	Supply voltage detection interrupt (EXTI line 16)	0x0000_0044
2	9	configurable	RTC	RTC interruptions (combined EXTI lines 19)	0x0000_0048
3	10	configurable	Flash	Flash Global Interrupt	0x0000_004C
4	11	configurable	RCC	RCC Global Interrupt	0x0000_0050
5	12	configurable	EXTI0_1	EXTI line[1:0] interrupt	0x0000_0054
6	13	configurable	EXTI2_3	EXTI line[3:2] interrupt	0x0000_0058
7	14	configurable	EXTI4_15	EXTI line[15:4] interrupt	0x0000_005C
8	15	configurable	LCD	LCD Global Interrupt	0x0000_0060
9	16	configurable	DMA_Channel1	DMA Channel 1 Interrupt	0x0000_0064
10	17	configurable	DMA_Channel2_3	DMA Channel 2 & 3 Interrupts	0x0000_0068

place ment	prioriti- zation	Priority type	name (of a thing)	clarification	address
11	18	configurable	DMA_Channel4_5_6_7	DMA Channel 4 & 5 & 6 & 7 Inter- rupts	0x0000_006C
12	19	configurable	ADC_COMP	ADC and COMP interrupts (COMP combined with EXTI 17 & 18 & 20)	0x0000_0070
13	20	configurable	TIM1_BRK_UP_TRG_C OM	TIM1 Disconnect, Update, Trigger, and Communication Interruptions	0x0000_0074
14	21	configurable	TIM1_CC	TIM1 capture/compare interrupt	0x0000_0078
15	22	configurable	TIM2	TIM2 global interrupt	0x0000_007C
16	23	configurable	TIM3	TIM3 Global Interrupt	0x0000_0080
17	24	configurable	TIM6/LPTIM1/DAC	TIM6/LPTIM/DAC global interrupts	0x0000_0084
18	25	configurable	TIM7	TIM7 Global Interrupt	0x0000_0088
19	26	configurable	TIM14	TIM14 Global Interrupt	0x0000_008C
20	27	configurable	TIM15	TIM15 Global Interrupt	0x0000_0090
21	28	configurable	TIM16	TIM16 Global Interrupt	0x0000_0094
22	29	configurable	TIM17	TIM17 Global Interrupt	0x0000_0098
23	30	configurable	I2C1	I2C1 global interrupt	0x0000_009C
24	31	configurable	I2C2	I2C2 Global Interrupt	0x0000_00A0
25	32	configurable	SPI1	SPI1 global interrupt	0x0000_00A4
26	33	configurable	SPI2	SPI2 Global Interrupt	0x0000_00A8
27	34	configurable	USART1	USART1 global interrupt	0x0000_00AC
28	35	configurable	USART2	USART2 Global Interrupt	0x0000_00B0
29	36	configurable	USART3_4	USART3_4 Global Interrupt	0x0000_00B4
30	37	configurable	CAN	CAN global interrupt	0x0000_00B8
31	38	configurable	USB	USB Global Interrupt	0x0000_00BC

13.2. External Interrupt/Event Controller (EXTI)

The Extended Interrupt and Event Controller manages the CPU and system wake-up functions through configurable and direct event inputs (Lines), and outputs the request signals described below:

- Interrupt request, sent to the CPU's IRQ
- Event request, event input sent to the CPU (RXEV)
- Wake-up request to the power management control module

EXTI wakeup requests allow the system to wake up from Stop mode, interrupt requests and event requests can also be used in Run mode.

EXTI allows the management of up to 22 configurable/direct event lines (20 configurable event lines and 2 direct event lines).

13.2.1. EXTI Key Features

- The system can be woken up by GPIO and specified module (PVD/COMP/RTC/LPTIM) input events
- Configurable events (from I/O, or peripherals with stateless Pending bits, peripherals generating pulses)

- Selectable active trigger edge (rising/falling edge)
- interrupt hang flag (computing)
- Independent interrupt and event generation masks
- software-triggered
- Direct-type events (peripherals with associated flags and interrupt Pending status bits)
 - Fixed rising edge trigger
 - There is no interrupt Pending bit in the EXTI module.
 - Independent interrupt and event generation masks
 - No software trigger
- Configurable IO port selection

13.2.2. EXTI Block Diagram

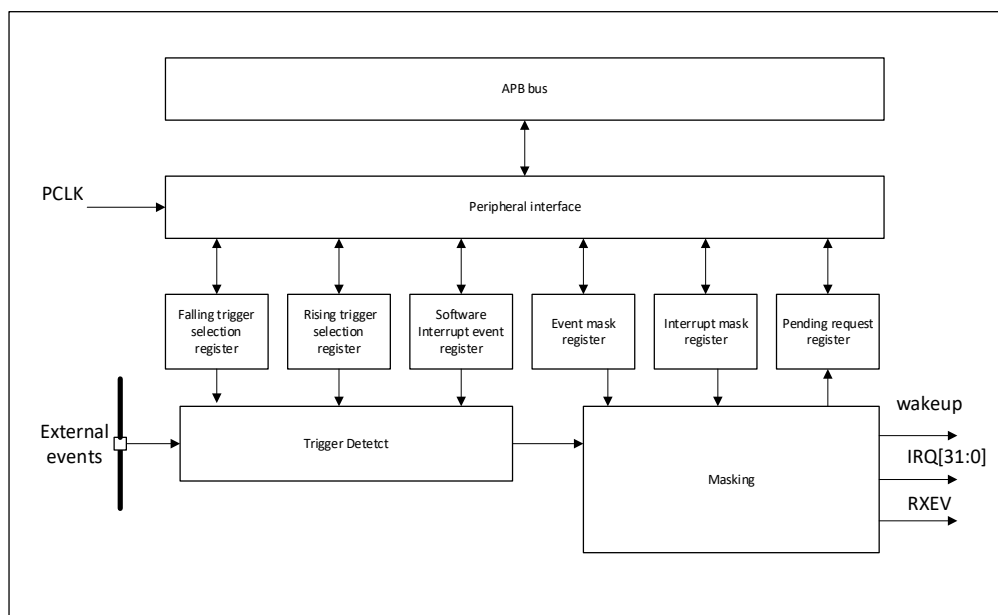


Figure 13-1 EXTI Block Diagram

13.2.3. Interruption management

Wakeup events can be generated in one of two ways:

- interrupt method

Enable interrupts in the peripheral control registers, but not in the NVIC (Nested Vector Interrupt Controller), and enable the SEVONPEND bit in the Cortex-M0 system control registers.

When the MCU recovers from the WFE state, the EXTI peripheral interrupt pending bit and the pending bit of the peripheral NVIC IRQ channel (located in the NVIC interrupt clear pending register) must be cleared.

- event method

Configure the external or internal EXTI line for event mode. When the CPU recovers from the WFE state, there is no need to clear the peripheral interrupt pending bits or the pending bits of the NVIC IRQ channel because the pending bits corresponding to the event lines will not be set.

13.2.4. Functional Description

For an external interrupt line, to generate an interrupt, the interrupt line must be configured and enabled. This is accomplished by programming the two trigger registers to set the desired edge detection, and enabling the interrupt request by writing a '1' to the corresponding bit in the interrupt mask register. An interrupt request is generated when the selected edge appears on the external interrupt line. At the same time, the pending bit corresponding to this interrupt line is set. This request can be cleared by writing a '1' to the corresponding bit in the pending register.

For internal interrupt lines, the active edge is always a rising edge, interrupts are enabled by default in the interrupt mask register, and there is no corresponding pending bit in the pending register.

In order to generate events, the event line needs to be configured and enabled. This is accomplished by programming the two trigger registers to set the desired edge detection and enabling the event request by writing a '1' to the corresponding bit in the event mask register. An event pulse is generated when the selected edge appears on the event line. The pending bit corresponding to this event line will not be set.

For external lines, an interrupt or event request can also be generated by software by writing a '1' in the software interrupt/event register.

NOTE: Interrupts or events related to the internal line can only be triggered when the system is in Stop mode. If the system is still running, no interrupts or events are generated.

13.2.5. Hardware Interrupt Selection

Direct type events generate interrupts in the EXTI module and will generate event signals that wake up the system and CPU subsystem. The CPU clears the interrupt status bit of the peripheral module when processing interrupts generated by this type of trigger event.

To configure a line as an interrupt source, follow these steps:

- Configure the corresponding mask bit in the EXTI_IMR register: enable this interrupt line by setting the corresponding bit in the EXTI_IMR (interrupt mask register).
- Configure the trigger select bits for the interrupt line: set the desired edge detection (rising edge, falling edge, or both) by programming EXTI_RTSR (Rising Edge Trigger Select Register) and EXTI_FTSR (Falling Edge Trigger Select Register).
- Configure Enable and Mask Bits for NVIC Interrupt Channels: Configure the enable and mask bits of the NVIC IRQ channels that control the mapping to EXTI so that interrupts from a particular EXTI line are responded to correctly.

13.2.6. Hardware event selection

To configure a line as an event source, follow these steps:

- Configure the corresponding mask bit in the EXTI_EMR register: enable this event line by setting the corresponding bit in the EXTI_EMR (Event Mask Register).

- Configure the trigger select bit for the event line: set the desired edge detection (rising edge, falling edge, or both) by programming EXTI_RTSR and EXTI_FTSR.

13.2.7. Software interrupt/event selection

Any external line can be configured as a software interrupt/event line. The following are the steps to generate a software interrupt:

- Configure the corresponding mask bits: Configure the corresponding bits in EXTI_IMR (Interrupt Mask Register) or EXTI_EMR (Event Mask Register) as required.
- Setting the corresponding bit in the software interrupt register: Generate a software interrupt by setting the required bit in the EXTI_SWIER (software interrupt/event register).

13.2.8. EXTI Selector

The GPIOs are connected to 16 external interrupt/event lines in the following way:

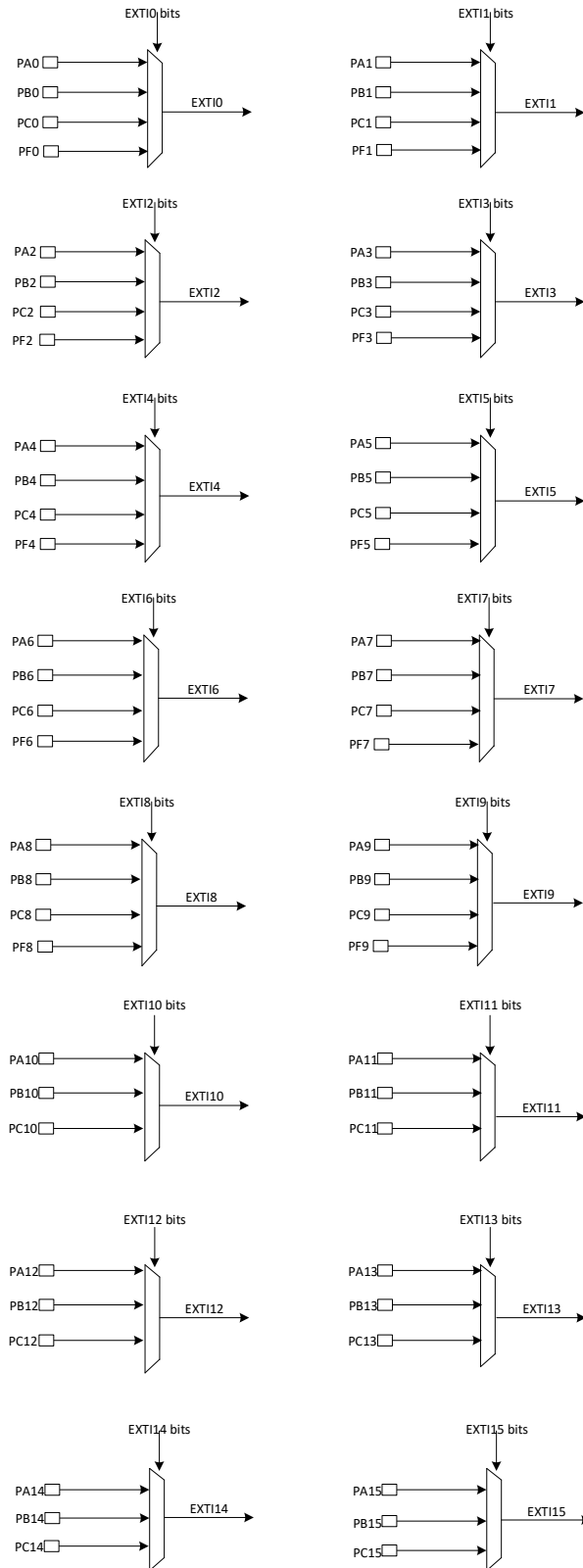


Figure 13-2 External Interrupt/Event GPIO Image

The contents of all LINE connections are shown in the table below:

EXTI line	Line source	Line type
Line 0-15	GPIO	Configurable
Line 16	PVD output	Configurable

EXTI line	Line source	Line type
Line 17	COMP 1 output	Configurable
Line 18	COMP 2 output	Configurable
Line 19	RTC	direct
Line 20	COMP3 output	Configurable
Line 21~28	Res	-
Line 29	LPTIM	direct
Line 30~31	Res	-

13.3. EXTI register

The registers of this peripheral can be accessed in word (32bit), half-word (16bit) and byte (8bit).

13.3.1. Rising edge trigger select register (EXTI_RTSTR)

Address offset: 0x00

Reset value: 0x0000 0000

Contains only register control bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res											RT20	Res	RT18	RT17	RT16
-											RW	-	RW	RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RT15	RT14	RT13	RT12	RT11	RT10	RT9	RT8	RT7	RT6	RT5	RT4	RT3	RT2	RT1	RT0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 21	Res	-	-	Res
20	RT20	RW	0	Configurable type EXTI line20 rising edge triggered configuration. 0: Prohibited 1: Enabling
19	Res	-	-	Res
18	RT18	RW	0	Configurable type EXTI line18 rising edge triggered configuration. 0: Prohibited 1: Enabling
17	RT17	RW	0	Configurable type EXTI line17 rising edge triggered configuration. 0: Prohibited 1: Enabling

Bit	Name	R/W	Reset Value	Function
16	RT16	RW	0	Configurable type EXTI line16 rising edge triggered configuration. 0: Prohibited 1: Enabling
15	RT15	RW	0	Configurable type EXTI line15 rising edge triggered configuration. 0: Prohibited 1: Enabling
14	RT14	RW	0	Configurable type EXTI line14 rising edge triggered configuration. 0: Prohibited 1: Enabling
13	RT13	RW	0	Configurable type EXTI line13 rising edge triggered configuration. 0: Prohibited 1: Enabling
12	RT12	RW	0	Configurable type EXTI line12 rising edge trigger configuration. 0: Prohibited 1: Enabling
11	RT11	RW	0	Configurable type EXTI line11 rising edge triggered configuration. 0: Prohibited 1: Enabling
10	RT10	RW	0	Configurable type EXTI line10 rising edge trigger configuration. 0: Prohibited 1: Enabling
9	RT9	RW	0	Configurable type EXTI line9 rising edge trigger configuration. 0: Prohibited 1: Enabling
8	RT8	RW	0	Configurable type EXTI line8 rising edge triggered configuration. 0: Prohibited 1: Enabling
7	RT7	RW	0	Configurable type EXTI line7 rising edge triggered configuration. 0: Prohibited 1: Enabling
6	RT6	RW	0	Configurable type EXTI line6 rising edge triggered configuration. 0: Prohibited 1: Enabling

Bit	Name	R/W	Reset Value	Function
5	RT5	RW	0	Configurable type EXTI line5 rising edge trigger configuration. 0: Prohibited 1: Enabling
4	RT4	RW	0	Configurable type EXTI line4 rising edge trigger configuration. 0: Prohibited 1: Enabling
3	RT3	RW	0	Configurable type EXTI line3 rising edge trigger configuration. 0: Prohibited 1: Enabling
2	RT2	RW	0	Configurable type EXTI line2 rising edge trigger configuration. 0: Prohibited 1: Enabling
1	RT1	RW	0	Configurable type EXTI line1 rising edge triggered configuration. 0: Prohibited 1: Enabling
0	RT0	RW	0	Configurable type EXTI line0 rising edge triggered configuration. 0: Prohibited 1: Enabling

Configurable lines are edge-triggered and no burrs can be generated on these lines. If a rising edge occurs on the Configurable line during a write to the EXTI_RTSR register, the associated Pending bit is not set.

It is possible to set both rising and falling edges on the same Line, in which case both edges will generate a trigger condition.

13.3.2. Falling edge trigger select register (EXTI_FTSR)

Address offset: 0x04

Reset value: 0x0000 0000

Contains only register control bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res											FT20	Res	FT18	FT17	FT16
-											RW	-	RW	RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FT15	FT14	FT13	FT12	FT11	FT10	FT9	FT8	FT7	FT6	FT5	FT4	FT3	FT2	FT1	FT0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 21	Res	-	-	Res
20	FT20	RW	0	Configurable type EXTI line20 falling edge trigger configuration.

Bit	Name	R/W	Reset Value	Function
				0: Prohibited 1: Enabling
19	Res	-	-	Res
18	FT18	RW	0	Configurable type EXTI line18 falling edge triggered configuration. 0: Prohibited 1: Enabling
17	FT17	RW	0	Configurable type EXTI line17 falling edge triggered configuration. 0: Prohibited 1: Enabling
16	FT16	RW	0	Configurable type EXTI line16 falling edge triggered configuration. 0: Prohibited 1: Enabling
15	FT15	RW	0	Configurable type EXTI line15 falling edge triggered configuration. 0: Prohibited 1: Enabling
14	FT14	RW	0	Configurable type EXTI line14 falling edge triggered configuration. 0: Prohibited 1: Enabling
13	FT13	RW	0	Configurable type EXTI line13 falling edge triggered configuration. 0: Prohibited 1: Enabling
12	FT12	RW	0	Configurable type EXTI line12 falling edge trigger configuration. 0: Prohibited 1: Enabling
11	FT11	RW	0	Configurable type EXTI line11 falling edge trigger configuration. 0: Prohibited 1: Enabling
10	FT10	RW	0	Configurable type EXTI line10 falling edge trigger configuration. 0: Prohibited 1: Enabling
9	FT9	RW	0	Configurable type EXTI line9 falling edge trigger configuration. 0: Prohibited

Bit	Name	R/W	Reset Value	Function
				1: Enabling
8	FT8	RW	0	Configurable type EXTI line8 falling edge trigger configuration. 0: Prohibited 1: Enabling
7	FT7	RW	0	Configurable type EXTI line7 falling edge trigger configuration. 0: Prohibited 1: Enabling
6	FT6	RW	0	Configurable type EXTI line6 falling edge trigger configuration. 0: Prohibited 1: Enabling
5	FT5	RW	0	Configurable type EXTI line5 falling edge trigger configuration. 0: Prohibited 1: Enabling
4	FT4	RW	0	Configurable type EXTI line4 falling edge trigger configuration. 0: Prohibited 1: Enabling
3	FT3	RW	0	Configurable type EXTI line3 falling edge trigger configuration. 0: Prohibited 1: Enabling
2	FT2	RW	0	Configurable type EXTI line2 falling edge trigger configuration. 0: Prohibited 1: Enabling
1	FT1	RW	0	Configurable type EXTI line1 falling edge trigger configuration. 0: Prohibited 1: Enabling
0	FT0	RW	0	Configurable type EXTI line0 falling edge trigger configuration. 0: Prohibited 1: Enabling

Configurable lines are edge-triggered and no burrs can be generated on these lines. If a falling edge occurs on the Configurable line during a write to the EXTI_FTSR register, the associated Pending bit is not set.

It is possible to set both rising and falling edges on the same Line, in which case both edges will generate a trigger condition.

13.3.3. Software Interrupt Event Register (EXTI_SWIER)

Address offset: 0x08

Reset value: 0x0000 0000

Contains only register control bits for configurable events.

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

Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	SW2	Res	SW1	SW1	SW1
-	-	-	-	-	-	-	-	-	-	-	0	-	8	7	6
-	-	-	-	-	-	-	-	-	-	-	RW	-	RW	RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SW1	SW1	SW1	SW1	SW1	SW1	SW	SW	SW	SW	SW	SW4	SW	SW2	SW1	SW0
5	4	3	2	1	0	9	8	7	6	5		3			
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 21	Res	-	-	Res
20	SWI20	RW	0	Configurable type EXTI line20 software rising edge trigger configuration. 0: no effect 1: Generate a rising edge trigger event, which in turn generates an interrupt This bit is cleared by hardware and a read returns 0 (after hardware clear) or the configured value (before hardware clear)
19	Res	-	-	Res
18	SWI18	RW	0	Configurable type EXTI line18 software rising edge triggered configuration. 0: no effect 1: Generate a rising edge trigger event, which in turn generates an interrupt This bit is cleared by hardware and a read returns 0 (after hardware clear) or the configured value (before hardware clear)
17	SWI17	RW	0	Configurable type EXTI line17 software rising edge triggered configuration. 0: no effect 1: Generate a rising edge trigger event, which in turn generates an interrupt This bit is self-cleared by the hardware. Read returns 0.
16	SWI16	RW	0	Configurable type EXTI line16 software rising edge triggered configuration. 0: no effect 1: Generate a rising edge trigger event, which in turn generates an interrupt This bit is cleared by hardware and a read returns 0 (after hardware clear) or the configured value (before hardware clear)

Bit	Name	R/W	Reset Value	Function
15	SWI15	RW	0	Configurable type EXTI line15 software rising edge triggered configuration. 0: no effect 1: Generate a rising edge trigger event, which in turn generates an interrupt This bit is cleared by hardware and a read returns 0 (after hardware clear) or the configured value (before hardware clear)
14	SWI14	RW	0	Configurable type EXTI line14 software rising edge triggered configuration. 0: no effect 1: Generate a rising edge trigger event, which in turn generates an interrupt This bit is self-cleared by the hardware. Read returns 0.
13	SWI13	RW	0	Configurable type EXTI line13 software rising edge triggered configuration. 0: no effect 1: Generate a rising edge trigger event, which in turn generates an interrupt This bit is self-cleared by the hardware. Read returns 0.
12	SWI12	RW	0	Configurable type EXTI line12 software rising edge trigger configuration. 0: no effect 1: Generate a rising edge trigger event, which in turn generates an interrupt This bit is cleared by hardware and a read returns 0 (after hardware clear) or the configured value (before hardware clear)
11	SWI11	RW	0	Configurable type EXTI line11 software rising edge triggered configuration. 0: no effect 1: Generate a rising edge trigger event, which in turn generates an interrupt This bit is cleared by hardware and a read returns 0 (after hardware clear) or the configured value (before hardware clear)
10	SWI10	RW	0	Configurable type EXTI line10 software rising edge triggered configuration. 0: no effect 1: Generate a rising edge trigger event, which in turn generates an interrupt

Bit	Name	R/W	Reset Value	Function
				This bit is cleared by hardware and a read returns 0 (after hardware clear) or the configured value (before hardware clear)
9	SWI9	RW	0	Configurable type EXTI line9 software rising edge triggered configuration. 0: no effect 1: Generate a rising edge trigger event, which in turn generates an interrupt This bit is self-cleared by the hardware. Read returns 0.
8	SWI8	RW	0	Configurable type EXTI line8 software rising edge triggered configuration. 0: no effect 1: Generate a rising edge trigger event, which in turn generates an interrupt This bit is cleared by hardware and a read returns 0 (after hardware clear) or the configured value (before hardware clear)
7	SWI7	RW	0	Configurable type EXTI line7 software rising edge triggered configuration. 0: no effect 1: Generate a rising edge trigger event, which in turn generates an interrupt This bit is cleared by hardware and a read returns 0 (after hardware clear) or the configured value (before hardware clear)
6	SWI6	RW	0	Configurable type EXTI line6 software rising edge triggered configuration. 0: no effect 1: Generate a rising edge trigger event, which in turn generates an interrupt This bit is cleared by hardware and a read returns 0 (after hardware clear) or the configured value (before hardware clear)
5	SWI5	RW	0	Configurable type EXTI line5 software rising edge triggered configuration. 0: no effect 1: Generate a rising edge trigger event, which in turn generates an interrupt This bit is cleared by hardware and a read returns 0 (after hardware clear) or the configured value (before hardware clear)
4	SWI4	RW	0	Configurable type EXTI line4 software rising edge triggered configuration. 0: no effect

Bit	Name	R/W	Reset Value	Function
				1: Generate a rising edge trigger event, which in turn generates an interrupt This bit is cleared by hardware and a read returns 0 (after hardware clear) or the configured value (before hardware clear)
3	SWI3	RW	0	Configurable type EXTI line3 software rising edge triggered configuration. 0: no effect 1: Generate a rising edge trigger event, which in turn generates an interrupt This bit is cleared by hardware and a read returns 0 (after hardware clear) or the configured value (before hardware clear)
2	SWI2	RW	0	Configurable type EXTI line2 software rising edge triggered configuration. 0: no effect 1: Generate a rising edge trigger event, which in turn generates an interrupt This bit is cleared by hardware and a read returns 0 (after hardware clear) or the configured value (before hardware clear)
1	SWI1	RW	0	Configurable type EXTI line1 software rising edge triggered configuration. 0: no effect 1: Generate a rising edge trigger event, which in turn generates an interrupt This bit is cleared by hardware and a read returns 0 (after hardware clear) or the configured value (before hardware clear)
0	SWI0	RW	0	Configurable type EXTI line0 software rising edge triggered configuration. 0: no effect 1: Generate a rising edge trigger event, which in turn generates an interrupt This bit is cleared by hardware and a read returns 0 (after hardware clear) or the configured value (before hardware clear)

13.3.4. Pending Register (EXTI_PR)

Address offset: 0x0C

Reset value: 0x0000 0000

Contains only register control bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	PR20	Res	PR18	PR17	PR16

-											RC_W1	-	RC_W1		
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
RC_W1															

Bit	Name	R/W	Reset Value	Function
31: 21	Res	-	-	-
20	PR20	RC_W1	0	Configurable type EXTI line20 event hang flag. This position bit when software or hardware generates a rising/falling edge trigger event. Software write 1 to clear. 0: No event request was generated; 1: Generate a request for a rising edge/falling edge/software-triggered event;
19	Res	-	-	-
18	PR18	RC_W1	0	Configurable type EXTI line18 event hang flag. This position bit when software or hardware generates a rising/falling edge trigger event. Software write 1 to clear. 0: No event request was generated; 1: Generate a request for a rising edge/falling edge/software-triggered event;
17	PR17	RC_W1	0	Configurable type EXTI line17 event hang flag. This position bit when software or hardware generates a rising/falling edge trigger event. Software write 1 to clear. 0: No event request was generated; 1: Generate a request for a rising edge/falling edge/software-triggered event;
16	PR16	RC_W1	0	Configurable type EXTI line16 event hang flag. This position bit when software or hardware generates a rising/falling edge trigger event. Software write 1 to clear. 0: No event request was generated; 1: Generate a request for a rising edge/falling edge/software-triggered event;

Bit	Name	R/W	Reset Value	Function
15	PR15	RC_W1	0	Configurable type EXTI line15 event hang flag. This position bit when software or hardware generates a rising/falling edge trigger event. Software write 1 to clear. 0: No event request was generated; 1: Generate a request for a rising edge/falling edge/software-triggered event;
14	PR14	RC_W1	0	Configurable type EXTI line14 event hang flag. This position bit when software or hardware generates a rising/falling edge trigger event. Software write 1 to clear. 0: No event request was generated; 1: Generate a request for a rising edge/falling edge/software-triggered event;
13	PR13	RC_W1	0	Configurable type EXTI line13 event hang flag. This position bit when software or hardware generates a rising/falling edge trigger event. Software write 1 to clear. 0: No event request was generated; 1: Generate a request for a rising edge/falling edge/software-triggered event;
12	PR12	RC_W1	0	Configurable type EXTI line12 event hang flag. This position bit when software or hardware generates a rising/falling edge trigger event. Software write 1 to clear. 0: No event request was generated; 1: Generate a request for a rising edge/falling edge/software-triggered event;
11	PR11	RC_W1	0	Configurable type EXTI line11 event hang flag. This position bit when software or hardware generates a rising/falling edge trigger event. Software write 1 to clear. 0: No event request was generated; 1: Generate a request for a rising edge/falling edge/software-triggered event;
10	PR10	RC_W1	0	Configurable type EXTI line10 event hang flag. This position bit when software or hardware generates a rising/falling edge trigger event. Software write 1 to clear.

Bit	Name	R/W	Reset Value	Function
				0: No event request was generated; 1: Generate a request for a rising edge/falling edge/software-triggered event;
9	PR9	RC_W1	0	Configurable type EXTI line9 event hang flag. This position bit when software or hardware generates a rising/falling edge trigger event. Software write 1 to clear. 0: No event request was generated; 1: Generate a request for a rising edge/falling edge/software-triggered event;
8	PR8	RC_W1	0	Configurable type EXTI line8 event hang flag. This position bit when software or hardware generates a rising/falling edge trigger event. Software write 1 to clear. 0: No event request was generated; 1: Generate a request for a rising edge/falling edge/software-triggered event;
7	PR7	RC_W1	0	Configurable type EXTI line7 event hang flag. This position bit when software or hardware generates a rising/falling edge trigger event. Software write 1 to clear. 0: No event request was generated; 1: Generate a request for a rising edge/falling edge/software-triggered event;
6	PR6	RC_W1	0	Configurable type EXTI line6 event hang flag. This position bit when software or hardware generates a rising/falling edge trigger event. Software write 1 to clear. 0: No event request was generated; 1: Generate a request for a rising edge/falling edge/software-triggered event;
5	PR5	RC_W1	0	Configurable type EXTI line5 event hang flag. This position bit when software or hardware generates a rising/falling edge trigger event. Software write 1 to clear. 0: No event request was generated; 1: Generate a request for a rising edge/falling edge/software-triggered event;

Bit	Name	R/W	Reset Value	Function
4	PR4	RC_W1	0	Configurable type EXTI line4 event hang flag. This position bit when software or hardware generates a rising/falling edge trigger event. Software write 1 to clear. 0: No event request was generated; 1: Generate a request for a rising edge/falling edge/software-triggered event;
3	PR3	RC_W1	0	Configurable type EXTI line3 event hang flag. This position bit when software or hardware generates a rising/falling edge trigger event. Software write 1 to clear. 0: No event request was generated; 1: Generate a request for a rising edge/falling edge/software-triggered event;
2	PR2	RC_W1	0	Configurable type EXTI line2 event hang flag. This position bit when software or hardware generates a rising/falling edge trigger event. Software write 1 to clear. 0: No event request was generated; 1: Generate a request for a rising edge/falling edge/software-triggered event;
1	PR1	RC_W1	0	Configurable type EXTI line1 event hang flag. This position bit when software or hardware generates a rising/falling edge trigger event. Software write 1 to clear. 0: No event request was generated; 1: Generate a request for a rising edge/falling edge/software-triggered event;
0	PR0	RC_W1	0	Configurable type EXTI line0 event hang flag. This position bit when software or hardware generates a rising/falling edge trigger event. Software write 1 to clear. 0: No event request was generated; 1: Generate a request for a rising edge/falling edge/software-triggered event;

13.3.5. External Interrupt Select Register 1 (EXTI_EXTICR1)

Address offset: 0x60

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	EXTI3[1: 0]		Res	Res	Res	Res	Res	Res	EXTI2[1: 0]	
-	-	-	-	-	-	RW	RW	-	-	-	-	-	-	RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	EXTI1[1: 0]		Res	Res	Res	Res	Res	Res	EXTI0[1: 0]	
-	-	-	-	-	-	RW	RW	-	-	-	-	-	-	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 26	Res	-	-	Res
25: 24	EXTI3[1: 0]	RW	0	EXTI3 corresponds to GPIO port selection. 2'b00: PA[3] pin 2'b01: PB[3] pin 2'b10: PC[3] pin 2'b11: PF[3] pin
23: 18	Res	-	-	Res
17: 16	EXTI2[1: 0]	RW	0	EXTI2 corresponds to GPIO port selection. 2'b00: PA[2] pin 2'b01: PB[2] pin 2'b10: PC[2] pin 2'b11: PF[2] pin
15: 10	Res	-	-	Res
9: 8	EXTI1[1: 0]	RW	0	EXTI1 corresponds to GPIO port selection. 2'b00: PA[1] pin 2'b01: PB[1] pin 2'b10: PC[1] pin 2'b11: PF[1] pin
7: 2	Res	-	-	Res
1: 0	EXTI0[1: 0]	RW	0	EXTI0 corresponds to GPIO port selection. 2'b00: PA[0] pin 2'b01: PB[0] pin 2'b10: PC[0] pin 2'b11: PF[0] pin

13.3.6. External Interrupt Select Register 2 (EXTI_EXTICR2)

Address offset: 0x64

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	EXTI7[1: 0]		Res	Res	Res	Res	Res	Res	EXTI6[1: 0]	
-	-	-	-	-	-	RW	RW	-	-	-	-	-	-	RW	RW

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	EXTI5[1: 0]		Res	Res	Res	Res	Res	Res	EXTI4[1: 0]	
-	-	-	-	-	-	RW	RW	-	-	-	-	-	-	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 26	Res	-	-	Res
25: 24	EXTI7[1: 0]	RW	0	EXTI7 corresponds to GPIO port selection. 2'b00: PA[7] pin 2'b01: PB[7] pin 2'b10: PC[7] pin 2'b11: PF[7] pin
23: 18	Res	-	-	Res
17: 16	EXTI6[1: 0]	RW	0	EXTI6 corresponds to GPIO port selection. 2'b00: PA[6] pin 2'b01: PB[6] pin 2'b10: PC[6] pin 2'b11: PF[6] pin
15: 10	Res	-	-	Res
9: 8	EXTI5[1: 0]	RW	0	EXTI5 corresponds to GPIO port selection. 2'b00: PA[5] pin 2'b01: PB[5] pin 2'b10: PC[5] pin 2'b11: PF[5] pin
7: 2	Res	-	-	Res
1: 0	EXTI4[1: 0]	RW	0	EXTI4 corresponds to GPIO port selection. 2'b00: PA[4] pin 2'b01: PB[4] pin 2'b10: PC[4] pin 2'b11: PF[4] pin

13.3.7. External Interrupt Select Register 3 (EXTI_EXTICR3)

Address offset: 0x68

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	EXTI11[1: 0]		Res	Res	Res	Res	Res	Res	EXTI10[1: 0]	
-	-	-	-	-	-	RW	RW	-	-	-	-	-	-	RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	EXTI9[1: 0]		Res	Res	Res	Res	Res	Res	EXTI8[1: 0]	
-	-	-	-	-	-	RW	RW	-	-	-	-	-	-	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 26	Res	-	-	Res
25: 24	EXTI11[1: 0]	RW	0	EXTI11 corresponds to GPIO port selection. 2'b00: PA[11] pin 2'b01: PB[11] pin 2'b10: PC[11] pin 2'b11: Reserved
23: 18	Res	-	-	Res
17: 16	EXTI10[1: 0]	RW	0	EXTI11 corresponds to GPIO port selection. 2'b00: PA[10] pin 2'b01: PB[10] pin 2'b10: PC[10] pin 2'b11: Reserved
15: 10	Res	-	-	Res
9: 8	EXTI9[1: 0]	RW	0	EXTI11 corresponds to GPIO port selection. 2'b00: PA[9] pin 2'b01: PB[9] pin 2'b10: PC[9] pin 2'b11: PF[9] pin
7: 2	Res	-	-	Res
1: 0	EXTI8[1: 0]	RW	0	EXTI8 corresponds to GPIO port selection. 2'b00: PA[8] pin 2'b01: PB[8] pin 2'b10: PC[8] pin 2'b11: PF[8] pin

13.3.8. External Interrupt Select Register 4 (EXTI_EXTICR4)

Address offset: 0x6C

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	EXTI15[1: 0]		Res	Res	Res	Res	Res	Res	EXTI14[1: 0]	
-	-	-	-	-	-	RW	RW	-	-	-	-	-	-	RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	EXTI13[1: 0]		Res	Res	Res	Res	Res	Res	EXTI12[1: 0]	
-	-	-	-	-	-	RW	RW	-	-	-	-	-	-	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 26	Res	-	-	Res

Bit	Name	R/W	Reset Value	Function
25: 24	EXTI15[1: 0]	RW	0	EXTI15 Corresponds to GPIO port selection. 2'b00: PA[15] pin 2'b01: PB[15] pin 2'b10: PC[15] pin 2'b11: Reserved
23: 18	Res	-	-	Res
17: 16	EXTI14[1: 0]	RW	0	EXTI14 Corresponds to GPIO port selection. 2'b00: PA[14] pin 2'b01: PB[14] pin 2'b10: PC[14] pin 2'b11: Reserved
15: 10	Res	-	-	Res
9: 8	EXTI13[1: 0]	RW	0	EXTI13 Corresponds to GPIO port selection. 2'b00: PA[13] pin 2'b01: PB[13] pin 2'b10: PC[13] pin 2'b11: Reserved
7: 2	Res	-	-	Res
1: 0	EXTI12[1: 0]	RW	0	EXTI12 corresponds to GPIO port selection. 2'b00: PA[12] pin 2'b01: PB[12] pin 2'b10: PC[12] pin 2'b11: Reserved

13.3.9. Interrupt Mask Register (EXTI_IMR)

Address offset: 0x80

Reset value: 0x2008 0000

Note: The interrupt mask bit for a Direct line defaults to 1, which allows the line; the mask bit for a configurable line defaults to 0, which masks the line.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	IM29	Res	Res	Res	Res	Res	Res	Res	Res	IM20	IM19	IM18	IM17	IM16
		RW									RW	RW	RW	RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IM15	IM14	IM13	IM12	IM11	IM10	IM9	IM8	IM7	IM6	IM5	IM4	IM3	IM2	IM1	IM0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 30	Res	-	-	Res

Bit	Name	R/W	Reset Value	Function
29	IM29	RW	1	EXTI line29 is used as interrupt wake-up CPU mask control. 0: Interrupt wakeup mask 1: Interrupt wakeup not masked
28: 21	Res	-	-	Res
20	IM20	RW	0	EXTI line20 is used as interrupt wake-up CPU mask control. 0: Interrupt wakeup mask 1: Interrupt wakeup not masked
19	IM19	RW	1	EXTI line19 is used as interrupt wake-up CPU mask control. 0: Interrupt wakeup mask 1: Interrupt wakeup not masked
18	IM18	RW	0	EXTI line18 is used as interrupt wake-up CPU mask control. 0: Interrupt wakeup mask 1: Interrupt wakeup not masked
17	IM17	RW	0	EXTI line17 is used as interrupt wake-up CPU mask control. 0: Interrupt wakeup mask 1: Interrupt wakeup not masked
16	IM16	RW	0	EXTI line16 is used as interrupt wake-up CPU mask control. 0: Interrupt wakeup mask 1: Interrupt wakeup not masked
15	IM15	RW	0	EXTI line15 is used as interrupt wake-up CPU mask control. 0: Interrupt wakeup mask 1: Interrupt wakeup not masked
14	IM14	RW	0	EXTI line14 is used as interrupt wake-up CPU mask control. 0: Interrupt wakeup mask 1: Interrupt wakeup not masked
13	IM13	RW	0	EXTI line13 is used as interrupt wake-up CPU mask control. 0: Interrupt wakeup mask 1: Interrupt wakeup not masked

Bit	Name	R/W	Reset Value	Function
12	IM12	RW	0	EXTI line12 is used as interrupt wake-up CPU mask control. 0: Interrupt wakeup mask 1: Interrupt wakeup not masked
11	IM11	RW	0	EXTI line11 is used as interrupt wake-up CPU mask control. 0: Interrupt wakeup mask 1: Interrupt wakeup not masked
10	IM10	RW	0	EXTI line10 is used as interrupt wake-up CPU mask control. 0: Interrupt wakeup mask 1: Interrupt wakeup not masked
9	IM9	RW	0	EXTI line9 is used as interrupt wake-up CPU mask control. 0: Interrupt wakeup mask 1: Interrupt wakeup not masked
8	IM8	RW	0	EXTI line8 is used as interrupt wake-up CPU mask control. 0: Interrupt wakeup mask 1: Interrupt wakeup not masked
7	IM7	RW	0	EXTI line7 is used as interrupt wake-up CPU mask control. 0: Interrupt wakeup mask 1: Interrupt wakeup not masked
6	IM6	RW	0	EXTI line6 is used as interrupt wake-up CPU mask control. 0: Interrupt wakeup mask 1: Interrupt wakeup not masked
5	IM5	RW	0	EXTI line5 is used as interrupt wake-up CPU mask control. 0: Interrupt wakeup mask 1: Interrupt wakeup not masked
4	IM4	RW	0	EXTI line4 is used as interrupt wake-up CPU mask control. 0: Interrupt wakeup mask 1: Interrupt wakeup not masked
3	IM3	RW	0	EXTI line3 is used as interrupt wake-up CPU mask control. 0: Interrupt wakeup mask

Bit	Name	R/W	Reset Value	Function
				1: Interrupt wakeup not masked
2	IM2	RW	0	EXTI line2 is used as interrupt wake-up CPU mask control. 0: Interrupt wakeup mask 1: Interrupt wakeup not masked
1	IM1	RW	0	EXTI line1 is used as interrupt wake-up CPU mask control. 0: Interrupt wakeup mask 1: Interrupt wakeup not masked
0	IM0	RW	0	EXTI line0 is used as interrupt wake-up CPU mask control. 0: Interrupt wakeup mask 1: Interrupt wakeup not masked

13.3.10. Event Mask Register (EXTI_EMR)

Address offset: 0x84

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	EM2 9	Res	Res	Res	Res	Res	Res	Res	Res	EM2 0	EM1 9	EM1 8	EM1 7	EM1 6
-	-	RW	-	-	-	-	-	-	-	-	RW	RW	RW	RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EM1 5	EM1 4	EM1 3	EM1 2	EM1 1	EM1 0	EM 9	EM 8	EM 7	EM 6	EM 5	EM4	EM3	EM2	EM1	EM0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 30	Res	-	-	Res
29	EM29	RW	0	EXTI line29 is used as an event to wake up the CPU mask control. 0: event wakeup mask 1: Event wakeup not blocked
28: 21	Res	-	-	Res
20	EM20	RW	0	EXTI line20 is used as an event to wake up the CPU mask control. 0: event wakeup mask 1: Event wakeup not blocked

Bit	Name	R/W	Reset Value	Function
19	EM19	RW	0	EXTI line19 is used as an event to wake up the CPU mask control. 0: event wakeup mask 1: Event wakeup not blocked
18	EM18	RW	0	EXTI line18 is used as an event wake-up CPU mask control. 0: event wakeup mask 1: Event wakeup not blocked
17	EM17	RW	0	EXTI line17 is used as an event to wake up the CPU mask control. 0: event wakeup mask 1: Event wakeup not blocked
16	EM16	RW	0	EXTI line16 is used as an event wake-up CPU mask control. 0: event wakeup mask 1: Event wakeup not blocked
15	EM15	RW	0	EXTI line15 is used as an event wake-up CPU mask control. 0: event wakeup mask 1: Event wakeup not blocked
14	EM14	RW	0	EXTI line14 serves as the event wake-up CPU mask control. 0: event wakeup mask 1: Event wakeup not blocked
13	EM13	RW	0	EXTI line13 is used as an event wake-up CPU mask control. 0: event wakeup mask 1: Event wakeup not blocked
12	EM12	RW	0	EXTI line12 is used as an event wake-up CPU mask control. 0: event wakeup mask 1: Event wakeup not blocked
11	EM11	RW	0	EXTI line11 is used as an event to wake up the CPU mask control. 0: event wakeup mask 1: Event wakeup not blocked
10	EM10	RW	0	EXTI line10 is used as an event to wake up the CPU mask control. 0: event wakeup mask

Bit	Name	R/W	Reset Value	Function
				1: Event wakeup not blocked
9	EM9	RW	0	EXTI line9 is used as an event to wake up the CPU mask control. 0: event wakeup mask 1: Event wakeup not blocked
8	EM8	RW	0	EXTI line8 is used as an event to wake up the CPU mask control. 0: event wakeup mask 1: Event wakeup not blocked
7	EM7	RW	0	EXTI line7 is used as an event to wake up the CPU mask control. 0: event wakeup mask 1: Event wakeup not blocked
6	EM6	RW	0	EXTI line6 is used as an event wake-up CPU mask control. 0: event wakeup mask 1: Event wakeup not blocked
5	EM5	RW	0	EXTI line5 is used as an event to wake up the CPU mask control. 0: event wakeup mask 1: Event wakeup not blocked
4	EM4	RW	0	EXTI line4 is used as an event to wake up the CPU mask control. 0: event wakeup mask 1: Event wakeup not blocked
3	EM3	RW	0	EXTI line3 is used as an event wake-up CPU mask control. 0: event wakeup mask 1: Event wakeup not blocked
2	EM2	RW	0	EXTI line2 is used as an event to wake up the CPU mask control. 0: event wakeup mask 1: Event wakeup not blocked
1	EM1	RW	0	EXTI line1 is used as an event to wake up the CPU mask control. 0: event wakeup mask 1: Event wakeup not blocked

Bit	Name	R/W	Reset Value	Function
0	EM0	RW	0	EXTI line0 is used as an event wake-up CPU mask control. 0: event wakeup mask 1: Event wakeup not blocked

14. Cyclic Redundancy Check (CRC)

14.1. Introduction

Based on the generating polynomial, the CRC calculation unit takes the input 32-bit data and operates to produce a CRC result.

14.2. CRC Key Features

- Using the CRC-32 (Ethernet) polynomial: $0x4C11DB7$
 $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$
- Supports 32-bit data input
- Single input/output 32 data and result outputs share a common register
- General purpose 8-bit register (can be used to store temporary data)
- Calculation time: 4 AHB clocks for 32 bits of data

14.3. CRC Functional Description

14.3.1. CRC Block Diagram

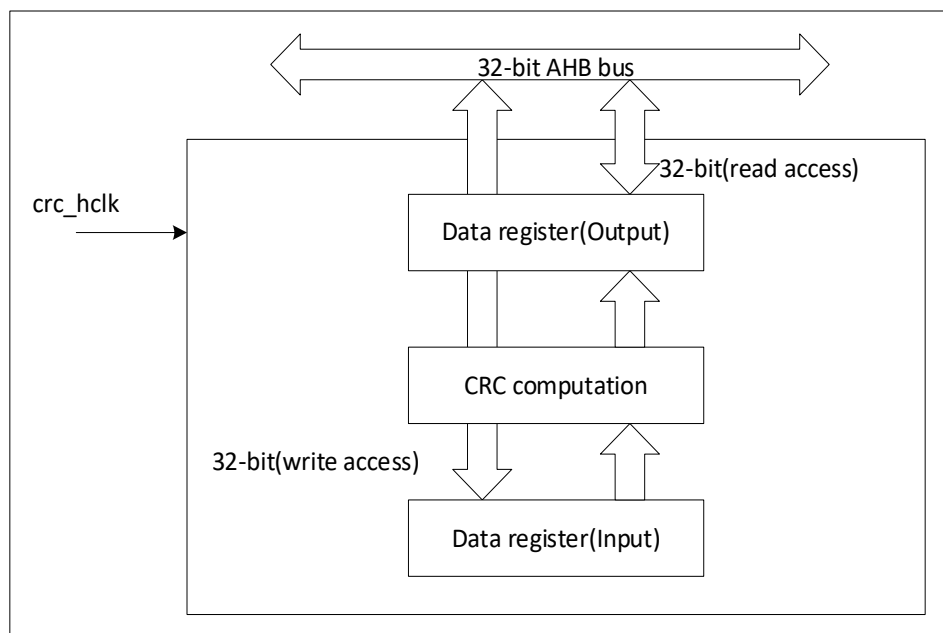


Figure 14-1 CRC calculation unit block diagram

The CRC calculation unit contains one 32-bit data register (CRC_DR):

- When a write operation is performed to this register, it serves as an input register for new data to be entered for CRC calculation.
- A read operation to this register returns the result of the last CRC calculation.

For each write to the data register, the result of the calculation is a combination of the result of the previous CRC calculation and the result of the new calculation (a CRC calculation is performed on the entire 32-bit word, not byte by byte).

Support for configuring the CRC initial value.

Register CRC_DR can be reset to 0xFFFF FFFF by setting the RESET bit of register CRC_CR. This operation does not affect the data in register CRC_IDR.

14.4. CRC Register

14.4.1. 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	Name	R/W	Reset Value	Function
31: 0	DR	RW	0xFFFF FFFF	Data Register. When new data is written, it is used as an input register. When read, the previous CRC calculation is maintained.

14.4.2. Independent Data Register (CRC_IDR)

Address offset: 0x04

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 8	Res	-	-	Res
7: 0	IDR [7:0]	RW	0	General purpose 8-bit data register bits. Can be used to temporarily store 1 byte of data. The CRC reset generated by the RESET bit of register CRC_CR has no effect on this register. Note: This register is not involved in CRC calculations and can hold any data.

14.4.3. Control Register (CRC_CR)

Address offset: 0x08

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 1	Res	-	-	Res
0	RESET	W	0	A software set will reset the CRC module. Software can only write 1, which is cleared by hardware.

15. Digital/Analog Conversion (DAC)

15.1. DAC Introduction

Digital/Analog Conversion Modules (DACs) are 12-bit digital input, voltage output digital/analog converters. The DAC can be configured in 8-bit or 12-bit mode and can also be used with a DMA controller. When the DAC operates in 12-bit mode, the data can be set to be left-aligned or right-aligned. The DAC module has 2 output channels, each with a separate converter. In Dual DAC mode, the 2 channels can be converted independently or simultaneously and the outputs of the 2 channels can be updated synchronously.

15.2. DAC Main Features

- 2 DAC converters: each converter corresponds to 1 output channel;
- Data is left- or right-aligned in 12-bit mode;
- Synchronized update function;
- Noise waveform generation;
- Triangular waveform generation;
- Dual DAC channels are converted simultaneously or separately;
- Each channel has a DMA function;
- Supports DMA underflow error detection;
- Externally triggered conversion;

The block diagram of a single DAC channel is shown below

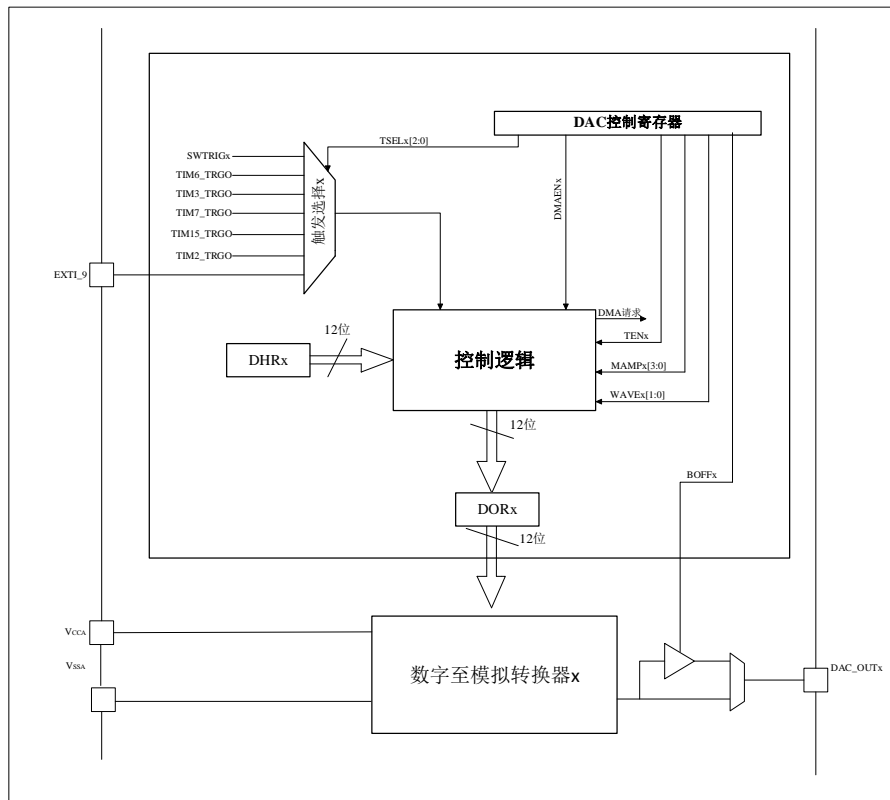


Figure 15-1 Single DAC channel block diagram

Table 15-1 DAC Pinout

name (of a thing)	Model Type	descriptive
VCCA	Input, Analog Power	analog power
VSSA	Input, Analog Power Ground	Analog Power Ground
DAC_OUTx	Analog Output Signal	DACx Analog Output

Note: Before enabling the DAC module, the GPIO ports (PA4 corresponds to DAC channel 1 and PA5 corresponds to DAC channel 2) should be configured in analog mode.

15.3. DAC Functional Description

15.3.1. Using DAC channels

Power to DAC channel x is turned on by setting the ENx position of the DAC_CR register to '1'. After a startup period (tWAKEUP), DAC channel x is enabled.

Note: The ENx bit will only enable the analog portion of DAC channel x. The digital portion of DAC channel x will still operate even if this bit is set to '0'.

15.3.2. Using the DAC Output Buffer

The DAC integrates two output buffers that can be used to reduce output impedance and directly drive external loads without the need for external op-amps. Each DAC channel output buffer can be enabled or disabled by setting the BOFFx bit in the DAC_CR register.

15.3.3. DAC Data Format

Depending on the configuration mode selected, data is written to the specified registers as described below:

- Single DAC channel x with 3 cases:
 - 8-bit data right-aligned: user has to write data into register DAC_DHR8Rx[7:0] bits (actually deposited into register DHRx[11:4] bits)
 - 12-bit data left-aligned: user has to write data into register DAC_DHR12Lx[15:4] bits (actually deposited into register DHRx[11:0] bits)
 - 12-bit data right-aligned: user has to write data into register DAC_DHR12Rx[11:0] bits (actually deposited into register DHRx[11:0] bits)

Based on the operation of the DAC_DHRyyyx (yyyy refers to different data formats and alignments) register, the written data is dumped into the DAC_DHRx register after the corresponding shifts (DHRx is the internal data saving register x). Subsequently, the contents of the DAC_DHRx register are either automatically transferred to the DAC_DORx register, or are transferred to the DAC_DORx register via a software trigger or external event trigger.

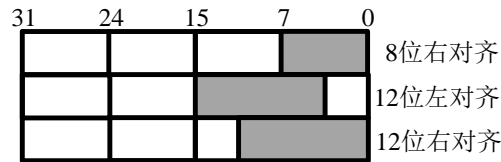


Figure 15-2 Data register for single DAC channel mode

■ Dual DAC channels with 3 cases:

- 8-bit data right-aligned: user has to write DAC channel 1 data into register DAC_DHR8RD[7:0] bits (actually into register DHR1[11:4] bits), and write DAC channel 2 data into register DAC_DHR8RD[15:8] bits (actually into register DHR2[11:4] bits)
- 12-bit data left-aligned: user has to write DAC channel 1 data into register DAC_DHR12LD[15:4] bits (actually into register DHR1[11:0] bits), and write DAC channel 2 data into register DAC_DHR12LD[31:20] bits (actually into register DHR2[11:0] bits)
- 12-bit data right-aligned: user has to write DAC channel 1 data into register DAC_DHR12RD[11:0] bits (actually stored in register DHR1[11:0] bits), and write DAC channel 2 data into register DAC_DHR12RD[27:16] bits (actually stored in register DHR2[11:0] bits)

Based on the operation of the DAC_DHRyyyx (yyyy refers to different data formats and alignments) registers, the written data is dumped into the DAC_DHR1 and DAC_DHR2 registers after the corresponding shifts (DAC_DHR1 and DAC_DHR2 are internal data saving registers x). Subsequently, the contents of DAC_DHR1 and DAC_DHR2 are automatically transferred to the DAC_DORx registers either by a software trigger or by an external event trigger.

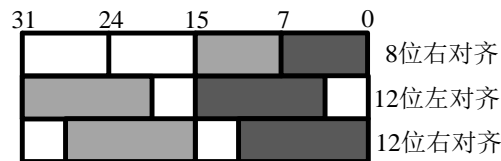


Figure 15-3 Data registers for dual DAC channel mode

15.3.4. DAC conversion

It is not possible to write directly to register DAC_DORx; any data output to DAC channel x must be written to the DAC_DHRx register (data is actually written to the DAC_DHR8Rx, DAC_DHR12Lx, DAC_DHR12Rx, DAC_DHR8RD, DAC_DHR12LD, or DAC_DHR12RD (registers).

If the hardware trigger is not checked (TENx position '0' of register DAC_CRx), the data deposited into register DAC_DHRx is automatically passed to register DAC_DORx after 1 APB clock cycle. If the hardware trigger is checked (TENx position '1' of register DAC_CR), the data transfer is completed 3 APB clock cycles after the trigger occurs.

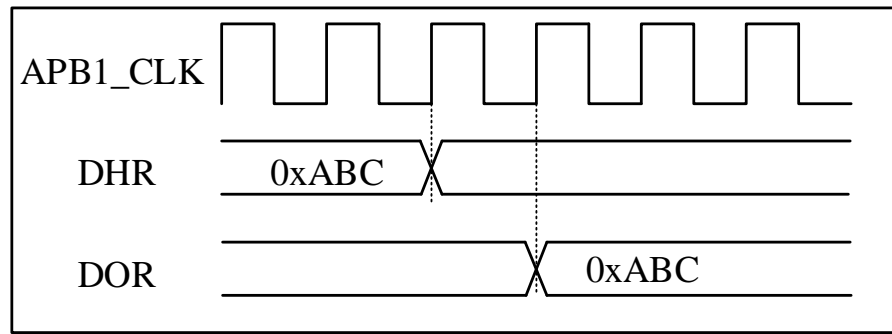


Figure 15-4 Timing block diagram of conversion when TEN=0

15.3.5. DAC Output Voltage

The digital input is linearly converted by the DAC to an analog voltage output ranging from 0 to VCCA.

The output voltage on any DAC channel pin satisfies the following relationship:

$$\text{DAC output} = \text{VCCA} \times (\text{DOR} / 4095).$$

15.3.6. Select DAC Trigger

If the TENx bit is set to 1, the DAC conversion can be triggered by some external event (timer counter, external interrupt line). Configuration control bits TSELx[2:0] select one of seven trigger events to trigger the DAC conversion.

Table 15-2 External Trigger

trigger source	typology	TSELx [2:0]
Timer 6 TRGO event	Internal signal from on-chip timer	000
Timer 3 TRGO event		001
Timer 7 TRGO event		010
Timer 15 TRGO event		011
Timer 2 TRGO event		100
EXTI Line 9	external pin	110
SWTRIG (software triggered)	software control bits	111

Each time the DAC interface detects a rising edge from the selected timer TRGO or external interrupt line 9, the last data stored in register DAC_DHRx is transferred to register DAC_DORx. After 3 APB clock cycles, register DAC_DORx is updated to the new value.

If software triggering is selected, the conversion starts as soon as the SWTRIG position is '1'. The SWTRIG bit is automatically cleared '0' by hardware after data is transferred from the DAC_DHRx register to the DAC_DORx register.

Attention:

1. It is not possible to change the TSELx[2:0] bits while ENx is '1'.
2. If software triggering is selected, data transfer from register DAC_DHRx to register DAC_DORx takes only 1 APB clock cycle.
3. The Trigger trigger frequency cannot exceed 1 MHz.

15.3.7. DMA function

15.3.7.1. DMA request

Either DAC channel has DMA capability. The 2 DMA channels can be used for DMA requests for each of the 2 DAC channels.

If the DMAENx position '1', once an external trigger (not a software trigger) occurs, a DMA request is generated after 3 APB clock cycles and then the data in the DAC_DHRx register is transferred to the DAC_DORx register.

In dual DAC mode (i.e., using DAC_DHR12RD or DAC_DHR12LD or DAC_DHR8RD registers), only one bit in DMAENx should be set.

15.3.7.2. DMA underflow detection

The DAC's DMA request has no buffer queue, so if the second external trigger arrives before the answer to the first external trigger (the first request) is received, no new DMA request is issued and the DMA underflow flag DMAUDRx in the DAC_SR register is set to "1" to report an error condition. DMA data transfers are then disabled and no further DMA requests are processed. The DAC channel continues to convert the old data.

Software should clear the DMAUDRx flag to zero by writing a "1", clear the DMAEN bit of the used DMA channel to zero, and reinitialize the DMA and DAC channels in order to restart the DMA transfer properly. The software should also modify the DAC trigger conversion frequency to reduce the DMA workload to avoid recurring DMA underflow.

For each DAC channel, if the corresponding DMAUDRIEx bit in the DAC_CR register is enabled, the corresponding interrupt will also be generated.

15.3.7.3. Noise generation

Linear Feedback Shift Register LFSR can be utilized to generate amplitude varying pseudo-noise. Setting the DAC_WAVE[1:0] bits to '01' selects the DAC noise generation function. The preloaded value of register LFSR is 0xAAA. The value of this register is updated after 3 APB clock cycles after each trigger event according to a specific algorithm.

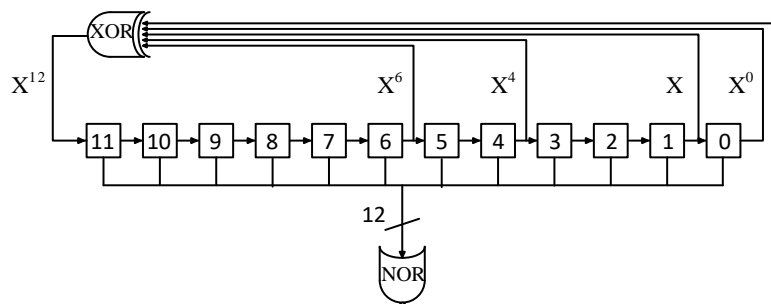


Figure 15-5 DAC LFSR register algorithm

Setting the MAMPx[3:0] bits of the DAC_CR register can mask some or all of the LFSR data, so that the value of the LFSR obtained is added to the value of the DAC_DHRx, and if the value obtained is overflowed, the maximum value that can be retained by the register will be written to the

DAC_DORx, and if the value obtained is not overflowed, the value will be written to the DAC_DORx.

If the register LFSR value is 0x000, a '1' is injected (anti-locking mechanism).

Positioning WAVEx[1:0] at '0' resets the LFSR waveform generation algorithm.

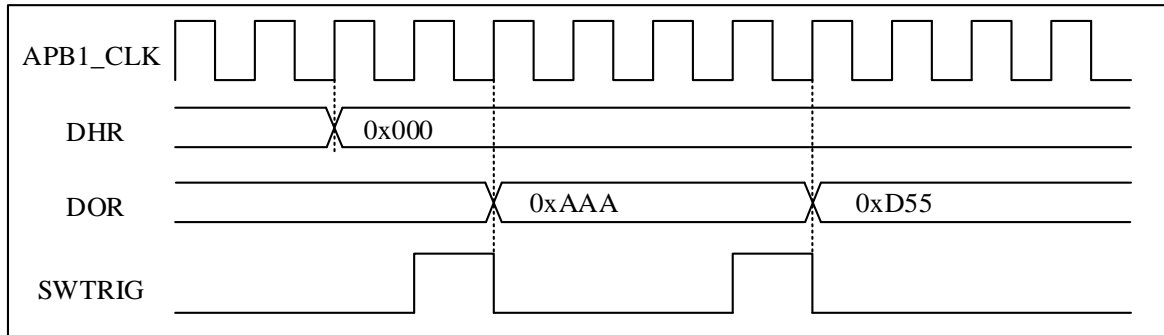


Figure 15-6 DAC conversion with LFSR waveform generation (enabling software trigger)

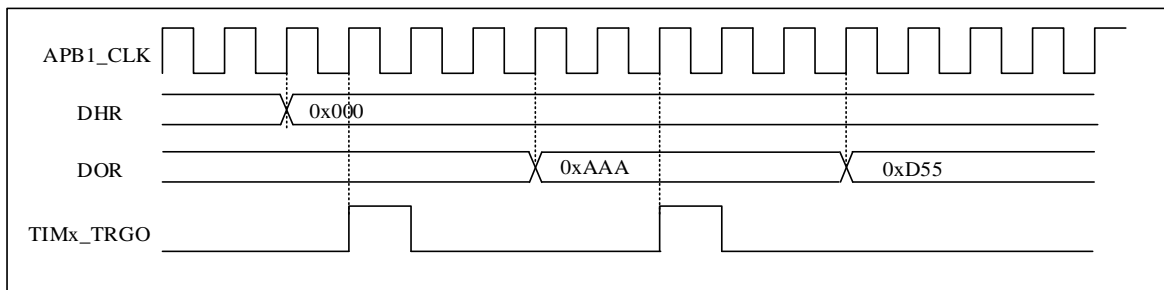


Figure 15-7 DAC conversion with LFSR waveform generation (enabling hardware trigger)

Note: 1. In order to generate noise, DAC triggering must be enabled, i.e., the TENx bit of the DAC_CR register must be set to '1';

2. If the value of DAC_DHR is changed, the value of the LFSR register is reset to 0xAAA;

3. If DAC_CR.TENx decreases from "1" to "0", the value of LFSR register will also be reset to 0xAAA.

15.3.7.4. Triangle wave generation

Setting the DAC_CR.WAVEx[1:0] bits to "10" selects the triangle wave generation function of the DAC. Set the MAMPx[3:0] bits of the DAC_CR register to select the amplitude of the triangle wave. The internal triangle wave counter accumulates 1 after 3 APB clock cycles after each trigger event. The counter value is added to the value of the DAC_DHRx register and the overflow bit is discarded and the DAC_DORx register is written. The delta wave counter accumulates gradually when the value passed into the DAC_DORx register is less than the maximum amplitude defined by the DAC_CR.MAMP[3:0] bits. Once the set maximum amplitude is reached, the counter starts to decrease, reaches 0 and then starts to accumulate, and so on.

Positioning DAC_CR.WAVEx[1:0] at '0' resets the triangle wave generation.

Note: When the base value is the maximum value of register DAC_DORx, the output of DAC_DORx is its maximum value regardless of the value of DAC_CR.MAMPx.

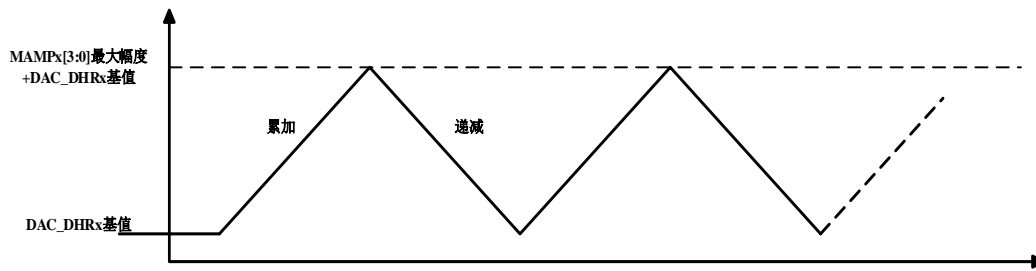


Figure 15-8 DAC Triangle Wave Generation

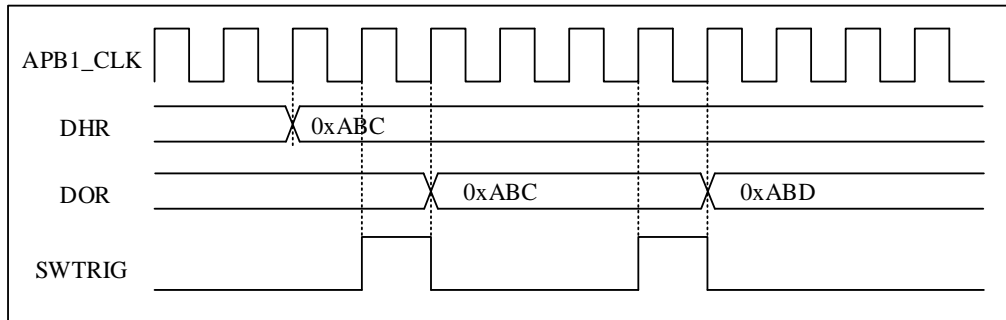


Figure 15- 9 DAC conversion with delta generation (enabling software trigger)

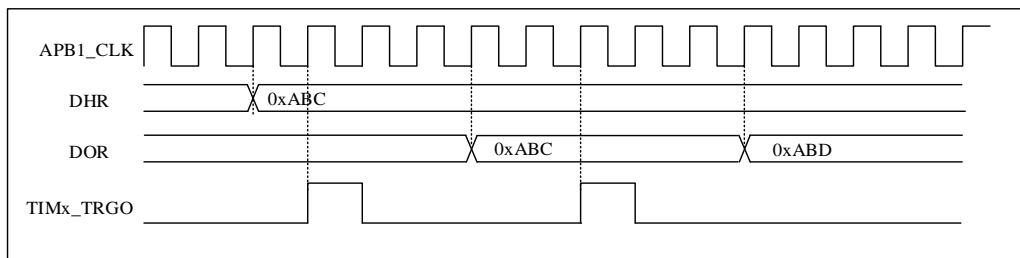


Figure 15- 10 DAC conversion with delta generation (enabling hardware trigger)

Note: 1. In order to generate a triangle wave, DAC triggering must be enabled, i.e., set the TENx bit of the DAC_CR register to '1';

2. The DAC_CR.MAMP[3:0] bits must be set before enabling the DAC or their value cannot be modified;

3. If the value of DAC_DHR is changed, the delta wave counter is reset to 0;

4. If DAC_CR.TENx decreases from "1" to "0", the delta wave counter is also reset to zero.

15.3.7.5. Dual DAC channel conversion

In the case where two DACs are required to operate simultaneously, in order to utilize the bus bandwidth more efficiently, the DAC integrates three registers for dual DAC mode: DHR8RD, DHR12RD, and DHR12LD, which require only one register to be accessed to complete the operation of driving two DAC channels simultaneously.

For dual DAC channel conversions and these dedicated registers, a total of 11 conversion modes are available. These conversion modes can still be operated through the separate DHRx registers when only one DAC channel is used.

All models are detailed in the following sections.

15.3.7.6. No waveform generator

Set the DAC to operate in this conversion mode in the following order:

- Set the trigger enable bits DAC_CR.TEN1 and DAC_CR.TEN2 of the 2 DAC channels to '1' respectively;
- Configure different trigger sources for each of the 2 DAC channels by setting the DAC_CR.TSEL1[2:0] and DAC_CR.TSEL2[2:0] bits to different values;
- Load the dual DAC channel conversion data into the desired DHR register (DHR12RD, DHR12LD, or DHR8RD).

When a DAC channel 1 trigger event occurs, (after a delay of 3 APB clock cycles) the value of register DAC_DHR1 is passed into register DAC_DOR1.

When a DAC channel 2 trigger event occurs, (after a delay of 3 APB clock cycles) the value of register DAC_DHR2 is passed into register DAC_DOR2.

15.3.7.7. Independent triggers using the same LFSR

Set the DAC to operate in this conversion mode in the following order:

- Set the trigger enable bits DAC_CR.TEN1 and DAC_CR.TEN2 of the 2 DAC channels to '1' respectively;
- Configure different trigger sources for each of the 2 DAC channels by setting the DAC_CR.TSEL1[2:0] and DAC_CR.TSEL2[2:0] bits to different values;
- Set the DAC_CR.WAVEx[1:0] bits of the 2 DAC channels to "01" and set DAC_CR.MAMPx[3:0] to the same LFSR mask value;
- Load the dual DAC channel conversion data into the desired DHR register (DHR12RD, DHR12LD, or DHR8RD).

When a DAC channel 1 trigger event occurs, the LFSR1 counter value with the same mask is added to the DAC_DHR1 register value, and (after a delay of 3 APB clock cycles) the result is passed into register DAC_DOR1, which then updates the LFSR1 counter. When a DAC channel 2 trigger event occurs, the LFSR2 counter value with the same mask is added to the DAC_DHR2 register value, and (after a delay of 3 APB clock cycles) the result is passed into register DAC_DOR2, which then updates the LFSR2 counter.

15.3.7.8. Independent triggering using different LFSRs

Set the DAC to operate in this conversion mode in the following order:

- Set the trigger enable bits DAC_CR.TEN1 and DAC_CR.TEN2 of the 2 DAC channels to '1' respectively;
- Configure different trigger sources for each of the 2 DAC channels by setting the DAC_CR.TSEL1[2:0] and DAC_CR.TSEL2[2:0] bits to different values;
- Set the DAC_CR.WAVEx[1:0] bits of the 2 DAC channels to "01" and set DAC_CR.MAMPx[3:0] to different LFSR mask values;
- Load the dual DAC channel conversion data into the desired DHR register (DHR12RD, DHR12LD, or DHR8RD).

When a DAC channel 1 trigger event occurs, the LFSR1 counter value masked according to DAC_CR.MAMP1[3:0] is added to the DAC_DHR1 register value, and (after a delay of 3 APB clock cycles) the result is passed into register DAC_DOR1, which then updates the LFSR1 counter. When a DAC channel 2 trigger event occurs, the LFSR2 counter value masked according to DAC_CR.MAMP2[3:0] is added to the DAC_DHR2 register value, and (after a delay of 3 APB clock cycles) the result is passed into register DAC_DOR2, which then updates the LFSR2 counter.

15.3.7.9. Separate triggers that generate the same triangle wave

Set the DAC to operate in this conversion mode in the following order:

- Set the trigger enable bits DAC_CR.TEN1 and DAC_CR.TEN2 of the 2 DAC channels to '1' respectively;
- Configure different trigger sources for each of the 2 DAC channels by setting the DAC_CR.TSEL1[2:0] and DAC_CR.TSEL2[2:0] bits to different values;
- Set the DAC_CR.WAVEx[1:0] bits of the 2 DAC channels to "1x" and set DAC_CR.MAMPx[3:0] to the same triangle wave amplitude;
- Load the dual DAC channel conversion data into the desired DHR register (DHR12RD, DHR12LD, or DHR8RD).

When a DAC channel 1 trigger event occurs, the same triangular wave amplitude is added to the value of the DAC_DHR1 register, and (after a delay of 3 APB clock cycles) the result is passed into register DAC_DOR1, which then updates the DAC channel 1 triangular wave counter. When a DAC channel 2 trigger event occurs, the same triangular wave amplitude is added to the value of the DAC_DHR2 register, and (after a delay of 3 APB clock cycles) the result is passed into register DAC_DOR2, which then updates the DAC channel 2 triangular wave counter.

15.3.7.10. Separate triggers for generating different triangle waves

Set the DAC to operate in this conversion mode in the following order:

- Set the trigger enable bits DAC_CR.TEN1 and DAC_CR.TEN2 of the 2 DAC channels to '1' respectively;
- Configure different trigger sources for each of the 2 DAC channels by setting the DAC_CR.TSEL1[2:0] and DAC_CR.TSEL2[2:0] bits to different values;
- Set the DAC_CR.WAVEx[1:0] bits of the 2 DAC channels to "1x" and set DAC_CR.MAMPx[3:0] to different triangle wave amplitudes;
- Load the dual DAC channel conversion data into the desired DHR register (DHR12RD, DHR12LD, or DHR8RD).

When a DAC channel 1 trigger event occurs, the different triangular wave amplitudes are added to the value of the DAC_DHR1 register, and (after a delay of 3 APB clock cycles) the result is passed into register DAC_DOR1, which then updates the DAC channel 1 triangular wave counter. When a DAC channel 2 trigger event occurs, the different delta wave amplitudes are added to the value of the DAC_DHR2 register, and (after a delay of 3 APB clock cycles) the result is passed into register DAC_DOR2, which then updates the DAC channel 2 delta wave counter.

15.3.7.11. Simultaneous software startup

Follow the procedure below to set the DAC to work in this conversion mode:

- Load the dual DAC channel conversion data into the desired DHR register (DHR12RD, DHR12LD, or DHR8RD).

In this configuration, after one APB clock cycle, the values of the DHR1 and DHR2 registers are passed into the DAC_DOR1 and DAC_DOR2 registers, respectively.

15.3.7.12. Simultaneous triggering without waveform generator

Set the DAC to operate in this conversion mode in the following order:

- Set the trigger enable bits DAC_CR.TEN1 and DAC_CR.TEN2 of the 2 DAC channels to '1' respectively;
- Configure the 2 DAC channels to use the same trigger source by setting the DAC_CR.TSEL1[2:0] and DAC_CR.TSEL2[2:0] bits to the same value, respectively;
- Load the dual DAC channel conversion data into the desired DHR register (DHR12RD, DHR12LD, or DHR8RD).

When a trigger event occurs, (after a delay of 3 APB clock cycles) the values of the DAC_DHR1 and DAC_DHR2 registers are passed into the DAC_DOR1 and DAC_DOR2 registers, respectively.

15.3.7.13. Simultaneous triggering using the same LFSR

Set the DAC to operate in this conversion mode in the following order:

- Set the trigger enable bits DAC_CR.TEN1 and DAC_CR.TEN2 of the 2 DAC channels to '1' respectively;
- Configure the 2 DAC channels to use the same trigger source by setting the DAC_CR.TSEL1[2:0] and DAC_CR.TSEL2[2:0] bits to the same value, respectively;
- Set the DAC_CR.WAVEx[1:0] bits of the 2 DAC channels to "01" and set DAC_CR.MAMPx[3:0] to the same LFSR mask value;
- Load the dual DAC channel conversion data into the desired DHR register (DHR12RD, DHR12LD, or DHR8RD);

When a trigger event occurs, the LFSR1 counter value of the mask set by DAC_CR.MAMP1[3:0] is added to the value of the DAC_DHR1 register, and (after a delay of 3 APB clock cycles) the result is passed into the DAC_DOR1 register, which then updates the LFSR1 counter.

Similarly, the LFSR2 counter value of the mask set by DAC_CR.MAMP2[3:0] is added to the value of the DAC_DHR2 register, and (after a delay of 3 APB clock cycles) the result is passed into register DAC_DOR2, which then updates the LFSR2 counter.

15.3.7.14. Simultaneous triggering using different LFSRs

Set the DAC to operate in this conversion mode in the following order:

- Set the trigger enable bits DAC_CR.TEN1 and DAC_CR.TEN2 of the 2 DAC channels to '1' respectively;
- Configure the 2 DAC channels to use the same trigger source by setting the DAC_CR.TSEL1[2:0] and DAC_CR.TSEL2[2:0] bits to the same value, respectively;

- Set the DAC_CR.WAVEx[1:0] bits of the 2 DAC channels to "01" and set DAC_CR.MAMPx[3:0] to different LFSR mask values;
- Load the dual DAC channel conversion data into the desired DHR register (DHR12RD, DHR12LD, or DHR8RD).

When a trigger event occurs, the LFSR1 counter value with the same mask is added to the DAC_DHR1 register value, and (after a delay of 3 APB clock cycles) the result is passed into register DAC_DOR1, which then updates the LFSR1 counter.

At the same time, the LFSR2 counter value with the same mask is added to the DAC_DHR2 register value, and (after a delay of 3 APB clock cycles) the result is passed into register DAC_DOR2, which then updates the LFSR2 counter.

15.3.7.15. Simultaneous triggering using the same triangle wave generator

Set the DAC to operate in this conversion mode in the following order:

- Set the trigger enable bits DAC_CR.TEN1 and DAC_CR.TEN2 of the 2 DAC channels to '1' respectively;
- By setting DAC_CR.TSEL1[2:0] and DAC_CR.TSEL2[2:0] bits are the same value to configure the 2 DAC channels separately to use the same trigger source.
- Set the DAC_CR.WAVEx[1:0] bits of the 2 DAC channels to "1x" and set DAC_CR.MAMPx[3:0] to the same triangle wave amplitude.
- Load the dual DAC channel conversion data into the desired DHR register (DHR12RD, DHR12LD, or DHR8RD).

When a trigger event occurs, the same delta wave amplitude is added to the DAC_DHR1 register value and (after a delay of 3 APB clock cycles) the result is passed into register DAC_DOR1, which then updates the delta wave counter.

At the same time, the same triangular wave amplitude is added to the DAC_DHR2 register value and (after a delay of 3 APB clock cycles) the result is passed into register DAC_DOR2, which then updates the triangular wave counter.

15.3.7.16. Simultaneous triggering using different triangle wave generators

Set the DAC to operate in this conversion mode in the following order:

- Set the trigger enable bits DAC_CR.TEN1 and DAC_CR.TEN2 of the 2 DAC channels to '1' respectively;
- Configure the 2 DAC channels to use the same trigger source by setting the DAC_CR.TSEL1[2:0] and DAC_CR.TSEL2[2:0] bits to the same value, respectively.
- Set the DAC_CR.WAVEx[1:0] bits of the 2 DAC channels to "1x" and set DAC_CR.MAMPx[3:0] to different triangle wave amplitudes.
- Load the dual DAC channel conversion data into the desired DHR register (DHR12RD, DHR12LD, or DHR8RD).

When a trigger event occurs, the delta wave amplitude value set by DAC_CR.MAMP1[3:0] is added to the DAC_DHR1 register value, and (after a delay of 3 APB clock cycles) the result is passed into register DAC_DOR1, which then updates the delta wave counter.

At the same time, the delta wave amplitude set by MAMP2[3:0] is added to the DAC_DHR2 register value, and (after a delay of 3 APB clock cycles) the result is passed into register DAC_DOR2, which then updates the delta wave counter.

DAC Output Clock

The DAC output clock (DAC_CLK) is the analog side sampling DAC_DOR data clock, which is valid only when DAC_CR.ENx is "1". When software trigger is selected, set "1" for 3 APB cycles after the rising edge of software trigger is detected, and clear 0 after keeping 3 APB cycles; when no trigger is selected, set "1" for 2 APB cycles after the data change is detected, and clear 0 after keeping 3 APB cycles.

Note: When TEN drops from "1" to "0" for one more APB cycle, the data in DAC_DOR will be updated to the data in DAC_DHR register. If the data in the DAC_DHR register has been updated, a DAC_CLK is generated when DAC_DOR is updated to the value of DAC_DHR; conversely, no DAC_CLK is generated.

15.4. DAC Register

These peripheral registers must be operated in word (32-bit) format.

15.4.1. DAC Control Register (DAC_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	DMAUD RIE2	DMAE N2	MAMP2[3: 0]				WAVE2[1: 0]		TSEL2[2: 0]			TEN2	BOF F2	EN2
-	-	RW	RW	RW				RW		RW			RW	RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	DMAUD RIE1	DMAE N1	MAMP1[3: 0]				WAVE1[1: 0]		TSEL1[2: 0]			TEN1	BOF F1	EN1
-	-	RW	RW	RW				RW		RW			RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31	Res	-	-	Res
30	Res	-	-	Res
29	DMAUDRIE2	RW	0	DAC channel 2 DMA underflow interrupt enable This bit is set and cleared by software. 0: Do not enable DAC channel 2 DMA underflow interrupt; 1: Enable DAC channel 2 DMA underflow interrupt.
28	DMAEN2	RW	0	DAC channel2 DMA enable (DAC channel2 DMA enable)

				<p>This bit is set and cleared by software.</p> <p>0: Disables DAC channel 2 DMA mode;</p> <p>1: Enable DAC channel 2 DMA mode.</p>
27: 24	MAMP2[3: 0]	RW	0	<p>DAC channel2 mask/amplitude selector</p> <p>Set by software, these bits are used to select the mask bit in noise generation mode and the amplitude of the waveform in triangle wave generation mode.</p> <p>0000: Unmasked LSFR bit 0/triangular wave amplitude equal to 1;</p> <p>0001: Unmasked LSFR bits [1:0]/Triangle wave amplitude equal to 3;</p> <p>0010: Unmasked LSFR bits [2:0]/Triangle wave amplitude equal to 7;</p> <p>0011: Unmasked LSFR bits [3:0]/Triangle wave amplitude equal to 15;</p> <p>0100: Unmasked LSFR bits [4:0]/Triangle wave amplitude equal to 31;</p> <p>0101: Unmasked LSFR bits [5:0]/Triangle wave amplitude equal to 63;</p> <p>0110: Unmasked LSFR bits [6:0]/Triangle wave amplitude equal to 127;</p> <p>0111: Not masked LSFR bits [7:0] / Triangular wave amplitude equal to 255;</p> <p>1000: Unmasked LSFR bits [8:0]/Triangle wave amplitude equal to 511;</p> <p>1001: Unmasked LSFR bits [9:0]/Triangle wave amplitude equal to 1023;</p> <p>1010: Unmasked LSFR bits [10:0]/Triangle wave amplitude equal to 2047;</p> <p>≥1011: unmasked LSFR bits [11:0]/triangular wave amplitude equal to 4095;</p>
23: 22	WAVE2[1: 0]	RW	0	<p>DAC channel2 noise/triangle wave generation enable (DAC channel2 noise/triangle wave generation enable)</p> <p>This 2-bit is set and cleared by software.</p> <p>00: Turns off the waveform generator;</p> <p>01: Enable the noise waveform generator;</p> <p>1x: Enable the triangle wave generator.</p>
21: 19	TSEL2[2: 0]	RW	0	<p>DAC channel2 trigger selection (DAC channel2 trigger selection)</p>

				<p>This 3-bit is used to select the external trigger event for DAC channel 2.</p> <p>000: TIM6 TRGO incident</p> <p>001: TIM3 TRGO incident</p> <p>010: TIM7 TRGO incident</p> <p>011: TIM15 TRGO incident</p> <p>100: TIM2 TRGO incident</p> <p>101: Reserved;</p> <p>110: External interrupt line 9</p> <p>111: Software triggers</p> <p>Note: This 3-bit can only be used when TEN2 = 1 (DAC channel 2 trigger enable).</p>
18	TEN2	RW	0	<p>DAC channel2 trigger enable (DAC channel2 trigger enable)</p> <p>This bit is set and cleared by software to enable/disable the triggering of DAC channel 2.</p> <p>0: Turns off DAC channel 2 triggering and data written to the DAC_DHRx register is transferred to the DAC_DOR2 register after 1 APB clock cycle;</p> <p>1: Enable DAC channel 2 trigger, data written to DAC_DHRx register is transferred to DAC_DOR2 register after 3 APB clock cycles.</p> <p>Note: If software triggering is selected, data written to register DAC_DHRx requires only one APB always cycle to pass into register DAC_DOR2.</p>
17	BOFF2	RW	0	<p>DAC channel2 output buffer disable</p> <p>This bit is set and cleared by software to enable/disable the output buffer for DAC channel 2.</p> <p>0: Enable DAC channel 2 output buffering;</p> <p>1: Disables the DAC channel 2 output buffer.</p>
16	EN2	RW	0	<p>DAC channel2 enable (DAC channel2 enable)</p> <p>This bit is set and cleared by software to enable/disable DAC channel 2.</p> <p>0: Close DAC channel 2</p> <p>1: Enable DAC channel 2</p>
15	Res	-	-	Res
14	Res	-	-	Res
13	DMAUDRIE1	RW	0	<p>DAC channel 1 DMA underflow interrupt enable</p> <p>This bit is set and cleared by software.</p>

				0: Do not enable DAC channel 1 DMA underflow interrupt 1: Enable DAC channel 1 DMA underflow interrupt
12	DMAEN1	RW	0	DAC channel1 DMA enable (DAC channel1 DMA enable) This bit is set and cleared by software. 0: Disable DAC channel 1 DMA mode 1: Enable DAC channel 1 DMA mode
11: 8	MAMP1[3: 0]	RW	0	DAC channel1 mask/amplitude selector These bits are set by software and are used to select the mask bit in noise generation mode and the increase value of the waveform in triangle wave generation mode. 0000: not masked LSFR bit 0/triangle wave amplitude equal to 1 0001: not masked LSFR bits [1:0] / triangle wave amplitude equal to 3 0010: not masked LSFR bits [2:0] / delta wave amplitude equal to 7 0011: not masked LSFR bits [3:0] / delta wave amplitude equal to 15 0100: No masking of LSFR bits [4:0] / Triangle wave amplitude equal to 31 0101: not masked LSFR bits [5:0] / triangle wave amplitude equal to 63 0110: No masking of LSFR bits [6:0] / Triangular wave amplitude equal to 127 0111: No masking of LSFR bits [7:0] / Triangle wave amplitude equal to 255 1000: Unmasked LSFR bits [8:0] / Triangle wave amplitude equal to 511 1001: no masking of LSFR bits [9:0] / triangle wave amplitude equal to 1023 1010: No masking of LSFR bits [10:0] / Triangle wave amplitude equal to 2047 ≥1011: not masked LSFR bits [11:0]/triangular wave amplitude equal to 4095
7: 6	WAVE1[1: 0]	RW	0	DAC channel1 noise/triangle wave generation enable (DAC channel1 noise/triangle wave generation enable) This 2-bit is set and cleared by software.

				00: Turns off the waveform generator; 01: Enable the noise waveform generator; 1x: Enable the triangle wave generator.
5: 3	TSEL1[2: 0]	RW	0	DAC channel1 trigger selection (DAC channel1 trigger selection) This 3-bit is used to select the external trigger event for DAC channel 1. 000: TIM6 TRGO incident 001: TIM3 TRGO incident 010: TIM7 TRGO incident 011: TIM15 TRGO incident 100: TIM2 TRGO incident 101: Reserved 110: External interrupt line 9 111: Software triggers Note: This 3-bit can only be used when TEN1 = 1 (DAC channel 1 trigger enable).
2	TEN1	RW	0	DAC channel1 trigger enable (DAC channel1 trigger enable) This bit is set and cleared by software to enable/disable the triggering of DAC channel 1. 0: Turn off DAC channel 1 triggering, data written to the DAC_DHRx register is transferred to the DAC_DOR1 register after 1 APB clock cycle; 1: Enable DAC channel 1 trigger, data written to DAC_DHRx register is transferred to DAC_DOR1 register after 3 APB clock cycles. Note: If software triggering is selected, data written to register DAC_DHRx requires only one APB always cycle to pass into register DAC_DOR1.
1	BOFF1	RW	0	DAC channel1 output buffer disable This bit is set and cleared by software to enable/disable the output buffer for DAC channel 1. 0: Enable DAC channel 1 output buffer 1: Disable DAC channel 1 output buffering
0	EN1	RW	0	DAC channel1 enable (DAC channel1 enable) This bit is set and cleared by software to enable/disable DAC channel 1. 0: Close DAC channel 1 1: Enable DAC channel 1

15.4.2. DAC Software Trigger Register (DAC_SWTRIGR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	SWTRIG2	SWTRIG1
-														RW	RW

Bit	Name	R/W	Reset Value	Function
31: 2	Res	-	-	Res
1	SWTRIG2	RW	0	<p>DAC channel2 software trigger (DAC channel2 software trigger)</p> <p>This bit is set and cleared by software to enable/disable software triggering.</p> <p>0: Disable DAC channel 2 software trigger</p> <p>1: Enable DAC channel 2 software trigger</p> <p>Note: Once the data from register DAC_DHR2 is passed into register DAC_DOR2, (after 1 APB clock cycle) this bit is set by hardware to '0'</p>
0	SWTRIG1	RW	0	<p>DAC channel1 software trigger (DAC channel1 software trigger)</p> <p>This bit is set and cleared by software to enable/disable software triggering.</p> <p>0: Disables DAC channel 1 software triggering;</p> <p>1: Enable DAC channel 1 software trigger.</p> <p>Note: Once the data from register DAC_DHR1 is passed into register DAC_DOR1, (after 1 APB clock cycle) this bit is set by hardware to '0'</p>

15.4.3. 12-bit right-aligned data hold register for DAC channel 1

(DAC_DHR12R1)

Address offset: 0x08

Reset value: 0x0000 0000

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

-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	DACC1DHR[11: 0]											
-				RW											

Bit	Name	R/W	Reset Value	Function
31: 12	Res	-	-	Res
11: 0	DACC1DHR[11: 0]	RW	0	DAC channel1 12-bit right-aligned data This bit is written by software and indicates the 12-bit data for DAC channel 1

15.4.4. 12-bit left-aligned data hold register for DAC channel 1 (DAC_DHR12L1)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC1DHR[11: 0]												Res	Res	Res	Res
RW												-			

Bit	Name	R/W	Reset Value	Function
31: 16	Res	-	-	Res
15: 4	DACC1DHR[11: 0]	RW	0	DAC channel1 12-bit left-aligned data This bit is written by software and indicates the 12-bit data for DAC channel 1.
3: 0	Res	-	-	Res

15.4.5. 8-bit right-aligned data hold register for DAC channel 1 (DAC_DHR8R1)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	DACC1DHR[7: 0]							
-								RW							

Bit	Name	R/W	Reset Value	Function
31: 8	Res	-	-	Res
7: 0	DACC1DHR[7: 0]	RW	0	DAC channel1 8-bit right-aligned data This bit is written by software and indicates the 8-bit data for DAC channel 1.

15.4.6. 12-bit right-aligned data hold register for DAC channel 2

(DAC_DHR12R2)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	DACC2DHR[11: 0]											
-				RW											

Bit	Name	R/W	Reset Value	Function
31: 12	Res	-	-	Res
11: 0	DACC2DHR[11: 0]	RW	0	DAC channel2 12-bit right-aligned data This bit is written by software and indicates the 12-bit data for DAC channel 2.

15.4.7. 12-bit left-aligned data hold register for DAC channel 2 (DAC_DHR12L2)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHR[11: 0]												Res	Res	Res	Res
RW												-			

Bit	Name	R/W	Reset Value	Function
31: 16	Res	-	-	Res
15: 4	DACC2DHR[11: 0]	RW	0	DAC channel2 12-bit left-aligned data This bit is written by software and indicates the 12-bit data for DAC channel 2.

3: 0	Res	-	-	Res
------	-----	---	---	-----

15.4.8. 8-bit right-aligned data hold register for DAC channel 2 (DAC_DHR8R2)

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	DACC2DHR[7: 0]							
-								RW							

Bit	Name	R/W	Reset Value	Function
31: 8	Res	-	-	Res
7: 0	DACC2DHR[7: 0]	RW	0	DAC channel2 8-bit right-aligned data This bit is written by software and indicates the 8-bit data for DAC channel 2.

15.4.9. 12-bit right-aligned data hold register for dual DACs (DAC_DHR12RD)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	DACC2DHR[11: 0]											
-				RW											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	DACC1DHR[11: 0]											
-				RW											

Bit	Name	R/W	Reset Value	Function
31: 28	Res	-	-	Res
27: 16	DACC2DHR[11: 0]	RW	0	DAC channel2 12-bit right-aligned data This bit is written by software and indicates the 12-bit data for DAC channel 2.
15: 12	Res	-	-	Res
11: 0	DACC1DHR[11: 0]	RW	0	DAC channel1 12-bit right-aligned data This bit is written by software and indicates the 12-bit data for DAC channel 1.

15.4.10. 12-bit left-aligned data hold register for dual DACs

(DAC_DHR12LD)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DACC2DHR[11: 0]												Res	Res	Res	Res
RW												-			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC1DHR[11: 0]												Res	Res	Res	Res
RW												-			

Bit	Name	R/W	Reset Value	Function
31: 20	DACC2DHR[11: 0]	RW	0	DAC channel2 12-bit left-aligned data This bit is written by software and indicates the 12-bit data for DAC channel 2.
19: 16	Res	-	-	Res
15: 4	DACC1DHR[11: 0]	RW	0	DAC channel1 12-bit left-aligned data This bit is written by software and indicates the 12-bit data for DAC channel 1.
3: 0	Res	-	-	Res

15.4.11. 8-bit right-aligned data hold register for dual DACs

(DAC_DHR8RD)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHR[7: 0]								DACC1DHR[7: 0]							
RW								RW							

Bit	Name	R/W	Reset Value	Function
31: 16	Res	-	-	Res
15: 8	DACC2DHR[7: 0]	RW	0	DAC channel2 8-bit right-aligned data

				This bit is written by software and indicates the 8-bit data for DAC channel 2.
7: 0	DACC1DHR[7: 0]	RW	0	DAC channel1 8-bit right-aligned data This bit is written by software and indicates the 8-bit data for DAC channel 1.

15.4.12. DAC channel 1 data output register (DAC_DOR1)

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	DACC1DOR[11: 0]											
-				R											

Bit	Name	R/W	Reset Value	Function
31: 12	Res	-	-	Res
11: 0	DACC1DOR[11: 0]	R	0	DAC channel1 data output This bit is read-only and indicates the output data for DAC channel 1.

15.4.13. DAC channel 2 data output register (DAC_DOR2)

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	DACC2DOR[11: 0]											
-				R											

Bit	Name	R/W	Reset Value	Function
31: 12	Res	-	-	Res
11: 0	DACC2DOR[11: 0]	R	0	DAC channel2 data output This bit is read-only and indicates the output data for DAC channel 2.

15.4.14. DAC Status Register (DAC_SR)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	DMAUDR2	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-		RC_W1	-												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	DMAUDR1	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-		RC_W1	-												

Bit	Name	R/W	Reset Value	Function
31: 30	Res	-	-	Res
29	DMAUDR2	RC_W1	0	<p>DAC channel 2 DMA underrun flag</p> <p>This bit is set to 1 by hardware and cleared to 0 by a software write 1</p> <p>0: No DMA underflow condition has occurred on DAC channel 2;</p> <p>1: A DMA underflow condition has occurred on DAC channel 2.</p> <p>Note: The software can write "1" and clear "0" only after DMAUDR2 is set to "1".</p>
28: 14	Res	-	-	Res
13	DMAUDR1	RC_W1	0	<p>DAC channel 1 DMA underrun flag</p> <p>This bit is set to 1 by hardware and cleared to 0 by a software write 1</p> <p>0: No DMA underflow condition has occurred on DAC channel 1;</p> <p>1: A DMA underflow condition has occurred on DAC channel 1.</p> <p>Note: The software can write "1" and clear "0" only after DMAUDR1 is set to "1".</p>
12: 0	Res	-	-	Res

16. Analog/Digital Conversion (ADC)

16.1. Introduction

A 12-bit ADC is a successive approximation analog-to-digital converter. It has up to 24 channels to measure 16 external and 8 internal signal sources. A/D conversion for each channel can be performed in single, continuous, scanning or intermittent mode. The ADC results can be stored left- or right-aligned in a 16-bit data register.

The analog watchdog feature allows the application to detect if the input voltage exceeds a user-defined high/low threshold.

16.2. ADC Main Characteristics

- High performance
 - 12-bit, 10-bit, 8-bit and 6-bit resolutions are configurable
 - ADC conversion time: 1 μ s@12-bit (1 Msps)
 - self-calibration
 - Programmable sampling time
 - Programmable data alignment modes
 - Rule group support for DMA
- Analog Input Channels
 - 16 external analog input channels
 - 1 internal temperature sensor channel (TSensor)
 - 1 internal reference voltage channel (VREFINT)
 - 1 internal reference voltage input channel (VREFBUF)
 - 3 internal OPA input voltage channels
 - 2 internal DAC input voltage channels
- Conversion operation startup can be performed with the
 - software launch
 - Hardware startup (TIM1, TIM2, TIM3, TIM15 or GPIO)
- conversion mode
 - Single pass mode: 1 single pass can be converted
 - Scanning mode: can scan a range of channels
 - Continuous Mode: Continuous switching of the selected channel
 - Intermittent mode: each trigger converts a subsequence channel, multiple triggers until the complete sequence has been converted.
- disruption generation
 - At the end of the conversion
 - Simulating Watchdog Events
- Analog Watchdog

16.3. ADC Functional Description

16.3.1. ADC Block Diagram

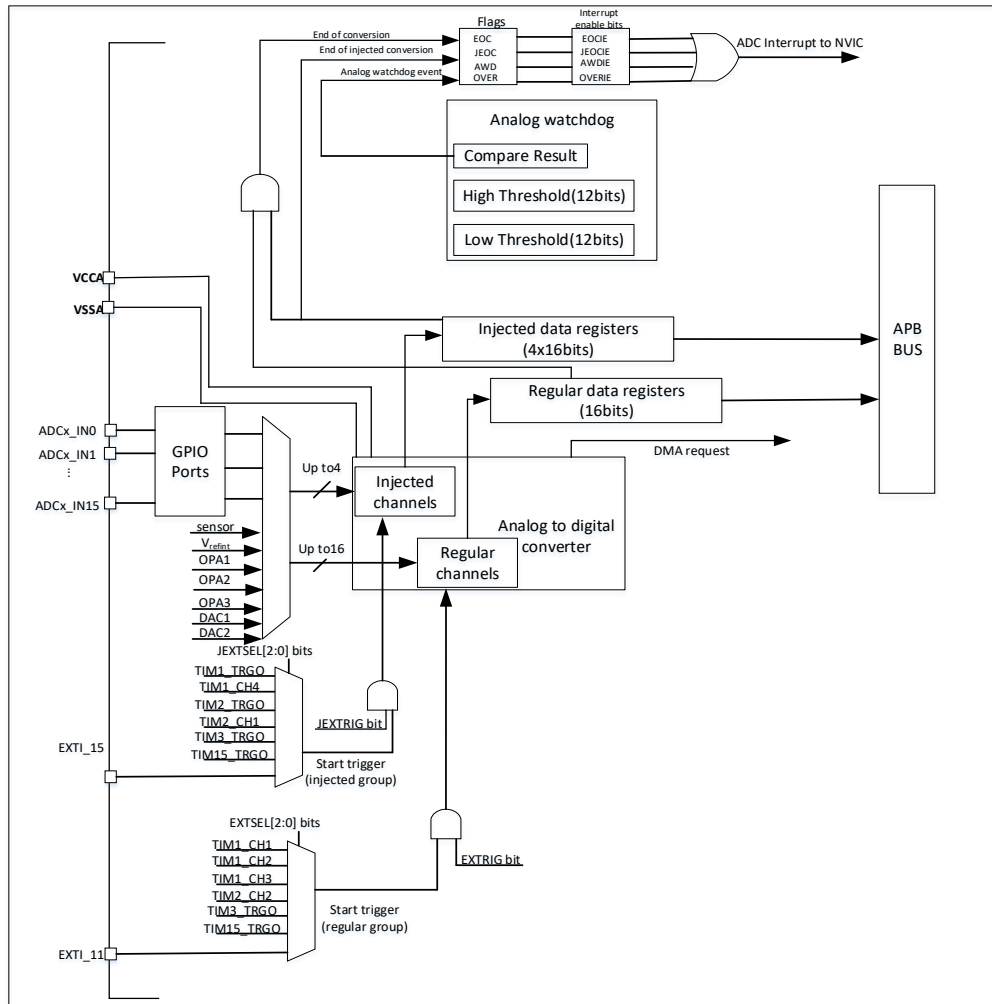


Figure 16-1 ADC Block Diagram

16.3.2. Calibrations

This ADC has a software calibration function. During calibration, the ADC calculates a calibration factor for use within the ADC (lost when the ADC is powered down). The application cannot use the ADC module during ADC calibration, before calibration is completed.

Perform a calibration operation before using the ADC conversion. Calibration is used to eliminate chip-to-chip offset errors due to process variations.

Software setting `ADC_CR2.CAL=1` initiates calibration, which can only be initiated when the ADC is not enabled (`ADC_CR2.ADON=0`) and only supports selection of the system clock as the ADC's clock. When calibration is complete, `CAL` is cleared to 0 by hardware.

When the operating conditions of the ADC change (VCCA change is the main factor for ADC offset offset, temperature change is the second), a recalibration operation is recommended.

Software procedure for calibration:

- Confirmation of ADON=0
- Set CAL=1
- Wait for CAL=0

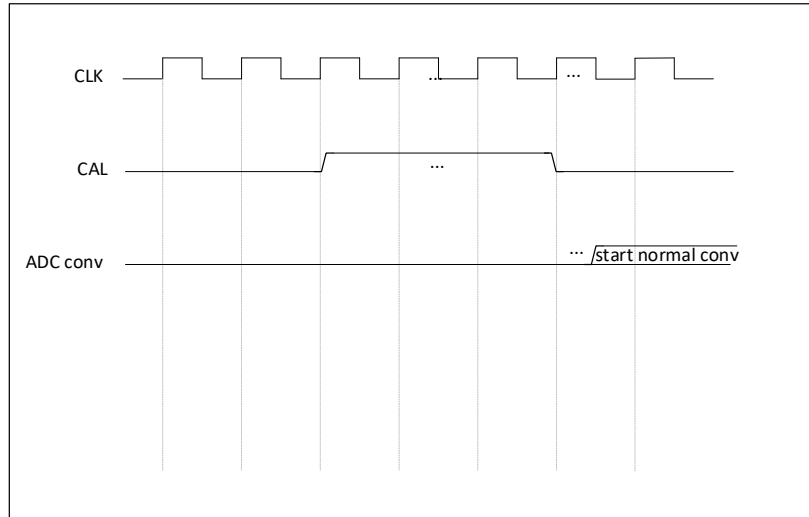


Figure 16-2 ADC calibration timing diagram

16.3.3. ADC Switch Control

The ADC can be powered up by setting the ADON bit in the ADC_CR2 register. When the ADON bit is set for the first time, it wakes up the ADC from a power-down state.

The ADC power-up delayed for a period of time (t_{STAB} , not less than 1 μs) to start the conversion.

To save power, the ADC analog submodule will enter power-down mode when ADON is 0. Conversion can be stopped by clearing the ADON bit and placing the ADC in power-down mode.

16.3.4. ADC Clock

The ADCCLK clock provided by the RCC control is synchronized with PCLK (APB clock). The RCC controller (CLK controller) provides a dedicated programmable prescaler for the ADC clock, the ADCCLK clock division is detailed in RCC_CR.ADC_DIV[22:21].

16.3.5. Channel Selection

There are 16 external channels and 8 internal channels, of which the internal channels have:

- Temperature Sensor/VREFINT Internal Channel

The temperature sensor is connected to channel ADC_IN23 and the internal reference voltage VREFINT is connected to ADC_IN17.

- VCCA/3

VCCA/3 is connected to channel ADC_IN18.

- DAC

DAC1_VIN is connected to channel ADC_IN19, and DAC2_VIN is connected to channel ADC_IN20.

■ OPA

OPA1_VIN is connected to channel ADC_IN21, OPA2_VIN is connected to channel ADC_IN22, and OPA3_VIN is connected to channel ADC_IN16. (Note: Use channel 16 to ensure that SMP16 and SMP0 are set to the same value)

Transformations can be organized into two groups: rule groups and injection groups. A series of conversions performed in any order on any plurality of channels constitutes a grouped conversion. For example, the conversion can be accomplished in the following order: channel 3, channel 8, channel 2, channel 2, channel 2, channel 0, channel 2, channel 2, channel 15.

The ruleset consists of up to 16 transitions. The rule channels and their conversion order are selected in the ADC_SQRx register. The total number of conversions in the rule group should be written to the L[3:0] bits of the ADC_SQR1 register.

The injection group consists of up to 4 transitions. The injection channels and their conversion order are selected in the ADC_JSQR register. The total number of conversions in the injection group shall be written to the JL[1:0] bits of the ADC_JSQR register.

If the ADC_SQRx or ADC_JSQR registers are changed during a conversion, the current conversion is cleared and a new start pulse is sent to the ADC to convert the newly selected group.

16.3.6. Programmable sampling time

The ADC samples the input voltage using a number of ADC_CLK cycles, and the number of sampling cycles can be changed by the SMP[2:0] bits in the ADC_SMPR1, ADC_SMPR2, and ADC_SMPR3 registers. Each channel can be sampled separately with different times.

The total conversion time is calculated as follows:

$$t_{\text{CONV}} = \text{sampling time} + 12.5 \text{ cycles (RESSEL=00B)}$$

Example:

When $f_{\text{ADC}} = 16 \text{ MHz}$, the sampling time is 3.5 cycles

$$t_{\text{CONV}} = 3.5 + 12.5 = 16 \text{ cycles} = 1 \mu\text{s}$$

16.3.7. Configurable resolution

Fast conversions can be performed by reducing the ADC resolution. The RESSEL bit of the ADC_CR1 register is used to select the number of bits available in the data register. The minimum conversion times for each resolution are as follows:

- 12-bit: $3.5 + 12.5 = 16 \text{ ADCCLK cycles}$
- 10 bits: $3.5 + 10.5 = 14 \text{ ADCCLK cycles}$
- 8-bit: $3.5 + 8.5 = 12 \text{ ADCCLK cycles}$
- 6-bit: $3.5 + 6.5 = 10 \text{ ADCCLK cycles}$

16.3.8. Single conversion mode

In single conversion mode, the ADC performs only one conversion. After enabling EX-TTRIG/JEXTTRIG, an external event (e.g., timer capture, EXTI interrupt, software trigger) triggers the initiation of a conversion (for either the rule channel or the injected channel), at which point the CONT bit is 0.

Once the conversion of the selected channel is complete:

- If a rule channel is converted:
 - Conversion data is stored in the 16-bit ADC_DR register
 - The EOC (end of conversion) flag is set
 - If EOCIE is set, an interrupt is generated.
- If an injection channel is converted:
 - The conversion data is stored in the 16-bit ADC_JDRx register
 - JEOC (end of injection conversion) flag is set
 - If the JEOCIE bit is set, an interrupt is generated.

Then the ADC stops.

16.3.9. Continuous Conversion Mode

In continuous conversion mode, another conversion is initiated as soon as the previous ADC conversion is completed. After enabling EXTTRIG/JEXTTRIG, an external event (e.g., timer capture, EXTI interrupt, software trigger) triggers the initiation of a conversion (for regular channels; injected channels behave in single-conversion mode), at which time the CONT bit is one.

After each conversion:

- If a rule channel is converted:
 - Conversion data is stored in the 16-bit ADC_DR register
 - The EOC (end of conversion) flag is set
 - If EOCIE is set, an interrupt is generated.
- If an injection channel is converted:
 - EOC (end-of-conversion flag) flag set
 - The conversion data is stored in the 16-bit ADC_JDRx register
 - JEOC (end of injection conversion) flag is set
 - If the JEOCIE bit is set, an interrupt is generated.

16.3.10. Scanning mode

This mode is used to scan a set of analog channels.

The scan mode can be selected by setting the SCAN bit in the ADC_CR1 register. Once this bit is set, the ADC scans all channels that are selected by the ADC_SQRx register (for rule channels) or ADC_JSQR (for injection channels). Performs a single conversion on each channel of each group. At the end of each transition, the next channel in the same group is automatically converted. If the

CONT bit is set, the conversion does not stop on the last channel of the selection group, but continues again from the first channel of the selection group.

If the DMA bit is set, the DMA controller transfers the conversion data of the rule group channel into SRAM. And the data injected into the channel conversion is always stored in the ADC_JDRx register.

16.3.11. intermittent switching mode

16.3.11.1. Rules group

This mode is activated by setting the DISCEN bit on the ADC_CR1 register. It can be used to perform a short sequence of n conversions ($n \leq 8$) that are part of the conversion sequence selected by the ADC_SQRx register. The value n is given by the DISCNUM[2:0] bits of the ADC_CR1 register.

An external trigger signal can initiate the next round of n conversions described in the ADC_SQRx register until all conversions in this sequence are complete. The total sequence length is defined by L[3:0] of the ADC_SQR1 register.

Examples:

$n=3$, channels converted = 0, 1, 2, 3, 6, 7, 9, 10

First trigger: the sequence of conversions is 0, 1, 2

Second Trigger: Converted Sequence of 3, 6, 7

Third trigger: the converted sequence is 9, 10 and generates an EOC event

Fourth trigger: sequence of conversions 0, 1, 2

Note: When converting a rule group in interrupted mode, the conversion sequence does not automatically start from the beginning when it ends.

When all subsequence groups have been converted, the next trigger initiates the conversion of the first subsequence group. In the above example, the fourth trigger reconverts channels 0, 1 and 2 of the first subsequence group.

16.3.11.2. Injection group

This mode is activated by setting the JDISCEN bit in the ADC_CR1 register. After an external trigger event, this mode converts the sequence selected in the ADC_JSQR register one by one in channel order.

An external trigger signal can initiate the conversion of the next channel sequence selected by the ADC_JSQR register until all the channels in the sequence are converted until the conversion is complete. The total sequence length is defined by the JL[1:0] bits of the ADC_JSQR register.

Example:

$n = 1$, channel being converted = 1, 2, 3

First trigger: Channel 1 is converted

Second trigger: Channel 2 is converted

Third trigger: channel 3 is converted and EOC and JEOC events are generated

Fourth trigger: Channel 1 is converted

Note: When all injection channel transitions are completed, the next trigger initiates the transition of the 1st injection channel. In the above example, the fourth trigger reconverts the 1st injection channel 1.

It is not possible to use automatic injection and intermittent mode at the same time.

16.3.12. Injection channel management

The external trigger of the injected channel has a higher priority than the external trigger of the rule channel, i.e., the external trigger of the injected channel can interrupt a rule channel transition in progress. There are two ways to inject channels: triggered injection and automatic injection.

16.3.12.1. Trigger injection

In order to use trigger injection, the JAUTO bit of the ADC_CR1 register must be cleared and the SCAN bit set.

- By setting the ADON bit in the ADC_CR2 register, SWSTART is set and an external trigger initiates the conversion of a set of rule channels.
- If an external injection trigger is generated during a rule channel transition, the current transition is reset and the sequence of injected channels is transitioned in a single scan.
- Then, the last interrupted rule group channel transition is resumed. If a rule event is generated during an injection transition, the injection transition will not be interrupted, but the rule sequence will not be executed at the end of the injection sequence.
- Intermittent mode does not support triggered injection. Rule triggers during trigger injection are unresponsive.

Note: When using triggered injection transformations, it must be ensured that the interval between triggering events is longer than the injection sequence. For example, if the sequence length is 28 ADC clock cycles (i.e., 2 transitions with 1.5 clock interval sample times), the minimum interval between triggers must be 29 ADC clock cycles.

16.3.12.2. Auto-injection

If the JAUTO bit is set, the injection group channel is automatically converted after the rule group channel. This can be used to convert up to 20 channels set in the ADC_SQRx and ADC_JSQR registers.

In this mode, external triggering of the injection channel must be disabled.

If the CONT bit is set in addition to the JAUTO bit, the conversion sequence from the rule channel to the injection channel is executed consecutively.

Note: It is forbidden to use automatic injection and intermittent mode at the same time.

16.3.13. Stopping conversions in progress (ADSTP)

Setting ADSTP=1 in the ADC_CR1 register in software stops the currently running conversion, re-sets the ADC operation and puts the ADC into an idle state, ready for the next conversion.

When ADSTP is set to 1 by software, any current conversion is aborted and the conversion result is discarded (ADC_DR register is not updated with the current conversion value).

The scan sequence is also aborted and reset (i.e., the ADC is restarted with a new sequence for conversion)

Once the process is completed both the ADSTP and SWSTART bits are cleared to 0 by hardware.

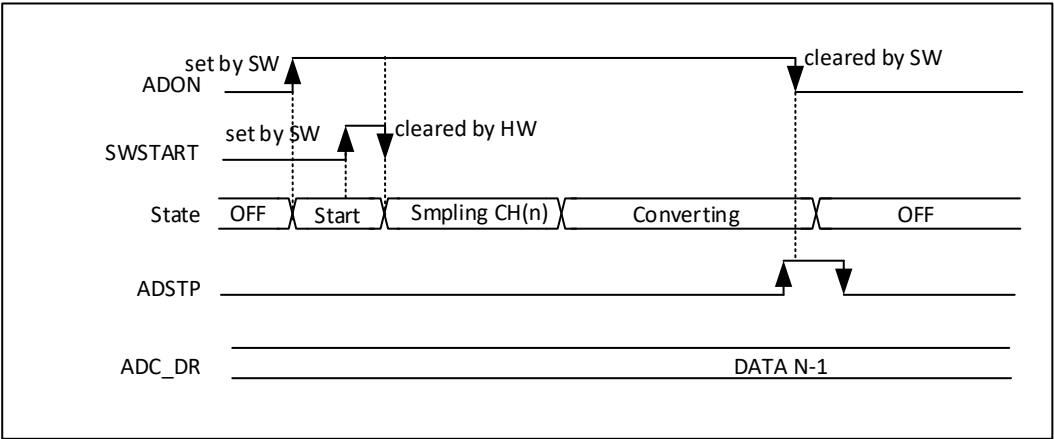


Figure 16-3 Stop timing

16.4. Analog Watchdog

16.4.1.1. The AWD analog watchdog status bit is set to 1 if the analog voltage converted by the ADC is below the lower threshold or above the upper threshold. These thresholds are set in the 12 least significant bits of the ADC_HTR and ADC_LTR registers. Interrupts are generated by setting the AWDIE bit in the ADC_CR1 register.

The threshold is independent of the selected alignment of the ALIGN bit in the ADC_CR2 register. Threshold comparison is done before alignment (before injecting the channel minus the offset value). By configuring the ADC_CR1 register, the analog watchdog can act on 1 or more channels as shown in the table below:

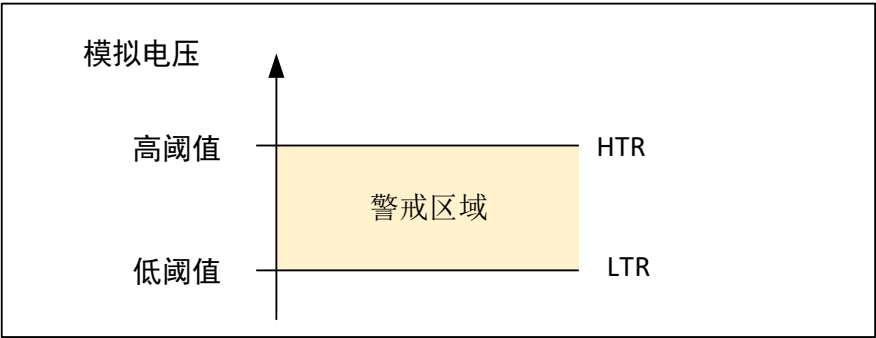


Figure 16-4 Analog Watchdog Protection Area

Table 16-1 Analog Watchdog Channel Selection

Analog Watchdog Protection Channel	ADC_CR1 register control bits		
	AWDSGL	AWDEN	JAWDEN
not have	X	0	0
All injection channels	0	0	1
All rules channel	0	1	0
All injection and rule channels	0	1	1
single injection channel	1	0	1
single-rule channel	1	1	0
Single injection or rule channel	1	1	1

16.5. Externally Triggered Conversion

Conversions can be triggered by external events (e.g. timer capture, EXTI interrupt). If the EXTTRIG or JEXTTRIG control bit is set, an external event can trigger the conversion. The EXTSEL[2:0] and JEXTSEL[2:0] control bits allow the application to select one of eight possible events that can trigger the sampling of rules and injection groups.

Note: When an external trigger signal is selected for ADC rule or injection conversion, only the rising edge can initiate the conversion.

The following table gives the possible external triggers for the conversion. Software source trigger events can be generated by setting the ADSTART bit in the ADC_CR register.

Table 16-2 ADC External Trigger for Rule Channel

trigger source	typology	EXTSEL [2:0]
CH1 output of timer 1	Internal signals for on-chip timer	000
CH2 output of timer 1		001
CH3 output of timer 1		010
TRGO output of timer 2		011
TRGO output of timer 3		100
TRGO output of timer 15		101
EXTI Line 11	External Pins	110
SWSTART	software control bits	111

Table 16-3 ADC External Trigger for Injection Channel

trigger source	typology	JEXTSEL [2:0]
TRGO output of timer 1	Internal signals for on-chip timer	000
CH4 output of timer 1		001
TRGO output of timer 2		010
CH1 output of timer 2		011
CH4 output of timer 3		100

TRGO output of timer 15		101
EXTI Line 15 Output	External Pins	110
JSWSTART	software control bits	111

16.6. Data alignment

At the end of each conversion, the conversion result data is stored in the 16-bit wide ADC_DR data register. The ALIGN bit in the ADC_CR2 register selects the alignment of the converted data storage. The data can be left- or right-aligned, e.g., the data value converted by the injection group channel has been subtracted from the offset defined in the ADC_JOFRx register, so the result can be a negative value. The SEXT bit is the sign value of the extension.

For rule group channels, there is no need to subtract the offset value, so only 12 bits are valid.

Table 16-4 Right-aligned data

injection group															
SEXT	SEXT	SEXT	SEXT	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
rules group															
0	0	0	0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Table 16-5 Left-aligned data

injection group															
SEXT	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0
rules group															
D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0	0

16.7. Data overload

The ADC Overload Flag (OVER) is a buffer overload event that occurs when another conversion data has been validated by the rule group when the converted data has not been read by the CPU or DMA in time.

16.8. DMA request

Because the value of a rule channel conversion is stored in a data-only register, DMA is required when converting multiple rule channels, which prevents the loss of data already stored in the ADC_DR register.

A DMA request is generated only at the end of the conversion of the rule channel and the converted data is transferred from the ADC_DR register to the user-specified destination address.

16.9. Temperature sensor and internal reference voltage

The temperature sensor can be used to measure the temperature (TA) around the device. The temperature sensor is internally connected to the ADC_IN23 channel, which converts the voltage output from the sensor into a digital value. The recommended sampling time for the temperature sensor analog input is 17.1us. When not in use, the sensor can be placed in power-down mode.

The output voltage of the temperature sensor varies linearly with temperature. Due to variations in the production process, the offset of the temperature variation curve can vary from chip to chip (up to 45 °C difference). Internal temperature sensors are better suited to detecting changes in temperature rather than measuring absolute temperatures. If accurate temperature measurements are required, an external temperature sensor should be used.

Note: The TSVREFE bit must be set to activate the conversion of the internal channels: ADC_IN23 (temperature sensor) and ADC_IN17 (VREFINT).

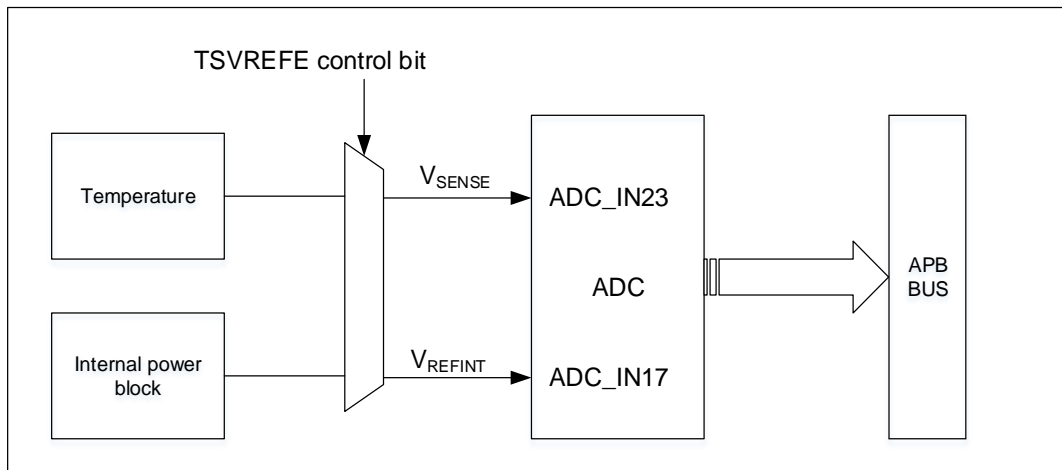


Figure 16-5 Temperature Sensor Channel Block Diagram

temperature reading

To use the sensor, do the following:

1. Select ADC_IN23
2. Select a sampling time that is greater than the minimum sampling time specified in the datasheet (SMP23).
3. Set TSVREFE position 1 in the ADC_CR2 register to wake up the temperature sensor from power-down mode.
4. Start ADC conversion by setting ADON to position 1 and enabling external trigger
5. Read the VSENSE data generated in the ADC data register.
6. Calculate the temperature using the following formula:

$$\text{Temperature (}^{\circ}\text{C)} = \{(\text{VSENSE} - \text{V30}) / \text{Avg_Slope}\} + 30^{\circ}\text{C}$$

Among them:

VSENSE value at V30 = 30 °C

Avg_Slope = average slope of the temperature vs. VSENSE curve in mV/°C or μV/°C

(See the Electrical Characteristics section of the datasheet for information on the actual values of V30 and Avg_Slope.)

NOTE: The sensor requires a build-up time to wake up from power-down mode and outputs the correct level of VSENSE after the start-up time has elapsed. The ADC also requires a build-up time after power-up, so to shorten the delay, ADON and TSVREFE should be set at the same time.

16.10. ADC Interrupt

Interrupts can be generated at the end of rule and injection group conversions, when the analog watchdog status bit is set, and when rule group conversion data is not read in time. They all have independent interrupt enable bits.

There are 2 other flags in the ADC_SR register, but they have no interrupts associated with them:

- JSTRT (initiation of injection group channel transformation)
- STRT (initiation of rule group channel transition)

Table 16-6 ADC Interrupts

Disruption event	Event marker	Enable Control
End of rule group conversion	EOC	EOCIE
End of injection group conversion	JEOC	JEOCIE
Analog watchdog status bits set	AWD	AWDIE
Overflow marker	OVER	OVERIE

16.11. ADC Registers

16.11.1. ADC Status Register (ADC_SR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res										OVER	STRT	JSTRT	JEOC	EOC	AWD
-										RC	RC_W0	RC_W0	RC_W0	RC_W0	RC_W0

Bit	Name	R/W	Reset Value	Function
31: 6	Res	-	-	Res
5	OVER	RC	0	<p>ADC overload</p> <p>Hardware sets this bit when an overload occurs. When the EOC flag has been set it indicates that a new conversion has been completed.</p> <p>0: No overload occurs (DMA or CPU has read the last conversion result)</p> <p>1: Overload has occurred</p>
4	STRT	RC_W0	0	<p>Rule Channel Start Status Bit</p> <p>This bit is set by hardware at the start of a rule channel transition and cleared by software.</p> <p>0: Rule channel conversion not started</p>

Bit	Name	R/W	Reset Value	Function
				1: Rule channel conversion has begun
3	JSTRT	RC_W0	0	Inject Channel Start Status Bit This bit is set by hardware at the start of the injected channel group conversion and cleared by software. 0: Injection channel conversion not started 1: Injection channel conversion has begun
2	JEOC	RC_W0	0	Inject Channel Transition End Status Bit This bit is set by hardware at the end of all injected channel group conversions and cleared by software. 0: Conversion not completed 1: Conversion completed
1	EOC	RC_W0	0	End-of-conversion status bit This bit is set by hardware at the end of a (regular or injected) channel group conversion, cleared by software, or cleared by reading ADC_DR. 0: Conversion not completed 1: Conversion completed
0	AWD	RC_W0	0	Analog Watchdog Flag Bit This bit is set by hardware to 1 when the converted voltage value is outside the range defined by the ADC_LTR or below the ADC_HTR register, and is cleared by software. 0: No analog watchdog event occurred 1: An analog watchdog event occurs

16.11.2. ADC Control Register 1 (ADC_CR1)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res		OVRIE	Res	AD-STP	Res	RESSEL [1:0]		AWD EN	JAWD EN	Res					
-		RW	-	R_W1	-	RW		RW	RW	-					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DISCNUM [2:0]			JDIS CEN	DIS-CEN	JAU TO	AWD SGL	SCA N	JEO CIE	AWDIE	EOCIE	AWDCH [4:0]				
RW			RW	RW	RW	RW	RW	RW	RW	RW	RW				

Bit	Name	R/W	Reset Value	Function
31: 30	Res	-	-	Res

Bit	Name	R/W	Reset Value	Function
29	OVRIE	RW	0	OVER FLAG interrupt enable 0: Disable overload interrupt 1: Overload interruptions allowed
28	Res	-	-	Res
27	ADSTP	R_W1	0	ADC conversion stop enable. Software Write 1 Set 1. Hardware set to 0 when the ADC stop signal is received. 1: Stop ADC conversion 0: Do not stop ADC conversion
26	Res	-	-	Res
25: 24	RESSEL [1:0]	RW	0	Resolution control bit. The resolution of the conversion can be selected by writing these bits in software. 00: 12 bits (16 ADCCLK cycles) 01: 10 bits (14 ADCCLK cycles) 10: 8 bits (12 ADCCLK cycles) 11: 6 bits (10 ADCCLK cycles)
23	AWDEN	RW	0	Rule channel analog watchdog enable. Turn on the analog watchdog on the rule channel. This bit is set by the software Write 1 to set 1 and Write 0 to set 0. 0: Disable analog watchdog on rule channel 1: Using an analog watchdog on a rule channel
22	JAWDEN	RW	0	Inject channel analog watchdog enable. Turn on the analog watchdog on the injection channel. This bit is set by the software Write 1 to set 1 and Write 0 to set 0. 0: Disable analog watchdog on injection channel 1: Using an analog watchdog on the injection channel
21: 16	Res	-	-	Res
15: 13	DISCNUM [2:0]	RW	0	Interrupt mode channel count. In intermittent mode, the number of channels converted by the rule channel group after receiving an external trigger. A software write operation sets these bits. 000: 1 channel 001: 2 channels

Bit	Name	R/W	Reset Value	Function
				111: 8 channels
12	JDISCEN	RW	0	<p>Inject channel interrupt mode enable.</p> <p>This bit is set by a software write 1 to 1 and a write 0 to 0 to enable or disable intermittent mode on the injected channel set.</p> <p>0: Disable intermittent mode on the injector channel group</p> <p>1: Use intermittent mode on injection channel groups</p>
11	DISCEN	RW	0	<p>Intermittent mode enable for rule channel</p> <p>This bit is set by a software write 1 to 1 and a write 0 to 0 to enable or disable intermittent mode on the rule channel set</p> <p>0: Intermittent mode disabled on rule channel groups</p> <p>1: Use intermittent mode on rule channel groups</p>
10	JAUTO	RW	0	<p>Auto Inject Enable</p> <p>This bit is set to 1 by software write 1 and 0 by software write 0 to enable or disable automatic injection of channel group transitions at the end of a rule channel group transition</p> <p>0: Disable automatic injection channel group switching</p> <p>1: Enable automatic injection channel group switching</p>
9	AWDSGL	RW	0	<p>Single channel watchdog enable.</p> <p>This bit is set by a software write 1 to 1 and a write 0 to 0 to turn on or off the analog watchdog on the channel defined by AWDCH[4:0].</p> <p>0: use analog watchdog on all channels</p> <p>1: Using an analog watchdog on a single channel</p>
8	SCAN	RW	0	<p>Scan Mode Enable</p> <p>This bit is set by a software write 1 to 1 and a write 0 to 0 to turn scan mode on or off. In scan mode, the channel selected by the ADC_SQRx or ADC_JSQRx register is converted.</p> <p>0: Turn off scanning mode</p> <p>1: Use scanning mode</p>

Bit	Name	R/W	Reset Value	Function
7	JEOCIE	RW	0	<p>Inject channel end-of-conversion interrupt enable</p> <p>This bit is set by the software Write 1 to set 1 and Write 0 to set 0. When this bit is enabled, an interrupt request is generated while JEOC is active.</p> <p>0: JEOC interrupt disabled 1: JEOC interrupts are allowed.</p>
6	AWDIE	RW	0	<p>Analog Watchdog Interrupt Enable</p> <p>This bit is set by the software Write 1 to set 1 and Write 0 to set 0. When this bit is enabled, an interrupt request is generated while AWD is active.</p> <p>0: Disable analog watchdog interrupt 1: Allow analog watchdog interrupt</p>
5	EOCIE	RW	0	<p>Rule channel transition end interrupt enable</p> <p>This bit is set by the software Write 1 to set 1 and Write 0 to set 0. When this bit is enabled, an interrupt request is generated while the EOC is active.</p> <p>0: Disable EOC interrupt 1: Allow EOC interrupt.</p>
4: 0	AWDCH [4:0]	RW	0	<p>Analog Watchdog Channel Select Bit</p> <p>This bit is set by a software write to select the input channel for the analog watchdog.</p> <p>00000: ADC analog input channel 0 00001: ADC analog input channel 1 01111: ADC analog input channel 15 10000: OPA3_VIN 10001: VREFINT input 10010: VCCA/3 10011: DAC1_VIN 10100 DAC2_VIN 10101: OPA1_VIN 10110: OPA2_VIN 10111: TS_VIN input Retain all other values.</p>

16.11.3. ADC Control Register (ADC_CR2)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res				VREF- BUFF_SEL		VREF BUFFERE	Res	TSVREFE	SWSTART	JSWSTART	EX- TTRIG	EXTSEL [2:0]			Res
-				RW		RW	-	RW	R_W1	R_W1	RW	RW			-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JEXTTRIG		JEXTSEL [2:0]			ALIGN	Res	Res	DMA	Res	Res	Res	RSTCAL	CAL	CONT	ADON
RW					-	-	RW	-				R_W1		RW	RW

Bit	Name	R/W	Reset Value	Function
31: 28	Res	-	-	Res
27: 26	VREFBUFF_SEL	RW	0	VREFBUF Voltage selection 00: 1.5 V 01: 2.048 V 10: 2.5V 11: Res
25	VREF BUFERE	RW	0	VerfBuffer enable The software writes 0 to set 0 and 1 to set 1. 0: Disable VerfBuffer 1: Enable VerfBuffer
24	Res	-	-	Res
23	TSVREFE	RW	0	Temperature Sensor and VREFINT Enable Write 1 to set 1 and Write 0 to set 0 via software are used to enable or disable the temperature sensor and VREFINT channels. can be enabled in the ADC. 0: Disable temperature sensor and VREFINT 1: Enable temperature sensor and VREFINT
22	SWSTART	R_W1	0	Start conversion rule channel enable The conversion is initiated by a software Write 1 Set 1, which is cleared by hardware and set to 0 immediately after the conversion is initiated. 0: reset state 1: Start converting rule channels
21	JSWSTART	R_W1	0	Start conversion injection channel enable

Bit	Name	R/W	Reset Value	Function
				<p>The conversion is initiated by a software write 1 and set to 1. The conversion is cleared by hardware and set to 0 immediately after initiating the conversion.</p> <p>0: reset state</p> <p>1: Start switching injection channels</p>
20	EXTTRIG	RW	0	<p>Rule Channel External Trigger Enable</p> <p>This bit is cleared by the software Write 1 Set 1 and Write 0 Set 0, and is used to enable or disable external trigger signals that can initiate rule channel group conversions.</p> <p>0: Disable external triggering of the rule channel</p> <p>1: Enable rule channel external trigger</p>
19: 17	EXTSEL [2:0]	RW	0	<p>Rule Channel External Trigger Event Selection Bit. Select the external event that initiates the rule channel group conversion</p> <p>The ADC external trigger events are as follows:</p> <p>000: CH1 event of timer 1</p> <p>001: CH2 event of timer 1</p> <p>010: CH3 event of timer 1</p> <p>011: CH2 event for timer 2</p> <p>100: TRGO event for timer 3</p> <p>101: trigger timer15_TRGO event</p> <p>110: EXTI11</p> <p>111: SWSTART</p>
16	Res	-	-	Res
15	JEXTTRIG	RW	0	<p>Inject Channel External Trigger Enable</p> <p>0: Disable injection channel external trigger</p> <p>1: Enable injection channel external trigger</p>
14: 12	JEXTSEL [2:0]	RW	0	<p>Inject channel group external trigger event select bit</p> <p>The external trigger events for the ADC are as follows:</p> <p>000: TRGO event of timer 1</p> <p>001: CH4 event for timer 1</p> <p>010: TRGO event of timer 2</p> <p>011: CH1 event of timer 2</p> <p>100: TRGO event for timer 3</p> <p>101: TRGO event of timer 15</p>

Bit	Name	R/W	Reset Value	Function
				110: EXTI15 111: JSWSTART
11	ALIGN	RW	0	Data Alignment Control Bit This bit is cleared by the software Write 1 Set 1 and Write 0 Set 0. 0: right-aligned 1: Left Alignment
10: 9	Res	-	-	Res
8	DMA	RW	0	DMA enable bit This bit is cleared by the software Write 1 Set 1 and Write 0 Set 0. 0: Disable DMA mode 1: Enable DMA mode
7: 4	Res	-	-	Res
3	RSTCAL	Res	0	Calibration reset enable bit This bit is set to 1 by a software write 1 and to 0 by a hardware clear. This bit is cleared after the calibration register is initialized (i.e., after RSTCAL is set to 1). 0: Calibration register initialized 1: Initialize the calibration registers Note: When a conversion is in progress, clearing the calibration register requires an additional cycle if RSTCAL is set.
2	CAL	R_W1	0	Calibration Enable This bit is set to 1 by software write 1 to start calibration and cleared by hardware on calibration failure or calibration success. When ADON is invalid and SWSTART, JSWSTART is invalid; the software initiates calibration. 0: Calibration complete 1: Enable Calibration
1	CONT	RW	0	Continuous Conversion Enable This bit is cleared by the software Write 1 Set 1 and Write 0 Set 0. If this bit is set, the conversion will continue until the bit is cleared. 0: Single conversion mode 1: Continuous conversion mode

Bit	Name	R/W	Reset Value	Function
0	ADON	RW	0	<p>On/Off A/D converter</p> <p>ADC converter work enable, when set to 1 the ADC converter wakes up, there is a delay tSTAB between the converter powering up and starting the conversion. When set to 0 the ADC converter is in the power-down state.</p> <p>0: ADC conversion disabled, ADC converter enters power-down mode</p> <p>1: Enable the ADC converter.</p> <p>Note: If SWSTART,JSWSTART is changed in this register along with ADON, the conversion is not triggered. This is to prevent triggering incorrect conversions.</p>

16.11.4. ADC Sample Time Register 1 (ADC_SMPR1)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res				SMP23[2: 0]			SMP22[2: 0]			SMP21[2: 0]			SMP20[2: 0]		
-				RW			RW			RW			RW		

Bit	Name	R/W	Reset Value	Function
31: 12	Res	-	-	Res
11: 0	SMPx[2:0]	RW	0	<p>Selects the sampling time for channel x</p> <p>Setting these bits via software is used to independently select the sampling time for each channel. The channel select bit must remain unchanged during the sample cycle.</p> <p>000: 3.5 cycles 100: 28.5 cycles</p> <p>001: 5.5 cycles 101: 41.5 cycles</p> <p>010: 7.5 cycles 110: 134.5 cycles</p> <p>011: 13.5 cycles 111: 239.5 cycles</p>

16.11.5. ADC Sample Time Register 2 (ADC_SMPR2)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res		SMP19[2: 0]			SMP18[2: 0]			SMP17[2: 0]			SMP16[2: 0]			SMP15[2: 1]	
-		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	

Bit	Name	R/W	Reset Value	Function
31: 30	Res	-	-	Res
29: 0	SMPx[2:0]	RW	0	<p>Selects the sampling time for channel x</p> <p>Setting these bits via software is used to independently select the sampling time for each channel. The channel select bit must remain unchanged during the sample cycle.</p> <p>000: 3.5 cycles 100: 28.5 cycles</p> <p>001: 5.5 cycles 101: 41.5 cycles</p> <p>010: 7.5 cycles 110: 134.5 cycles</p> <p>011: 13.5 cycles 111: 239.5 cycles</p>

16.11.6. ADC Sample Time Register 3 (ADC_SMPR3)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res		SMP9[2: 0]			SMP8[2: 0]			SMP7[2: 0]			SMP6[2: 0]			SMP5[2: 1]	
-		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	

Bit	Name	R/W	Reset Value	Function
31: 30	Res	-	-	Res
29: 0	SMPx[2:0]	RW	0	<p>Selects the sampling time for channel x</p> <p>Setting these bits via software is used to independently select the sampling time for each channel. The channel select bit must remain unchanged during the sample cycle.</p> <p>000: 3.5 cycles 100: 28.5 cycles</p> <p>001: 5.5 cycles 101: 41.5 cycles</p> <p>010: 7.5 cycles 110: 134.5 cycles</p> <p>011: 13.5 cycles 111: 239.5 cycles</p>

16.11.7. ADC injection channel data offset register x (ADC_JOFRx) (x=1..4)

Address offset: 0x18-0x24

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 12	Res	-	-	Res
11: 0	JOFFSETx[11:0]	RW	0	Inject channel xth conversion data offset The software configures the value of these bits. These bits define the value to be used to subtract from the original converted data when converting the injected channel. The final conversion result can be read out in the ADC_JDRx register.

16.11.8. ADC Watchdog High Threshold Register (ADC_HTR)

Address offset: 0x28

Reset value: 0x0000 0FFF

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

Bit	Name	R/W	Reset Value	Function
31: 12	Res	-	-	Res
11: 0	HTR [11:0]	RW	0xFFFF	Analog Watchdog High Threshold The software configures the value of these bits. These bits define the analog watchdog's threshold high limit.

16.11.9. ADC Watchdog Low Threshold Register (ADC_LTR)

Address offset: 0x2C

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 12	Res	-	-	Res
11: 0	LTR [11:0]	RW	0x000	Analog Watchdog Low Threshold The software configures the value of these bits. These bits define the threshold low limit for the analog watchdog.

16.11.10. ADC Rule Sequence Register 1 (ADC_SQR1)

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res								L[3:0]				SQ16[4: 1]			
-								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				

Bit	Name	R/W	Reset Value	Function
31: 24	Res	-	-	Res
23: 20	L[3:0]	RW	0	Rule channel sequence length The software configures the value of these bits. These bits define the number of channels in a regular channel conversion sequence. 0000: 1 conversion 0001: 2 conversions 1111: 16 conversions

19: 15	SQ16[4: 0]	RW	0	The software configures the value of these bits. 16th conversion in the rule sequence, these bits define the number of the 16th conversion channel in the conversion sequence (0 to 23).
14: 10	SQ15[4: 0]	RW	0	The software configures the value of these bits. 15th conversion in the rule sequence, these bits define the number of the 15th conversion channel in the conversion sequence (0 to 23).
9: 5	SQ14[4: 0]	RW	0	The software configures the value of these bits. The 14th conversion in the rule sequence, these bits define the number of the 14th conversion channel in the conversion sequence (0 to 23).
4: 0	SQ13[4: 0]	RW	0	The software configures the value of these bits. The 13th conversion in the rule sequence, these bits define the number of the 13th conversion channel in the conversion sequence (0 to 23).

16.11.11. ADC Rule Sequence Register 2 (ADC_SQR2)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res		SQ12[4: 0]						SQ11[4: 0]				SQ10[4: 1]			
-		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				

Bit	Name	R/W	Reset Value	Function
31: 30	Res	-	-	Res
29: 25	SQ12[4: 0]	RW	0	The software configures the value of these bits. The 12th conversion in the rule sequence, these bits define the number of the 12th conversion channel in the conversion sequence (0 to 23).
24: 20	SQ11[4: 0]	RW	0	The software configures the value of these bits. The 11th conversion in the rule sequence, these bits define the number of the 11th conversion channel in the conversion sequence (0 to 23).
19: 15	SQ10[4: 0]	RW	0	The software configures the value of these bits. The 10th conversion in the rule sequence, these bits define

				the number of the 10th conversion channel in the conversion sequence (0 to 23).
14: 10	SQ9[4: 0]	RW	0	The software configures the value of these bits. 9th conversion in the rule sequence, these bits define the number of the 9th conversion channel in the conversion sequence (0 to 23).
9: 5	SQ8[4: 0]	RW	0	The software configures the value of these bits. The 8th conversion in the rule sequence, these bits define the number of the 8th conversion channel in the conversion sequence (0 to 23).
4: 0	SQ7[4: 0]	RW	0	The software configures the value of these bits. The 7th conversion in the rule sequence, these bits define the number of the 7th conversion channel in the conversion sequence (0 to 23).

16.11.12. ADC Rule Sequence Register 3 (ADC_SQR3)

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res		SQ6[4: 0]						SQ5[4: 0]				SQ4[4: 1]			
-		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			

Bit	Name	R/W	Reset Value	Function
31: 30	Res	-	-	Res
29: 25	SQ6[4: 0]	RW	0	The software configures the value of these bits. The 6th conversion in the rule sequence, these bits define the number of the 6th conversion channel in the conversion sequence (0 to 23).
24: 20	SQ5[4: 0]	RW	0	The software configures the value of these bits. The 5th conversion in the rule sequence, these bits define the number of the 5th conversion channel in the conversion sequence (0 to 23).
19: 15	SQ4[4: 0]	RW	0	The software configures the value of these bits. 4th conversion in the rule sequence, these bits define the number of the 4th conversion channel in the conversion sequence (0 to 23).

14: 10	SQ3[4: 0]	RW	0	The software configures the value of these bits. The 3rd conversion in the rule sequence, these bits define the number of the 3rd conversion channel in the conversion sequence (0 to 23).
9: 5	SQ2[4: 0]	RW	0	The software configures the value of these bits. The 2nd conversion in the rule sequence, these bits define the number of the 2nd conversion channel in the conversion sequence (0 to 23).
4: 0	SQ1[4: 0]	RW	0	The software configures the value of these bits. The 1st conversion in the rule sequence, these bits define the number of the 1st conversion channel in the conversion sequence (0 to 23).

16.11.13. ADC Injection Sequence Register (ADC_JSQR)

Address offset: 0x3C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res										JL [1:0]		JSQ4[4: 1]			
-										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				

Bit	Name	R/W	Reset Value	Function
31: 22	Res	-	-	Res
21: 20	JL [1:0]	RW	0	Injection channel sequence length The software configures the value of these bits. These bits define the number of channels in the injected channel transition sequence. 00: 1 conversion 01: 2 conversions 10: 3 conversions 11: 4 conversions
19: 15	JSQ4[4: 0]	RW	0	Inject the 4th conversion in the sequence The software configures the value of these bits. These bits define the number (0 to 23) of the 4th conversion channel in the conversion sequence.

				<p>Note: Unlike regular conversion sequences, if the length of JL[1:0] is less than 4, the sequence order of conversions starts from (4-JL).</p> <p>For example, ADC_JSQR[21:0] = 10 00011 00011 00111 00010 means that the scan conversion will be done on the following channels</p> <p>Order conversion: 7, 3, 3 instead of 2, 7, 3</p>
14: 10	JSQ3[4: 0]	RW	0	The software configures the value of these bits. Inject the 3rd transition in the sequence
9: 5	JSQ2[4: 0]	RW	0	The software configures the value of these bits. Inject the 2nd transition in the sequence
4: 0	JSQ1[4: 0]	RW	0	The software configures the value of these bits. Inject the 1st conversion in the sequence

16.11.14. ADC Injection Data Register x (ADC_JDRx) (x= 1..4)

Address offset: 0x40-4C

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 16	Res	-	-	Res
15: 0	JDATA [15:0]	R	0	<p>Injecting channel conversion results</p> <p>The software can read the value of these bits. These bits are read-only and contain the conversion results of the injected channel. Whether the data is left- or right-aligned</p>

16.11.15. ADC rule data register (ADC_DR)

Address offset: 0x50

Reset value: 0x0000 0000

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

DATA[15:0]
R

Bit	Name	R/W	Reset Value	Function
31: 16	Res	-	-	Res
15: 0	DATA[15:0]	R	0	Rule channel conversion results The software can read the value of these bits. These bits are read-only and contain the conversion results for the rule channel. Data is left or right aligned

16.11.16. ADC Calibration Configuration and Status Register (ADC_CCSR)

Address offset: 0x54

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CALON.	CAPSUC	OFFSUC	Res												
R	RC_W1	RC_W1	-												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CALSET	CALBYP	CALSMP [1:0]		CALSEL	Res										
R_W1	R_W1	RW		RW	-										

Bit	Name	R/W	Reset Value	Function
31	CALON	R	0	Calibration flag, signifies that ADC calibration is in progress. 1: ADC calibration is in progress 0: ADC calibration is finished or ADC calibration has not been initiated
30	CAPSUC	RC_W1	0	Capacitor calibration status bit. Indicates if the ADC capacitor calibration was successful. Hardware set to 1; software write 1 set to 0; CALON=0, CALSEL=0,CALSUC=1: invalid state CALON=0, CALSEL=0, CALSUC=0: CAPs not calibrated CALON = 0, CALSEL = 1, CALSUC = 1: ADC CAPs calibrated successfully CALON = 0, CALSEL = 1, CALSUC = 0: ADC CAPs calibration failure

Bit	Name	R/W	Reset Value	Function
29	OFFSUC	RC_W1	0	<p>Offset calibration status bit.</p> <p>Indicates whether ADC offset calibration was successful. Hardware set to 1; software write 1 set to 0;</p> <p>CALON=0, CALSEL=0, OFFSUC=0: ADC OFFSET calibration failure</p> <p>CALON=0, CALSEL=0, OFFSUC=1: ADC OFFSET calibration successful!</p> <p>CALON=0, CALSEL=1, OFFSUC=1: ADC OFFSET calibration successful!</p> <p>CALON=0, CALSEL=1, OFFSUC=0: ADC OFFSET calibration failure</p>
28: 16	Res	-	-	Res
15	CALSET	RC_W1	0	<p>Calibration factor selection. When CAL is 0, software write 1 is set to 1. Hardware set to 0 when CAL is valid or injection/rule channel SWSTART, JWSTART is valid.</p> <p>1: Setting CAL_CXIN data as the final calibration data</p> <p>0: Closes the path from CAL_CXIN to CAL_CXOUT and selects the result generated internally by the calibration circuit.</p>
14	CALBYP	R_W1	0	<p>Calibration factor bypass. When CAL is 0, software write 1 is set to 1. Hardware set to 0 when CAL is valid or injection/rule channel SWSTART, JWSTART is valid.</p> <p>1: Calibration results in a reset value</p> <p>0: Calibration results are self-calibration results or calibration factor inputs</p>
13: 12	CALSMP [1:0]	RW	0	<p>Calibration sample time selection Configure the number of clock cycles for the calibration sample phase based on the following information:</p> <p>00: 1 ADC clock cycle</p> <p>01: 2 ADC clock cycles</p> <p>10: 4 ADC clock cycles</p> <p>11: 8 ADC clock cycles</p>

Bit	Name	R/W	Reset Value	Function
				The longer the SMP period is configured during calibration, the more accurate the calibration results will be, but this configuration introduces the problem of extended calibration period
11	CALSEL	RW	0	Calibration content selection bit, used to select the content to be calibrated 1: calibrate OFFSET and linearity 0: calibrate OFFSET only
10: 0	Res	-	-	Res

17. Liquid Crystal Controller (LCD)

17.1. Introduction

The LCD Controller is a digital controller/driver for monochrome passive liquid crystal displays (LCDs) with up to 8 common terminals (COM) and 40 segment terminals (SEG) to drive 160 (4x40) or 288 (8x36) LCD pixels. The number of terminals depends on the device pins described in the datasheet. The LCD consists of a number of segments (pixels or full symbols) that can be lit up or extinguished. Each zone contains a layer of liquid crystal molecules aligned between two electrodes. When a voltage above the threshold voltage is applied to the liquid crystal, the corresponding zone is visible. The zone voltage must be AC to avoid electrophoretic effects in the liquid crystal (which would affect the display). After that, waveforms must be generated at both ends of the zone to avoid DC.

Glossary

- Liquid Crystal (LCD): Passive display panel with terminals leading directly to the zone.
- Common (COM): Electrical connection terminals connected to multiple zones.
- Bias (BIAS): Level used when driving the LCD, defined as $1/(\text{Voltage level to drive the LCD display} - 1)$.
- Segment (SEG): The smallest visual unit (the smallest constituent element, line or dot, on an LCD display).
- Duty Cycle (DUTY): $1/\text{number of common terminals on the LCD display}$.
- Frame: one cycle of the waveform written to the zone.
- Frame Rate: The number of frames per second, i.e., the number of times per second that the LCD segment is excited.

17.2. LCD Main Characteristics

- Highly flexible frame rate control.
- Supports static, 1/2, 1/3, 1/4, 1/6, and 1/8 duty cycles.
- Supports static, 1/2, and 1/3 bias voltages.
- Up to 16 registers of LCD data RAM.
- The contrast of the LCD can be configured through software.
- 2 drive waveform generation methods
 - Internal resistor voltage divider, external resistor voltage divider
 - Software configurable internal resistor divider power consumption to match the capacitive charge required by the LCD panel
- Supports low-power modes: The LCD controller can display in Run, Sleep, and Stop modes.
- Configurable frame interrupt.
- Supports LCD blinking function and configurable blinking frequency.
- Unused LCD segments and common pins can be configured for digital or analog functions.

17.3. LCD Block Diagram

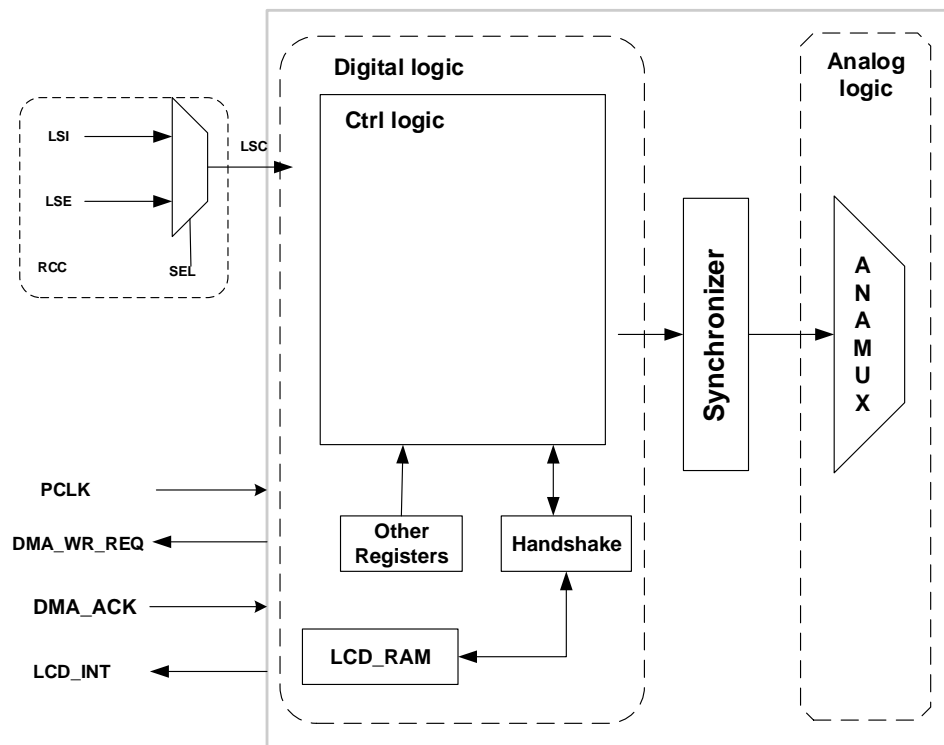


Figure 17-1 LCD Block Diagram

17.4. LCD clock

The LCD clock source can be selected as LSI or LSE, and the input clock can be determined by LSCSEL and LSCOEN of RCC_BDCR register.

17.5. LCD Driver Waveforms

LCD supports 5 duty cycle (Duty) drive waveforms: Static, 1/2, 1/3, 1/4, 1/6 and 1/8, which are set by LCD_CR0.DUTY.

LCD supports 2 types of bias (BIAS) drive waveforms: Static, 1/2, 1/3, set by LCD_CR0.BIAS.

The recommended combination is shown in the table below:

Table 17-1 LCD Supported Driver Waveforms

-	1/2 DUTY	1/3 DUTY	1/4 DUTY	1/6 DUTY	1/8 DUTY
1/2 BIAS	✓	✓	not recommended	not recommended	not recommended
1/3 BIAS	not recommended	not recommended	✓	✓	✓

17.5.1. Static drive waveform

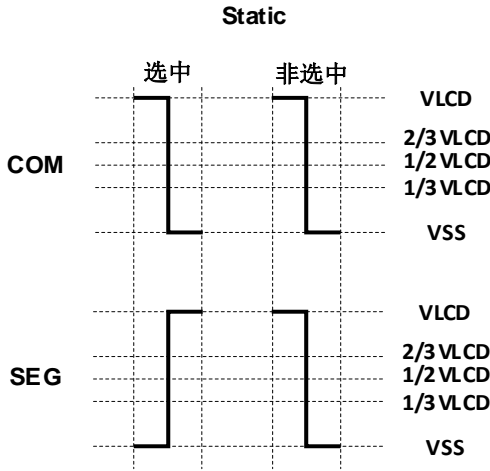


Figure 17-2 Static drive waveform

17.5.2. 1/2DUTY 1/2BIAS Drive Waveforms

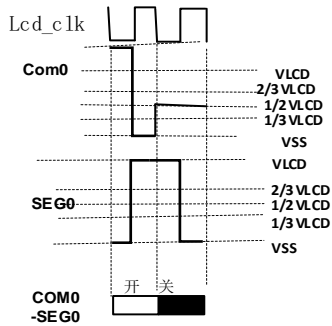


Figure 17-3 1/2DUTY 1/2BIAS Driver Waveforms

17.5.3. 1/8DUTY 1/3BIAS Drive Waveforms

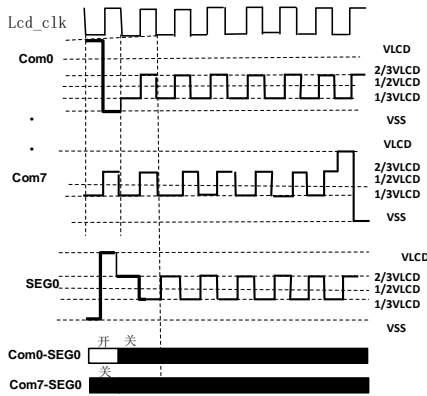


Figure 17-4 1/8DUTY 1/3 BIAS Drive Waveforms

17.6. LCD BIAS Generation Circuit

The BIAS voltage of LCD has two sources: internal resistor voltage divider and external resistor voltage divider. When internal resistor divider is selected, the chip automatically switches the internal circuitry to produce a voltage that complies with BIAS and DUTY. When external resistor voltage divider is selected, it requires the user to build the relevant circuits on the peripheral pins of the chip.

17.6.1. Internal Resistance Mode

Internal Resistor Mode VLCDH, VLCD1~VLCD3 can be used as LCD SEG output or IO port.

For internal resistor mode, the LCD drive voltage is controlled by CR0.CONTRAST as shown in the table below:

Table 17-2 Internal Resistance Mode

CR0.CONTRAST	VLCD (1/3 BIAS)	VLCD (1/2 BIAS)
0	1.00 * VCCA	1.00 * VCCA
1	0.95 * VCCA	0.92 * VCCA
2	0.9 * VCCA	0.86 * VCCA
3	0.86 * VCCA	0.8 * VCCA
4	0.82 * VCCA	0.75 * VCCA
5	0.78 * VCCA	0.71 * VCCA
6	0.75 * VCCA	0.67 * VCCA
7	0.72 * VCCA	0.63 * VCCA
8	0.69 * VCCA	0.60 * VCCA
9	0.67 * VCCA	0.57 * VCCA
10	0.64 * VCCA	0.55 * VCCA
11	0.62 * VCCA	0.52 * VCCA
12	0.60 * VCCA	0.50 * VCCA
13	0.58 * VCCA	0.48 * VCCA
14	0.56 * VCCA	0.46 * VCCA
15	0.55 * VCCA	0.44 * VCCA

17.6.2. External Resistance Mode

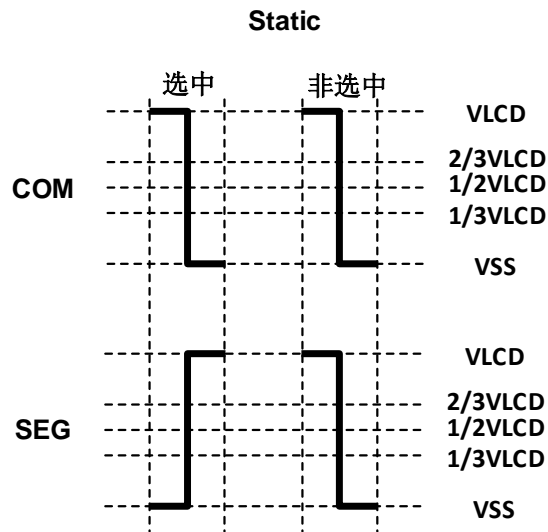


Figure 17-5 External Resistance Mode

Attention:

1. Rx is an adjustable resistor used to adjust the LCD display contrast.
2. Select the appropriate resistor R according to the LCD screen to be used.

17.7. DMA

The LCD supports both software and hardware triggered DMA data transfers to automatically move the content to be displayed from the storage content to the LCD display RAM. Hardware triggering uses a frame interrupt signal.

DMA Data Transfer Configuration Flow:

1. Enable DMA
2. Select LCD DMA
3. Setting the transmission type, transmission length, and transmission method
4. Set source start address, destination start address
5. Setting the incremental method of source and destination addresses
6. Enable DMA interrupts as needed
7. Enable LCD DMA Trigger

17.8. Disruptions

When the LCD setting is active, the LCD interrupt can be configured to generate an interrupt for the number of frames.

17.9. LCD display mode

The LCD supports two display modes. One type of display unit is COM, and all COM segments of the same SEG are in the same byte (mode 0). The other is that different SEGs of the same COM are in the same byte (mode 1).

Selecting the appropriate display for the LCD panel can simplify program operation.

17.9.1. LCD display mode 1 (MODE = 1)

■ 1/8 DUTY

	Bit31	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24	Bit23	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	
	SEG31	SEG30	SEG29	SEG28	SEG27	SEG26	SEG25	SEG24	SEG23	SEG22	SEG21	SEG20	SEG19	SEG18	SEG17	SEG16	SEG15	SEG14	SEG13	SEG12	SEG11	SEG10	SEG9	SEG8	SEG7	SEG6	SEG5	SEG4	SEG3	SEG2	SEG1	SEG0	
COM0																																	LCDRAM0
COM1																																	LCDRAM1
COM2																																	LCDRAM2
COM3																																	LCDRAM3
COM4																																	LCDRAM4
COM5																																	LCDRAM5
COM6																																	LCDRAM6
COM7																																	LCDRAM7
																													SEG35	SEG34	SEG33	SEG32	
																								COM0									LCDRAM8
																								COM1									LCDRAM9
																								COM2									LCDRAMA
																								COM3									LCDRAMB
																								COM4									LCDRAMC
																								COM5									LCDRAMD
																								COM6									LCDRAME
																								COM7									LCDRAMF

Figure 17-6 1/8 DUTY LCD Display Mode 1

■ 1/6 DUTY

	Bit31	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24	Bit23	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	
	SEG31	SEG30	SEG29	SEG28	SEG27	SEG26	SEG25	SEG24	SEG23	SEG22	SEG21	SEG20	SEG19	SEG18	SEG17	SEG16	SEG15	SEG14	SEG13	SEG12	SEG11	SEG10	SEG9	SEG8	SEG7	SEG6	SEG5	SEG4	SEG3	SEG2	SEG1	SEG0	
COM0																																	LCDRAM0
COM1																																	LCDRAM1
COM2																																	LCDRAM2
COM3																																	LCDRAM3
COM4																																	LCDRAM4
COM5																																	LCDRAM5
																																	LCDRAM6
																																	LCDRAM7
																											SEG37	SEG36	SEG35	SEG34	SEG33	SEG32	
COM0																																	LCDRAM8
COM1																																	LCDRAM9
COM2																																	LCDRAMA
COM3																																	LCDRAMB
COM4																																	LCDRAMC
COM5																																	LCDRAMD
																																	LCDRAME
																																	LCDRAMF

Figure 17-7 1/6 DUTY LCD Display Mode 1

■ 1/4 DUTY

	Bit31	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24	Bit23	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	
	SEG31	SEG30	SEG29	SEG28	SEG27	SEG26	SEG25	SEG24	SEG23	SEG22	SEG21	SEG20	SEG19	SEG18	SEG17	SEG16	SEG15	SEG14	SEG13	SEG12	SEG11	SEG10	SEG9	SEG8	SEG7	SEG6	SEG5	SEG4	SEG3	SEG2	SEG1	SEG0	
COM0																																	LCDRAM0
COM1																																	LCDRAM1
COM2																																	LCDRAM2
COM3																																	LCDRAM3
																																	LCDRAM4
																																	LCDRAM5
																																	LCDRAM6
																																	LCDRAM7
																									SEG39	SEG38	SEG37	SEG36	SEG35	SEG34	SEG33	SEG32	
COM0																																	LCDRAM8
COM1																																	LCDRAM9
COM2																																	LCDRAMA
COM3																																	LCDRAMB
																																	LCDRAMC
																																	LCDRAMD
																																	LCDRAME
																																	LCDRAMF

Figure 17-8 1/4 DUTY LCD Display Mode 1

■ 1/3 DUTY 1/2 DUTY

	Bit31	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24	Bit23	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	
	SEG31	SEG30	SEG29	SEG28	SEG27	SEG26	SEG25	SEG24	SEG23	SEG22	SEG21	SEG20	SEG19	SEG18	SEG17	SEG16	SEG15	SEG14	SEG13	SEG12	SEG11	SEG10	SEG9	SEG8	SEG7	SEG6	SEG5	SEG4	SEG3	SEG2	SEG1	SEG0	
COM0																																	LCDRAM0
COM1																																	LCDRAM1
COM2																																	LCDRAM2
																																	LCDRAM3
																																	LCDRAM4
																																	LCDRAM5
																																	LCDRAM6
																																	LCDRAM7
																									SEG39	SEG38	SEG37	SEG36	SEG35	SEG34	SEG33	SEG32	
COM0																																	LCDRAM8
COM1																																	LCDRAM9
COM2																																	LCDRAMA
																																	LCDRAMB
																																	LCDRAMC
																																	LCDRAMD
																																	LCDRAME
																																	LCDRAMF

Figure 17-9 1/3 1/2 DUTY LCD Display Mode 1

17.9.2. LCD display mode 0 (MODE = 0)

■ 1/8 DUTY

Bit31	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24	Bit23	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	
COM7	COM6	COM5	COM4	COM3	COM2	COM1	COM0	COM7	COM6	COM5	COM4	COM3	COM2	COM1	COM0	COM7	COM6	COM5	COM4	COM3	COM2	COM1	COM0	COM7	COM6	COM5	COM4	COM3	COM2	COM1	COM0	
							SEG3								SEG2								SEG1								SEG0	LCDRAM0
							SEG7								SEG6								SEG5								SEG4	LCDRAM1
							SEG11								SEG10								SEG9								SEG8	LCDRAM2
							SEG15								SEG14								SEG13								SEG12	LCDRAM3
							SEG19								SEG18								SEG17								SEG16	LCDRAM4
							SEG23								SEG22								SEG21								SEG20	LCDRAM5
							SEG27								SEG26								SEG25								SEG24	LCDRAM6
							SEG31								SEG30								SEG29								SEG28	LCDRAM7
																															SEG32	LCDRAM8
																															SEG33	LCDRAM9
																															SEG34	LCDRAMA
																															SEG35	LCDRAMB
																																LCDRAMC
																																LCDRAMD
																																LCDRAME
																																LCDRAMF

Figure 17-10 1/8 DUTY LCD Display Mode 0

■ 1/6 DUTY

Bit31	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24	Bit23	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0																								
		COM5	COM4	COM3	COM2	COM1	COM0			COM5	COM4	COM3	COM2	COM1	COM0			COM5	COM4	COM3	COM2	COM1	COM0			COM5	COM4	COM3	COM2	COM1	COM0																								
							SEG3								SEG2								SEG1								SEG0	LCDRAM0																							
							SEG7								SEG6								SEG5								SEG4	LCDRAM1																							
							SEG11								SEG10								SEG9								SEG8	LCDRAM2																							
							SEG15								SEG14								SEG13								SEG12	LCDRAM3																							
							SEG19								SEG18								SEG17								SEG16	LCDRAM4																							
							SEG23								SEG22								SEG21								SEG20	LCDRAM5																							
							SEG27								SEG26								SEG25								SEG24	LCDRAM6																							
							SEG31								SEG30								SEG29								SEG28	LCDRAM7																							
																																																				SEG32	LCDRAM8		
																																																						SEG33	LCDRAM9
																																																						SEG34	LCDRAMA
																																																						SEG35	LCDRAMB
																																																						SEG36	LCDRAMC
																																																						SEG37	LCDRAMD
																																																							LCDRAME
																																																					LCDRAMF		

Figure 17-11 1/6 DUTY LCD Display Mode 0

■ 1/4 DUTY

Bit31	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24	Bit23	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0																							
				COM3	COM2	COM1	COM0					COM3	COM2	COM1	COM0					COM3	COM2	COM1	COM0					COM3	COM2	COM1	COM0																							
							SEG3								SEG2								SEG1								SEG0	LCDRAM0																						
							SEG7								SEG6								SEG5								SEG4	LCDRAM1																						
							SEG11								SEG10								SEG9								SEG8	LCDRAM2																						
							SEG15								SEG14								SEG13								SEG12	LCDRAM3																						
							SEG19								SEG18								SEG17								SEG16	LCDRAM4																						
							SEG23								SEG22								SEG21								SEG20	LCDRAM5																						
							SEG27								SEG26								SEG25								SEG24	LCDRAM6																						
							SEG31								SEG30								SEG29								SEG28	LCDRAM7																						
																																																				SEG32	LCDRAM8	
																																																					SEG33	LCDRAM9
																																																					SEG34	LCDRAMA
																																																					SEG35	LCDRAMB
																																																					SEG36	LCDRAMC
																																																					SEG37	LCDRAMD
																																																					SEG38	LCDRAME
																																																			SEG39	LCDRAMF		

Figure 17-12 1/4 DUTY LCD Display Mode 0

■ 1/3 DUTY 1/2 DUTY

Bit31	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24	Bit23	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0																								
					COM2	COM1	COM0						COM2	COM1	COM0						COM2	COM1	COM0						COM2	COM1	COM0																								
							SEG3								SEG2								SEG1								SEG0	LCDRAM0																							
							SEG7								SEG6								SEG5								SEG4	LCDRAM1																							
							SEG11								SEG10								SEG9								SEG8	LCDRAM2																							
							SEG15								SEG14								SEG13								SEG12	LCDRAM3																							
							SEG19								SEG18								SEG17								SEG16	LCDRAM4																							
							SEG23								SEG22								SEG21								SEG20	LCDRAM5																							
							SEG27								SEG26								SEG25								SEG24	LCDRAM6																							
							SEG31								SEG30								SEG29								SEG28	LCDRAM7																							
																																																				SEG32	LCDRAM8		
																																																						SEG33	LCDRAM9
																																																						SEG34	LCDRAMA
																																																						SEG35	LCDRAMB
																																																						SEG36	LCDRAMC
																																																						SEG37	LCDRAMD
																																																						SEG38	LCDRAME
																																																			SEG39	LCDRAMF			

Figure 17-13 1/3 1/2 DUTY LCD Display Mode 0

17.10. LCD Register

17.10.1. Configuration Register 0 (LCD_CR0)

Address offset: 0x00

Reset value: 0x0000 00C2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CONTRAST [3:0]				BSEL [2:0]			DUTY [2:0]			BIAS	Res	Res	LCDCLK[1:0]		EN
RW				RW			RW			RW	-	-	RW		RW

Bit	Name	R/W	Reset Value	Function
31: 16	Res	-	-	Res
15: 12	CONTRAST [3:0]	RW	0	<p>LCD Contrast Adjustment</p> <p>Note: Valid only if internal resistor divider is selected for the BIAS voltage source.</p> <p>The larger the value, the smaller the amplitude of the LCD waveform.</p> <p>At 0x0, the LCD waveform amplitude is maximum and the contrast is maximum;</p> <p>.....</p> <p>At 0xF, the LCD waveform amplitude is minimized and the contrast is minimized;</p>
11: 9	BSEL [2:0]	RW	0	<p>BIAS voltage source selection</p> <p>111: Res</p> <p>110: Internal resistor divider, high power mode</p> <p>101: Res</p> <p>100: Internal resistor divider, small power consumption mode</p> <p>011: Res</p> <p>010: Internal resistor divider, mid-power mode</p> <p>001: Res</p> <p>000: External resistor mode, requires external circuitry</p> <p>Note: See datasheet for specific power consumption values.</p>
8: 6	DUTY [2:0]	RW	3'b011	LCD DUTY Configuration

Bit	Name	R/W	Reset Value	Function
				000: Static 001: 1/2 DUTY 010: 1/3 DUTY 011: 1/4 DUTY 100: Res 101: 1/6 DUTY 110: Res 111: 1/8 DUTY
5	BIAS	RW	0	0: 1/3 bias (initial value) 1: 1/2 Bias
4: 3	Res	-	-	Res
2: 1	LCDCLK[1:0]	RW	2'b01	LCD scanning frequency selection 00: 64 Hz 01: 128 Hz 10: 256 Hz 11: 512 Hz Note: LCD frame rate = LCD scanning frequency × DUTY
0	EN	RW	0	LCD Enable Control 1: Enabling 0: Prohibited

17.10.2. Configuration Register 1 (LCD_CR1)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	INTF	DMAEN	IE	MODE	Res	BLINKEN	BLINKCNT [5:0]					
-	-	-	-	R	RW	RW	RW	-	RW	RW					

Bit	Name	R/W	Reset Value	Function
31: 12	Res	-	-	Res
11	INTF	R	0	LCD interrupt flag 1: Interruptions 0: no interrupt
10	DMAEN	RW	0	DMA hardware trigger enable 1: Enable LCD interrupt to trigger DMA

Bit	Name	R/W	Reset Value	Function
				0: disable LCD interrupt trigger DMA
9	IE	RW	0	interrupt enable 1: Enabling 0: Prohibited
8	MODE	RW	0	LCD RAM display mode selection 0: Mode 0 1: Mode 1
7	Res	-	-	Res
6	BLINKEN	RW	0	LCD Splash Screen Configuration 1: Enabling 0: Prohibited
5: 0	BLINKCNT [5:0]	RW	0	Splash screen frequency and LCD interrupt interval setting Note: LCD blink frequency is = LCD frame frequency / (BLINKCNT+1) LCD interrupt interval = (BLINKCNT+1)*(1/LCD frame frequency)

17.10.3. Interrupt Clear Register (LCD_INTCLR)

Address offset: 0x08

Reset value: 0x0000 0400

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	INTF_CLR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-					R1W0	-									

Bit	Name	R/W	Reset Value	Function
31: 11	Res	-	-	Res
10	INTF_CLR	RC_W0	1	Interrupt flag clear, write 0 clear, write 1 invalid
9: 0	Res	-	-	Res

17.10.4. Output configuration register (LCD_POEN0)

Address offset: 0x0C

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
S31	S30	S29	S28	S27	S26	S25	S24	S23	S22	S21	S20	S19	S18	S17	S16

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
S15	S14	S13	S12	S11	S10	S9	S8	S7	S6	S5	S4	S3	S2	S1	S0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 0	Sx	RW	0xFFFFFFFF	SEGx output control bits 0: SEG output enable 1: SEG outputs are turned off, so that other functions such as IO, analog inputs and outputs can be used.

17.10.5. Output Configuration Register 1 (LCD_POEN1)

Address offset: 0x10

Reset value: 0x1FFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res			MUX	C3	C2	C1	C0	S36_C	S37_C	S38_C	S39_C	S35	S34	S33	S32
								7	6	5	C4				
-			RW												

Bit	Name	R/W	Reset Value	Function
31: 13	Res	-	-	Res
12	MUX	RW	1	SEG32~SEG35 port function selection, see the following table for details
11: 8	Cx	RW	0xF	COMx output control bits (COM0-COM3) 0: COM output enable 1: COM output off, can use other functions, such as IO, analog input and output
7: 4	SxCy [3:0]	RW	0xF	SEGx/COMy output control bits 0: SEG/COM output enable 1: SEG/COM outputs are turned off so that other functions such as IO, analog inputs and outputs can be used. SEG COM pin function selection is determined by CR0.DUTY
3: 0	S[3:0]	RW	0xF	SEGx output control bits 0: SEG output enable

Bit	Name	R/W	Reset Value	Function
				1: SEG outputs are turned off, so that other functions such as IO, analog inputs and outputs can be used.

Table 17-3 COM/SEG Multiplexing Selection Configuration

VLCDxSEGxPAD			How registers are configured				
			MUX	S<35:32>	BSEL		
When VLCDXSEGXPAD selects GPIO, LCD disable (LCD_ON =0)		VLCDHSEG35=IO	1	1111	X	X	X
		VLCDHSEG34=IO					
		VLCDHSEG33=IO					
		VLCDHSEG32=IO					
When VLCDXSEGXPAD selects IO, LCD enable (LCD_ON =1), only internal resistor operation mode can be selected at this time.	Small re- sistance (high current) mode	VLCDHSEG35=IO	1	1111	1	1	0
		VLCDHSEG34=IO					
		VLCDHSEG33=IO					
		VLCDHSEG32=IO					
	Medium re- sistance (me- dium current) mode	VLCDHSEG35=IO	1	1111	0	1	0
		VLCDHSEG34=IO					
		VLCDHSEG33=IO					
		VLCDHSEG32=IO					
	High re- sistance (low current) mode	VLCDHSEG35=IO	1	1111	1	0	0
		VLCDHSEG34=IO					
		VLCDHSEG33=IO					
		VLCDHSEG32=IO					
Selection of internal resistor operating mode	Small re- sistance (high current) mode	VLCDHSEG35=SEG35/IO	0	XXXX	1	1	0
		VLCDHSEG34=SEG34/IO					
		VLCDHSEG33=SEG33/IO					
		VLCDHSEG32=SEG32/IO					
	Medium re- sistance (me- dium current) mode	VLCDHSEG35=SEG35/IO	0	XXXX	0	1	0
		VLCDHSEG34=SEG34/IO					
		VLCDHSEG33=SEG33/IO					
		VLCDHSEG32=SEG32/IO					
	High re- sistance (low current) mode	VLCDHSEG35=SEG35/IO	0	XXXX	1	0	0
		VLCDHSEG34=SEG34/IO					
		VLCDHSEG33=SEG33/IO					
		VLCDHSEG32=SEG32/IO					
External resistor operating mode selected, internal resistor shorted		VLCDHSEG35=VOUT	0	1111	0	0	0
		VLCDHSEG34=VLCD3					
		VLCDHSEG33=VLCD2					

	VLCDHSEG32=VLCD1					
--	------------------	--	--	--	--	--

17.10.6. LCD_RAM0~7

Address offset: 0x14~0x30

Reset value: 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
D31	D30	D29	D28	D27	D26	D25	D24	D23	D22	D21	D20	D19	D18	D17	D16
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
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 0	Dx	RW	0	LCD dot output, display reference LCD display mode 0: Corresponding SEG COM intersection does not light up 1: Corresponding SEG COM cross is illuminated

Note: When configuring LCD_RAMx, make sure it is done within 2 LCD clocks (LSI or LSE). and the interval between two writes satisfies 2 LCD clocks.

17.10.7. LCD_RAM8~F

Address offset: 0x34~0x50

Reset value: 0x00

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	D7	D6	D5	D4	D3	D2	D1	D0
-	-	-	-	-	-	-	-	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 8	Res	-	-	Res
7: 0	Dx	RW	0	LCD dot output, display reference LCD display mode 0: The corresponding SEG COM intersection is not illuminated; 1: The corresponding SEG COM cross is lit;

Note: When configuring LCD_RAMx, make sure it is done within 2 LCD clocks (LSI or LSE). and the interval between two writes satisfies 2 LCD clocks.

18. Comparator (COMP)

18.1. Introduction

Three General Purpose Comparators (GPC) COMPs are integrated in the chip, namely COMP1, COMP2 and COMP3. These three modules can be used as standalone modules or in combination with Timer.

Comparators can be used as follows:

- Triggered by an analog signal to generate a low-power mode wake-up function
- Analog signal conditioning
- When connected to the PWM output from the Timer, it forms a cycle by cycle current control loop.

18.2. COMP Key Features

- Each comparator has configurable positive or negative inputs for flexible voltage selection:
 - Multiple I/O Pins
 - Temperature sensor output
 - Internal reference voltages VREFBUF, VCCA, VREF1P2 64 fractional values (1/64, 2/64 ... 64/64) supplied via voltage divider
 - VREFINT
 - DAC outputs as INP or INM inputs
 - OPA output as INP input
- Hysteresis function is configurable
- Programmable speed and power consumption
- Outputs can be used as triggers by inputs connected to I/O or Timer.
 - OCREF_CLR event (cycle-by-cycle current control)
 - Brake event (for fast PWM shutdown)
- COMP1 and COMP2 can be combined to form a window comparator
- Each COMP has an interrupt generation capability, which is used to wake up the chip (via EXTI) from low-power modes (Sleep and Stop modes).

18.3. COMP Functional Description

18.3.1. COMP Block Diagram

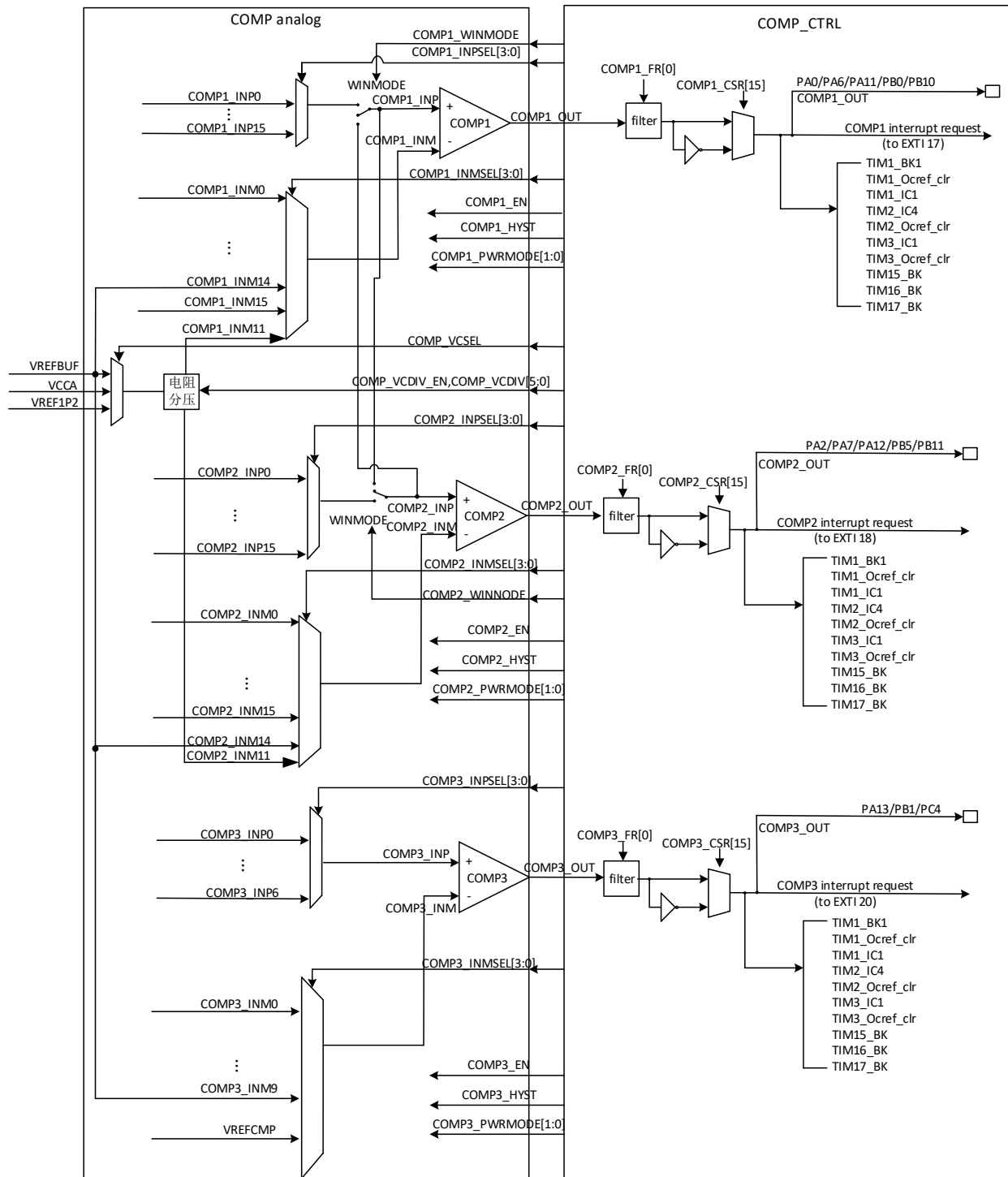


Figure 18-1 Comparator Architecture Block Diagram

18.3.2. COMP pin and internal signals

The I/Os used as comparator inputs must be configured in analog mode in the GPIO registers.

Comparator outputs can be connected to I/O pins through the Alternate function channel in the GPIO.

Outputs can also be internally connected to various Timer inputs for the following purposes:

- Emergency shutdown of the PWM signal when the brake input is connected
- Cycle-by-cycle current control using OCREF_CLR inputs
- Input capture for timing measurements

18.3.3. COMP Reset and Clock

The COMP module has two clock sources:

- PCLK (APB clock), used to clock the configuration registers
- COMP clock, used to clock circuits after the analog comparator output (latch circuits, burr filter circuits, etc. for analog outputs), can be selected as PCLK or LSI or LSE. When it is necessary to work in Stop mode, select LSI or LSE.

Attention:

1. PCLK and COMP CLK are enabled simultaneously by the RCC_APBENR2 control bit, which is turned off to turn PCLK and COMP CLK off, and turned on to turn both PCLK and COMP CLK on if enabled.
2. Before entering Stop mode, it is recommended that you configure the COMP CLK to be LSI or LSE and then enter Stop mode.
3. The COMP module reset signals contain the APB reset source and the COMP module software reset source:
 - 1) PCLK clock domain reset for COMP register reset
 - 2) COMP CLK clock domain reset for resetting circuits after analog comparator outputs (latch circuits, burr filter circuits, etc. for analog outputs).

18.3.4. Window comparator

The window comparator serves to monitor whether the analog voltage is within the low and high thresholds.

Window comparators can be created using two comparators. The analog voltage being monitored is connected to the non-inverting (+ terminal) inputs of both comparators at the same time, and the high and low thresholds are connected to the inverting inputs (- terminals) of each of the two comparators.

By enabling the WINMODE bit, the non-inverting (+ inputs) of the two comparators can be connected together, serving to save an I/O pin.

Note: The WINMODE mode of both COMPs cannot be enabled at the same time.

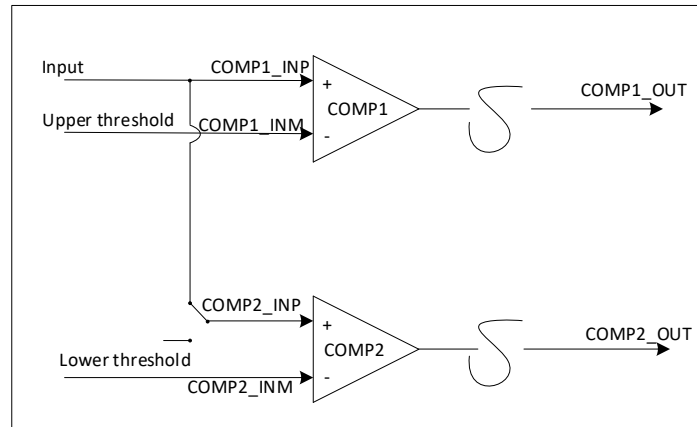


Figure 18-2 Window Comparator

18.3.5. Hysteresis

To avoid false output transitions in case of noisy signals, the comparator can be enabled with hysteresis (COMP1, COMP2 and COMP3 have separate hysteresis enable signals COMP1_HYST, COMP2_HYST and COMP3_HYST).

18.3.6. Power consumption mode

The power consumption and transmission delay of the comparator can be selected in different modes using the PWRMODE[1:0] bits of the COMPx_CSR register to achieve the most suitable trade-off for a particular application. The available modes include both high-speed and medium-speed modes, with relatively higher power consumption and lower transmission latency in high-speed mode. Note that before entering STOP, if PWR_CR2 register LPR=1 is selected (i.e., powering with Low power regulator is selected), COMP needs to be set in medium speed mode first (PWRMODE=01).

18.3.7. Comparator Filter

The output filtering function of COMP and the corresponding filtering width can be enabled by setting the COMP_FR register. Note that this setting should be done before COMP_EN is enabled.

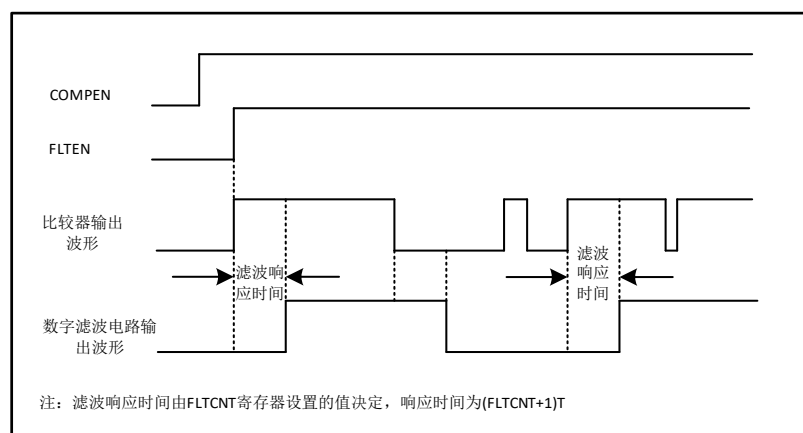


Figure 18-3 Comparator Filtering

18.3.8. COMP interrupt

The comparator output is connected internally to the EXTI controller. Each comparator has a separate EXTI line (17 and 18 and 20) and is capable of generating interrupts or events. The same mechanism is used as a wake-up from low power consumption.

18.4. COMP register

18.4.1. COMP1 Control and Status Register (COMP1_CSR)

Address offset: 0x0200_0000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	COMP_OUT	Res		COMP_VCSEL[1:0]					COMP_VCDIV[5:0]			PWRMODE [1:0]		Res	COMP1_HYST
-	R	-		RW					RW			RW		-	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PO-LARITY		Res		WINMODE	Res			INPSEL [3:0]				INNSEL [3:0]		Res	COMP1_EN
RW		-		RW	-			RW				RW		-	RW

Bit	Name	R/W	Reset Value	Function
31	Res	-	-	Res
30	COMP_OUT	R	0	COMP1 output status This bit is read-only and reflects the output level of COMP1 after polarity selection.
29: 28	Res	-	-	Res
27: 26	COMP_VCSEL	RW	0	COMP1,COMP2,COMP3 Reference Voltage VREF Selection 00: Voltage divider not enabled 01: VCCA 10: VREFBUF 11: VREF1P2
25: 20	COMP_VCDIV[5:0]	RW	100000	Voltage divider selection 00_0000: 1/64 VREFBUF or VCCA or VREF1P2 00_0001: 2/64 VREFBUF or VCCA or VREF1P200_0010: 3/64 VREFBUF or VCCA or VREF1P2...11_1110: 63/64 VREFBUF or VCCA or VREF1P211_111: VREFBUF or VCCA or VREF1P2

Bit	Name	R/W	Reset Value	Function
19: 18	PWRMODE [1:0]	RW	0	COMP1 power consumption mode selection Selected power consumption and consequent speed of COMP1 00: High-speed mode 01: Medium speed mode 10: Res 11: Res
17	Res	-	-	Res
16	COMP1_HYST	RW	0	COMP1 hysteresis function enable control 0: no hysteresis 1: Hysteresis voltage approx. 20 mV
15	POLARITY	RW	0	COMP1 output polarity selection Software readable and writable 0: No inversion 1: Inverted phase
14: 12	Res	-	-	Res
11	WINMODE	RW	0	COMP1 non-inverting output selection (window mode) Software readable and writable 0: Signal selected by INPSEL [3:0] 1: COMP2_INP signal of COMP2 Note that the WINMODE mode of both COMPs cannot be enabled at the same time.
10	Res	-	-	Res
9: 6	INPSEL [3:0]	RW	0	Signal selection for COMP1 non-inverting inputs 0000: COMP1_INP0 from OPA1 output 0001: COMP1_INP1 from PC1 0010: COMP1_INP2 from PC2 0011: COMP1_INP3 from PC3 0100: COMP1_INP4 from PA0 0101: COMP1_INP5 from PA1 0110: COMP1_INP6 from PA2 0111: COMP1_INP7 from PA3 1000: COMP1_INP8 from PA4 1001: COMP1_INP9 from PA5 1010: COMP1_INP10 from PA6 1011: COMP1_INP11 from PA7 1100: COMP1_INP12 from PB4

Bit	Name	R/W	Reset Value	Function
				1101: COMP1_INP13 from PB5 1110: COMP1_INP14 from PB6 1111: COMP1_INP15 from DAC1_VIN
5: 2	INNSEL [3:0]	RW	0	Signal selection for COMP1 inverting inputs 0000: COMP1_INM0 from PA0 0001: COMP1_INM1 from PA1 0010: COMP1_INM2 from PA2 0011: COMP1_INM3 from PA3 0100: COMP1_INM4 from PA4 0101: COMP1_INM5 from PA5 0110: COMP1_INM6 from PA6 0111: COMP1_INM7 from PA7 1000: COMP1_INM8 from PC4 1001: COMP1_INM9 from PC5 1010: COMP1_INM10 from DAC1_VIN 1011: COMP1_INM11 from VREFCMP 1100: COMP1_INM12 from TS_VIN (temperature sensor voltage) 1101: COMP1_INM13 from VREFINT (required to enable the ADC_CR2.TSVREFE register of the ADC module) 1110: COMP1_INM14 from VREFBUF 1111: COMP1_INM15 from PC0
1	Res	-	-	Res
0	COMP1_EN	RW	0	COMP1 enable bit Software is readable and writable (if not locked) 0: not enabled 1: Enabling

18.4.2. COMP1 Filter Register (COMP1_FR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FLTCNT1[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
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	FLTEN1
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	RW

Bit	Name	R/W	Reset Value	Function
31: 16	FLTCNT1[15:0]	RW	0	Comparator 1 Sample Filter Counter The sampling clock is APB or LSI or LSE. The filter count value is configurable. The results are consistently output when the number of samples reaches the filter count value. Sample count period = FLTCNT[15:0]
15: 1	Res	-	-	Res
0	FLTEN1	RW	0	Comparator 1 Digital Filter Function Configuration 0: Disable digital filter function 1: Enable digital filter function Note: This bit must be set when COMP1_EN is 0

18.4.3. COMP2 Control and Status Register (COMP2_CSR)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	COMP_O UT	Res										PWRMODE [1:0]		Res	Res
-	R	-										RW		-	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PO- LARI TY	Res	Res		WINM ODE	Res	INPSEL [3:0]				INNSEL [3:0]				COMP2_H YST	COMP2_E N
RW	-	-		RW	-	RW									

Bit	Name	R/W	Reset Value	Function
31	Res	-	-	Res
30	COMP_OUT	R	0	COMP2 output status This bit is read-only and reflects the output level of COMP2 after polarity selection.
29: 20	Res	-	-	Res
19: 18	PWRMODE [1:0]	RW	0	COMP2 Power Mode Selection Selected power consumption and consequent speed of COMP2 00: High-speed mode 01: Medium speed mode 10: Res 11: Res

Bit	Name	R/W	Reset Value	Function
17: 16	Res	-	-	Res
15	POLARITY	RW	0	COMP2 output polarity selection Software readable and writable 0: No inversion 1: Inverted phase
14: 12	Res	-	-	Res
11	WINMODE	RW	0	COMP2 non-inverting output selection Software readable and writable 0: Signal is selected by INPSEL [1: 0] 1: COMP1_INP signal of COMP1 Note that the WINMODE mode of both COMPs cannot be enabled at the same time.
10	Res	-	-	Res
9: 6	INPSEL [3:0]	RW	0	Signal selection for COMP2 non-inverting inputs, software readable and writable 0000: COMP2_INP0 from PA0 0001: COMP2_INP1 from PA1 0010: COMP2_INP2 from PA2 0011: COMP2_INP3 from PA3 0100: COMP2_INP4 from PA4 0101: COMP2_INP5 from PA5 0110: COMP2_INP6 from PB1 0111: COMP2_INP7 from PB2 1000: COMP2_INP8 from PB10 1001: COMP2_INP9 output from OPA2 1010: COMP2_INP10 from PB13 1011: COMP2_INP11 from PB14 1100: COMP2_INP12 from PB4 1101: COMP2_INP13 from PB6 1110: COMP2_INP14 from PB7 1111: COMP2_INP15 from DAC2_VIN
5: 2	INNSEL [3:0]	RW	0	Signal selection for COMP2 non-inverting inputs 0000: COMP2_INM0 from PC0 0001: COMP2_INM1 from PC1 0010: COMP2_INM2 from PC2 0011: COMP2_INM3 from PC3 0100: COMP2_INM4 from PA0 0101: COMP2_INM5 from PA1

Bit	Name	R/W	Reset Value	Function
				0110: COMP1_INM6 from PB0 0111: COMP1_INM7 from PB1 1000: COMP2_INM8 from PB2 1001: COMP2_INM9 from PB3 1010: COMP2_INM10 from DAC2_VIN 1011: COMP2_INM11 from VREFCMP 1100: COMP2_INM12 from TS_VIN (temperature sensor voltage) 1101: COMP2_INM13 from VREFINT (required to enable the ADC_CR2.TSVREFE register of the ADC module) 1110: COMP2_INM14 from VREFBUF 1111: COMP2_INM15 from PB12
1	COMP2_HYST	RW	0	COMP2 hysteresis function enable control 0: no hysteresis 1: Hysteresis voltage approx. 20 mV
0	COMP2_EN	RW	0	COMP2 enable bit Software readable and writable 0: not enabled 1: Enabling

18.4.4. COMP2 Filter Register (COMP2_FR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FLTCNT2[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
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	FLTEN2
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	RW

Bit	Name	R/W	Reset Value	Function
31: 16	FLTCNT2[15: 0]	RW	0	Comparator 2 Sample Filter Counter The sampling clock is APB or LSI or LSE. The filter count value is configurable. The results are consistently output when the number of samples reaches the filter count value. Sample count period = FLTCNT[15:0]
15: 1	Res	-	-	Res

0	FLTEN2	RW	0	Comparator 2 Digital Filter Function Configuration 0: Disable digital filter function 1: Enable digital filter function Note: This bit must be set when COMP2_EN is 0
---	--------	----	---	--

18.4.5. COMP3 Control and Status Register (COMP3_CSR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	COMP_OUT	Res										PWRMODE [1:0]		Res	
-	R	-										RW		-	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
POLARITY	Res					INPSEL [3:0]				INNSEL [3:0]				COMP3_HYST	COMP3_EN
RW	-					RW				RW				RW	RW

Bit	Name	R/W	Reset Value	Function
31	Res	-	-	Res
30	COMP_OUT	R	0	COMP3 Output Status This bit is read-only and reflects the COMP3 output level after polarity selection.
29: 20	Res	-	-	Res
19: 18	PWRMODE [1:0]	RW	0	COMP3 power consumption mode selection Selected power consumption and consequent speed of COMP3 00: High-speed mode 01: Medium speed mode 10: Res 11: Res
17: 16	Res	-	-	Res
15	POLARITY	RW	0	COMP3 Output Polarity Selection Software readable and writable 0: No inversion 1: Inverted phase
14: 10	Res	-	-	Res
9: 6	INPSEL [3:0]	RW	0	Signal selection for COMP3 non-inverting inputs, software readable and writable 0000: COMP3_INP0 from PA5 0001: COMP3_INP1 from PB1

Bit	Name	R/W	Reset Value	Function
				0010: COMP3_INP2 from PB11 0011: COMP3_INP3 from PB14 0100: COMP3_INP4 from OPA3 output 0101: COMP3_INP5 from DAC1_VIN 0110: COMP3_INP6 from DAC2_VIN Other: Reserved
5: 2	INNSEL [3:0]	RW	0	Signal selection for COMP3 non-inverting inputs 0000: COMP3_INM0 from PA5 0001: COMP3_INM1 from PB1 0010: COMP3_INM2 from PB11 0011: COMP3_INM3 from PB14 0100: COMP3_INM4 from PC7 0101: COMP3_INM5 from DAC1_VIN 0110: COMP3_INM6 from DAC2_VIN 0111: COMP3_INM7 from TS_VIN (temperature sensor voltage) 1000: COMP3_INM8 from VREFINT (required to enable the ADC_CR2.TSVREFE register of the ADC module) 1001: COMP3_INM9 from VREFBUF 1010: COMP3_INM10 from VREFCMP Other: Reserved
1	COMP3_HYST	RW	0	COMP3 hysteresis function enable control 0: no hysteresis 1: Hysteresis voltage approx. 20 mV
0	COMP3_EN	RW	0	COMP3 enable bit Software readable and writable 0: not enabled 1: Enabling

18.4.6. COMP3 Filter Register (COMP3_FR)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FLTCNT3[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
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	FLTEN3
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	RW

Bit	Name	R/W	Reset Value	Function
31: 16	FLTCNT3[15: 0]	RW	0	<p>Comparator 3 Sample Filter Counter</p> <p>The sampling clock is APB or LSI or LSE. The filter count value is configurable. The results are consistently output when the number of samples reaches the filter count value.</p> <p>Sample count period = FLTCNT[15:0]</p>
15: 1	Res	-	-	Res
0	FLTEN3	RW	0	<p>Comparator 3 Digital Filter Function Configuration</p> <p>0: Disable digital filter function</p> <p>1: Enable digital filter function</p> <p>Note: This bit must be set when COMP3_EN is 0</p>

19. Operational amplifiers (OPA)

19.1. OPA Introduction

OPA modules are suitable for simple amplifier applications. The three internal op-amps can be cascaded using external resistors. The OPA input range is $0 \sim V_{CCA}$ and the output range is $0.1 \sim V_{CCA} - 0.2 \text{ V}$. The OPA input range is $0 \sim V_{CCA}$ and the output range is $0.1 \sim V_{CCA} - 0.2 \text{ V}$.

19.2. OPA Main Characteristics

- 3 independently configurable op amps
- The OPA has an input range of 0 to V_{CCA} and an output range of 0.1 to $V_{CCA} - 0.2 \text{ V}$ programmable gain.
- Can be configured in the following modes
 - Universal Op Amp Mode

19.3. OPA Functional Description

The 3 OPAs can amplify small signal analog input signals by using external components to form an amplifier, and the output is an amplified signal.

19.3.1. OPA Block Diagram

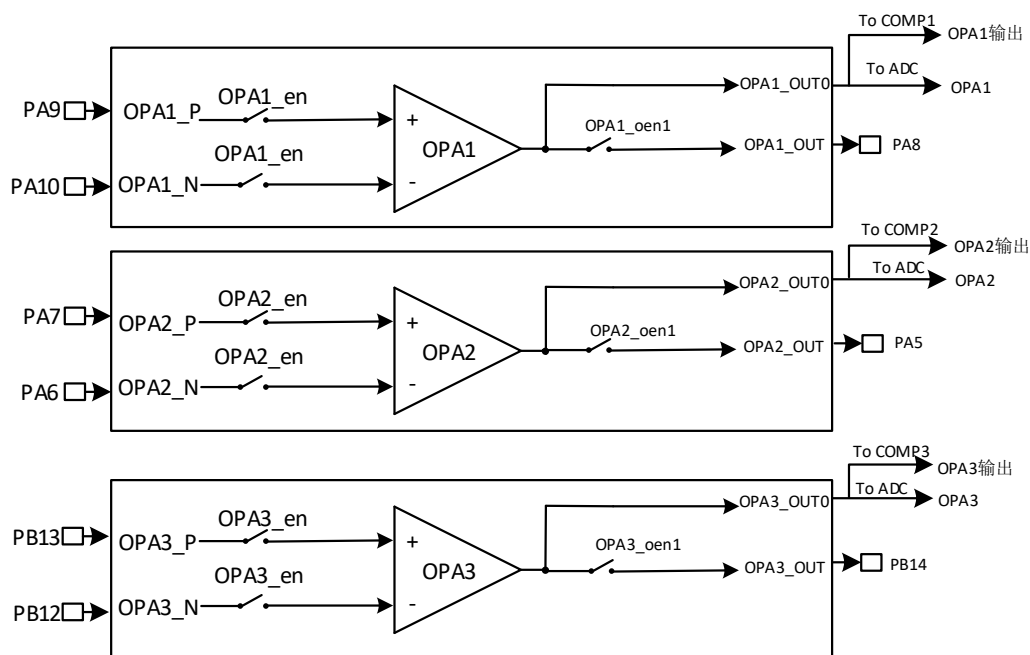


Figure 19-1 OPA Architecture Block Diagram

19.4. OPA Register

19.4.1. OPA output enable register (OPA_CR0)

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res				OP3OEN1	Res				OP2OEN1	Res				OP1OEN1	Res
-				RW	-				RW	-				RW	-

Bit	Name	R/W	Reset Value	Function
31: 12	Res	-	-	Res
11	OPA3OEN1	RW	0	OPA3 output 1 enable
10: 7	Res	-	-	Res
6	OPA2OEN1	RW	0	OPA2 output 1 enable
5: 2	Res	-	-	Res
1	OPA1OEN1	RW	0	OPA1 output 1 enable
0	Res	-	-	Res

19.4.2. OPA Control Register (OPA_CR1)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res								EN3	EN2	EN1	Res				
-								RW	RW	RW	-				

Bit	Name	R/W	Reset Value	Function
31: 8	Res	-	-	Res
7	EN3	RW	0	OPA3 Enable
6	EN2	RW	0	OPA2 Enable
5	EN1	RW	0	OPA1 Enable
4: 0	Res	-	-	Res

20. Hardware divider (DIV)

20.1. DIV Introduction

DIV (Divider) is a 32-bit signed/unsigned integer hardware divider.

20.2. DIV Key Features

- Supports 32-bit division
- The data in the registers cannot be changed while the current division is not running.
- Configurable signed/unsigned integer division calculations
- 32-bit Dividend, 32-bit Divisor
- Output 32-bit quotient and 32-bit remainder
- Division by zero warning flag bit, end of division flag bit
- 8 clock cycles to complete a division operation
- Write Divisor Register Triggers Division Circuitry
- After writing the divisor, you need to wait for the computation completion flag DIV_END before reading the quotient register and the remainder register.
- When the divisor is 0, the quotient and remainder result in 0

20.3. DIV Function Description

20.3.1. DIV operation flow

1. Turn on the module clock enable bit for the hardware divider in the RCC system clock controller.
2. Configuration register DIV_SIGN sets the signed/unsigned division operation.
3. Configuration register DIV_DEND sets the divisor.
4. Configuration register DIV_SOR sets the divisor and the division operation starts.
5. Query the end-of-operation flag bit DIV_END in the DIV_STAT register; a DIV_END of 1 signals the end of the operation. Read register DIV_QUOT to get the quotient and read register HDIV_REMD to get the remainder.
6. When the divisor is zero, the division operation ends immediately, the quotient and remainder are both set to zero, and the divisor-zero warning flag DIV_ZERO is set.
7. When reading register DIV_QUOT/DIV_REMD before the division operation is finished, the CPU will read out the value of the last calculation.

Example: Calculating an unsigned division with a dividend of 1917887483 (0x7250A3FB) and a divisor of 9597 (0x257D)

Step 1, configure register DIV_SIGN to 0, i.e., unsigned division operation

Step 2, configure register DIV_DEND to 0x7250A3FB, i.e., set divisor

Step 3, configure register DIV_SOR to 0x257D, i.e., set the divisor and the calculation begins

Step 4, query register DIV_STAT End-of-operation flag bit DIV_END, DIV_END is 1 flag.

End of operations.

Read register DIV_QUOT get merchant 199842 (0x30CA2)

Read register DIV_REMD to get remainder 3809 (0xEE1)

20.4. DIV register

20.4.1. DIV Divisor Register (DIV_DEND)

Address offset: 0x00

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 0	DIV_DEND	RW	0	dividend

20.4.2. DIV Divisor Register (DIV_SOR)

Address offset: 0x04

Reset value: 0x0000 0001

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

Bit	Name	R/W	Reset Value	Function
31: 0	DIV_SOR	RW	1	Divisor (writing this register automatically triggers the division operation)

20.4.3. DIV Business Register (DIV_QUOT)

Address offset: 0x08

Reset value: 0x0000 0000

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

RW

Bit	Name	R/W	Reset Value	Function
31: 0	DIV_QUOT	RW	0	Store the quotient calculated by the divider

20.4.4. DIV Remainder Register (DIV_REMD)

Address offset: 0x0C

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 0	DIV_REMD	RW	0	Store the remainder calculated by the divider

20.4.5. DIV symbol register (DIV_SIGN)

Address offset: 0x10

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 1	Res	-	-	Res
0	SIGN	RW	0	Symbol Selection 0: Unsigned division operation 1: Signed division operations

20.4.6. DIV Status Register (DIV_STAT)

Address offset: 0x14

Reset value: 0x0000 0000

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

-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res														DIV_ZERO	DIV_END
-														RW	RW

Bit	Name	R/W	Reset Value	Function
31: 2	Res	-	-	Res
1	DIV_ZERO	R	0	Divisor Zero Warning Flag Bit 0: divisor not zero 1: Division by zero
0	DIV_END	R	0	Division end flag 0: operation in progress 1: End of operation

21. Advanced Control Timer (TIM1)

21.1. TIM1 Introduction

The advanced timer (TIM1) consists of a 16-bit auto-reload counter driven by a programmable prescaler. It can be used in a variety of scenarios including: pulse length measurement of an input signal (input capture), or generating an output waveform (output comparison, output PWM, complementary PWM with deadband insertion).

Pulse length and waveform period can be modulated from microseconds to milliseconds using a timer divider and an RCC clock control divider. The advanced (TIM1) and general purpose (TIMx) timers are completely independent and do not share any resources. They can operate synchronously.

21.2. TIM1 Main Features

- 16-bit up, down or up-down auto-reload counter
- 16-bit programmable prescaler that can be modified in real-time, allowing the counter clock frequency to be divided from 1 to 65536
- Up to 4 independent channels
 - Input Capture
 - Output Comparison
 - PWM generation (edge or center aligned mode)
 - Single pulse mode output
- Dead-time programmable complementary outputs
- Synchronization circuits using external signals to control timers and timer interconnections
- Repeat counter that counts a specified number of cycles before updating the timer's registers
- The brake input can place the timer's output signal in a user-selectable safety configuration
- Interrupts/DMA are generated on the following events
 - Updates: Counter overflow up, counter overflow down, counter initialization (triggered by software or internally and externally)
 - trigger event
 - Input Capture
 - Output Comparison
 - Brake Input
- Supports incremental (quadrature) encoders and Hall sensor circuits for positioning.
- Trigger input as external clock or per-cycle current management

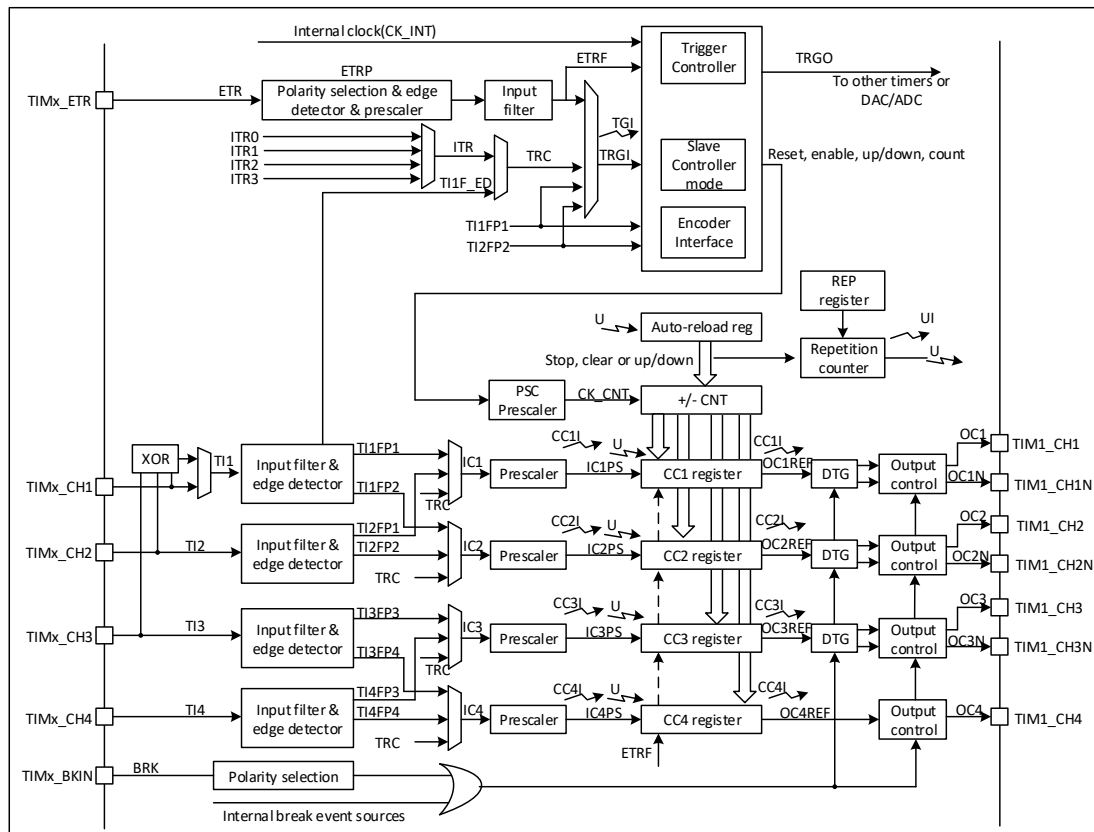


Figure 21-1 Advanced Control Timer Architecture Block Diagram

21.3. TIM1 Function Description

21.3.1. Time base unit (in computing)

The main part of the programmable advanced control timer is a 16-bit counter and its associated auto-reload register. This counter can count up, count down or count up and down in both directions. This counter clock is divided by a prescaler.

The counter, auto-reload registers, and prescaler registers can be read and written by software, and reads and writes remain valid even if the counter is still running.

The time base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Automatic Reload Register (TIMx_ARR)
- Repeat Count Register (TIMx_RCR)

The auto-reload registers are preloaded, and writing or reading the auto-reload registers will access the preloaded registers. Depending on the setting of the auto-reload preload enable bit (ARPE) in the TIMx_CR1 register, the contents of the preload register are transferred to the shadow register either immediately or at each update event UEV. The update event is generated when the counter reaches upflow (downflow condition in case of down counter) and when the UDIS bit in the TIMx_CR1 register is equal to zero. Update events can also be generated by software.

The counter is driven by the clock output of the prescaler, CK_CNT, which is valid only when the counter enable bit (CEN) in the counter TIMx_CR1 register is set.

Note that the counter starts counting one clock cycle after the CEN bit of the TIMx_CR register is set.

Prescaler Description

The prescaler divides the counter clock by any value between 1 and 65536. It is a 16-bit counter based on 16-bit register control (in the TIMx_PSC register). Because this control register has a buffer, it can be changed at runtime. The parameters of the new prescaler are adopted on the arrival of the next update event.

Figures 21-2 and 21-3 give examples of changing counter parameters while the prescaler is running.

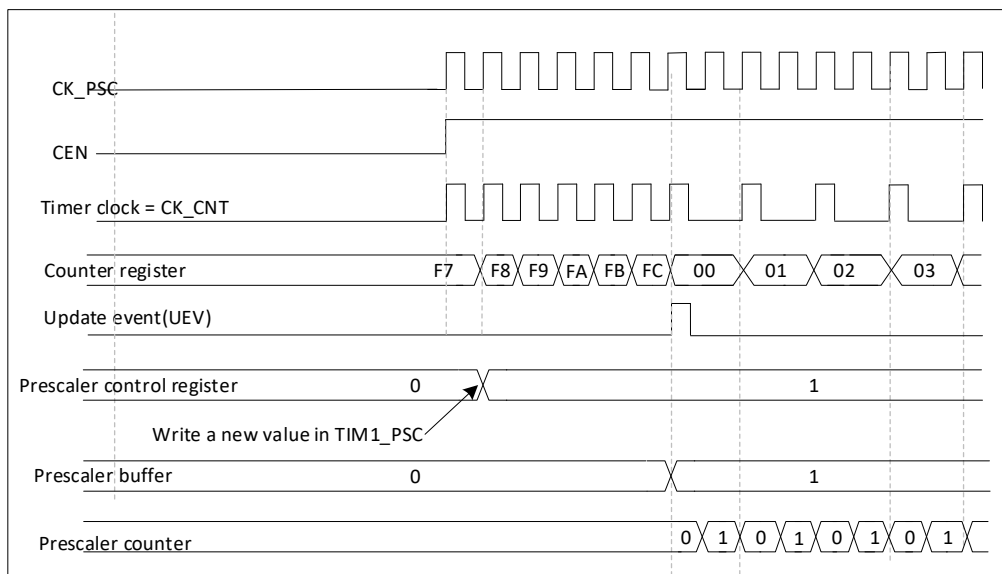


Figure 21-2 Timing diagram of the counter when the prescaler parameter is changed from 1 to 2

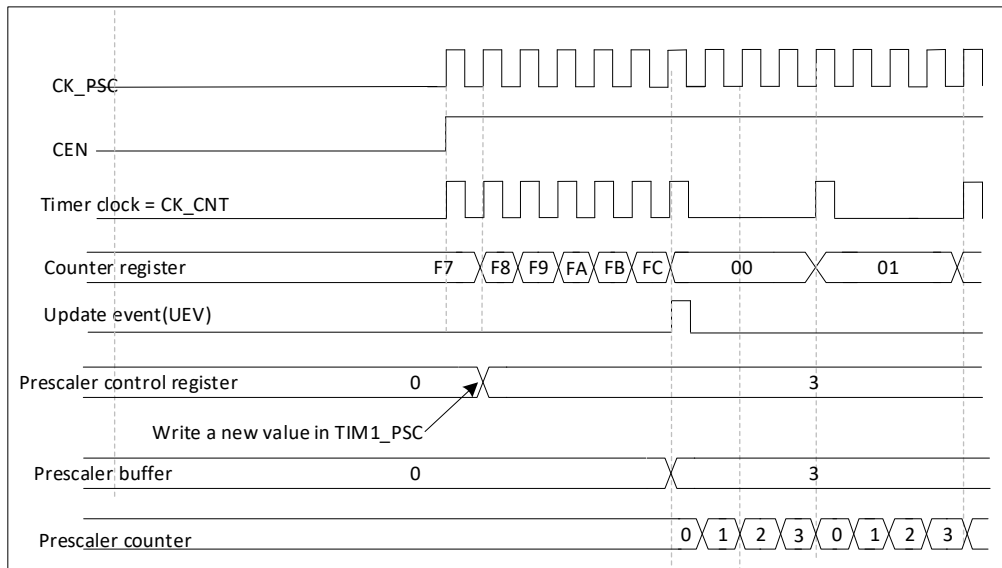


Figure 21-3 Timing diagram of the counter when the prescaler parameter is changed from 1 to 4

21.3.2. Counter Mode

Up Count Mode

Upward counting mode, which is a counter that goes from 0 to an auto-reloaded value, then restarts counting from 0 again and generates a count overflow event.

If the Repeat Counter is used, an Update Event (UEV) is generated after the Up Counter has repeated the number of times set in the Repeat Counter Register (TIMx_RCR), or else an Update Event is generated each time the counter overflows.

Setting the UG bit in the TIMx_EGR register (either by software or using a slave mode controller) can similarly generate an update event.

Setting the UDIS bit in the TIMx_CR1 register disables the update event; this also prevents the shadow register from being updated when a new value is written to the preloaded register. No update event will be generated until the UDIS bit is cleared. Even so, the counter is still cleared '0' when an update event should be generated, and the prescaler count is also invited to 0 (but the prescaler value remains unchanged). In addition, if the URS bit (select update request) in the TIMx_CR1 register is set, setting the UG bit will generate an update event UEV, but the hardware does not set the UIF flag (i.e., no interrupt or DMA request is generated). This is to avoid generating both update and capture interrupts when clearing the counter in capture mode.

When an update event occurs, all registers are updated and the hardware simultaneously (based on the URS bit) sets the update flag bit (UIF bit in the TIMx_SR register).

- The repeat counter is reloaded as the contents of the TIMx_RCR register.
- The auto-reload shadow register is reset to the value of the preload register (TIMx_ARR).
- The prescaler buffer is set to the value of the preload register (the contents of the TIMx_PSC register).

The following figure gives some examples of how the counter acts at different clock frequencies when TIMx_ARR=0x36.

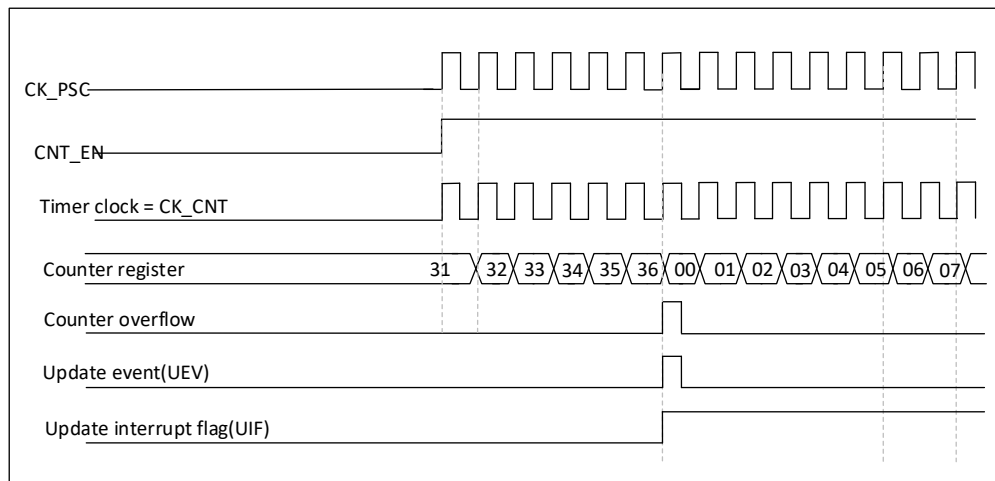


Figure 21-4 Timing diagram of the counter with internal clock division factor of 1

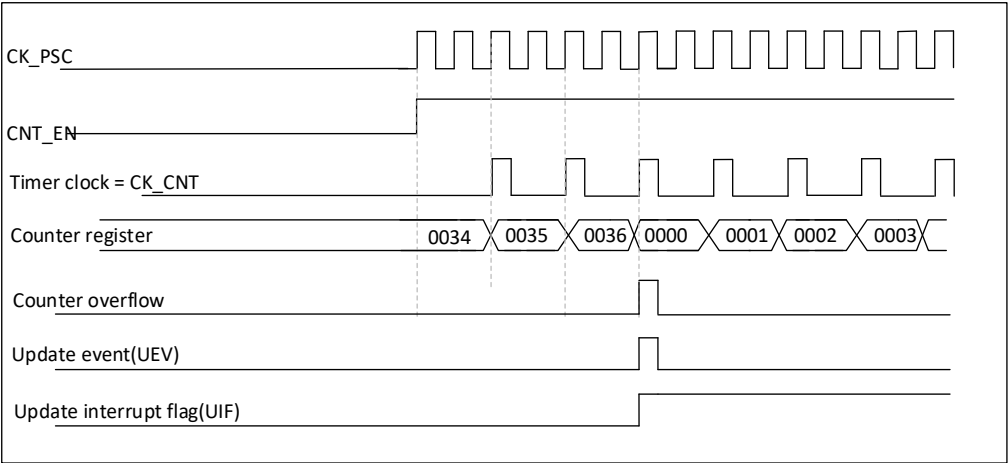


Figure 21-5 Timing diagram of the counter with internal clock divider factor of 2

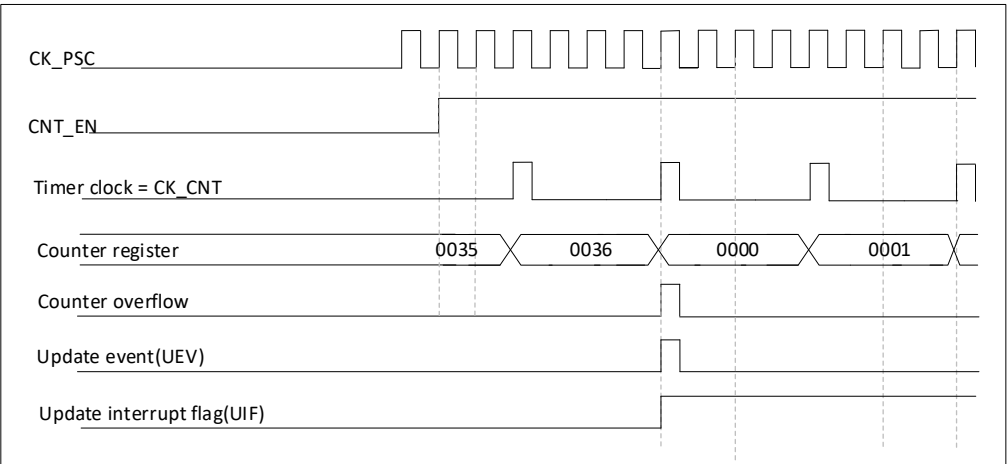


Figure 21-6 Timing diagram of the counter with internal clock divider factor of 4

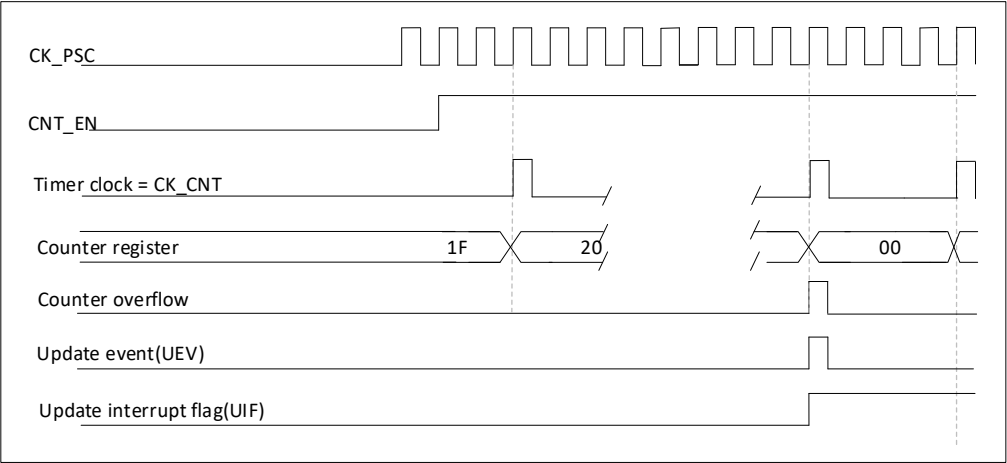


Figure 21-7 Timing diagram of the counter with internal clock dividing factor N

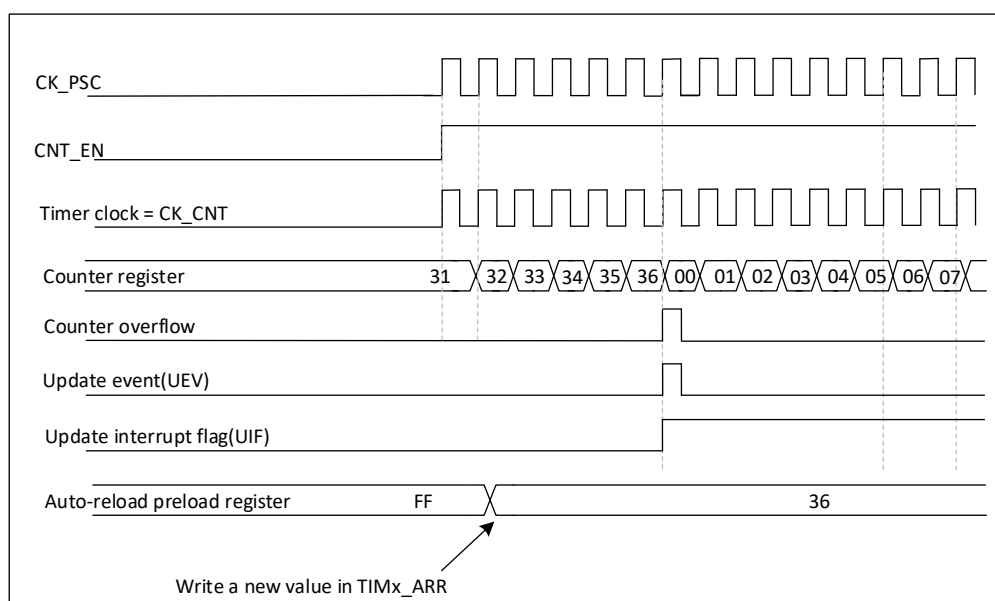


Figure 21-8 Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)

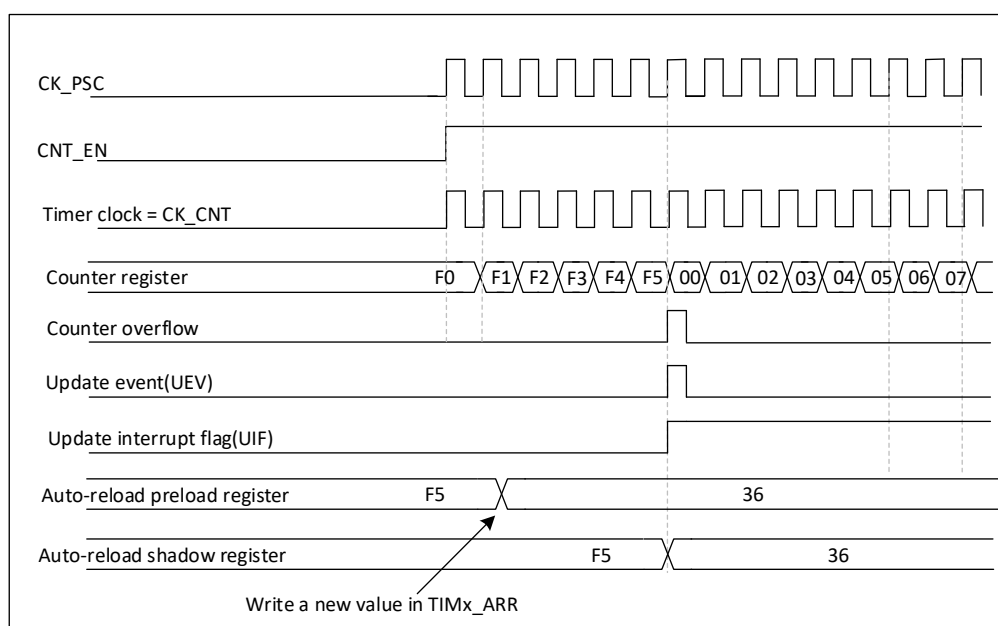


Figure 21-9 Counter timing diagram, update event when ARPE=1 (preloaded with TIMx_ARR)

Down Count Mode

Downward counting mode, which starts counting down to 0 from the auto-reloaded value, then re-starts counting down from the auto-reloaded value and generates a downward overflow event.

If a repeat counter is used, an update event (UEV) will be generated when the down count repeats the number of times set in the Repeat Counter Register (TIMx_RCR), otherwise the update event is generated each time the counter overflows.

Setting the UG bit in the TIMx_EGR register (either by software or using a slave mode controller) similarly generates an update event.

Setting the UDIS bit of the TIMx_CR1 register disables UEV events. This avoids updating the shadow registers when writing new values to the preloaded registers. Therefore no update event is

generated until the UDIS bit is cleared to zero. However, the counter still restarts counting from the current auto-load value and the prescaler's counter restarts from 0 (but the prescaler factor remains unchanged).

In addition, if the URS bit in the TIMx_CR1 register is set (selecting update requests), setting the UG bit will generate an update event UEV but not set the UIF flag (and therefore not generate interrupts and DMA requests), this is to avoid generating both update and capture interrupts when a capture event occurs and clears the counter.

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

- The repeat counter is reset to the contents of the TIMx_RCR register
- The prescaler's buffer is loaded with the preloaded value (the value of the TIMx_PSC register)
- The current auto-reload register is updated to the preloaded value (the contents of the TIMx_ARR register)

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

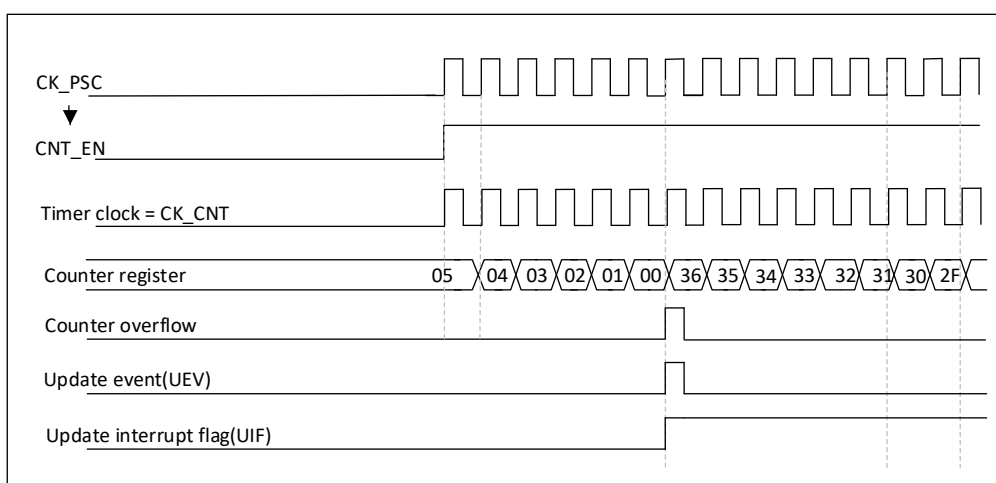


Figure 21-10 Timing diagram of the counter with internal clock division factor of 1

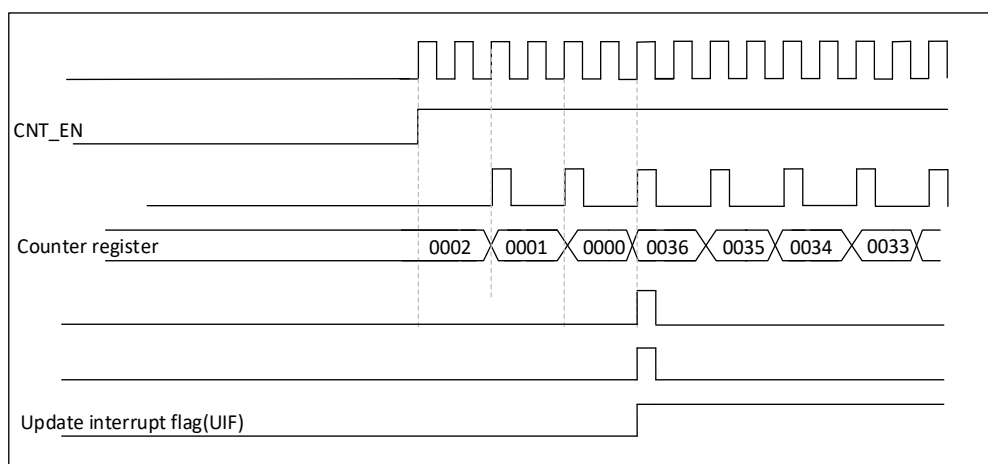


Figure 21-11 Timing diagram of the counter with internal clock divider factor of 2

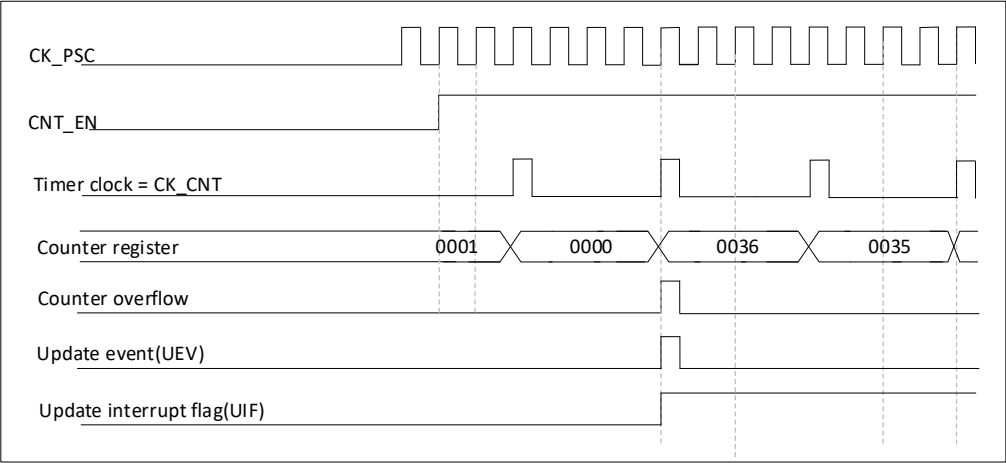


Figure 21-12 Timing diagram of the counter with internal clock divider factor of 4

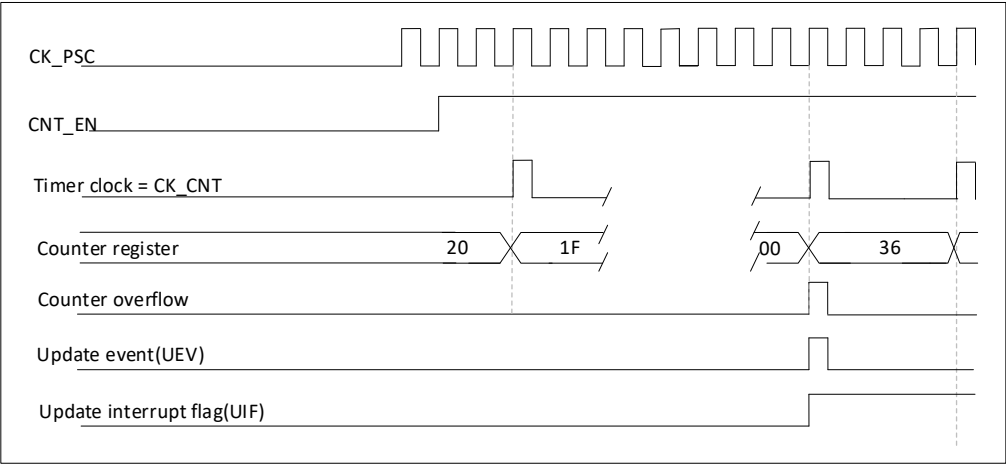


Figure 21-13 Timing diagram of the counter with internal clock dividing factor N

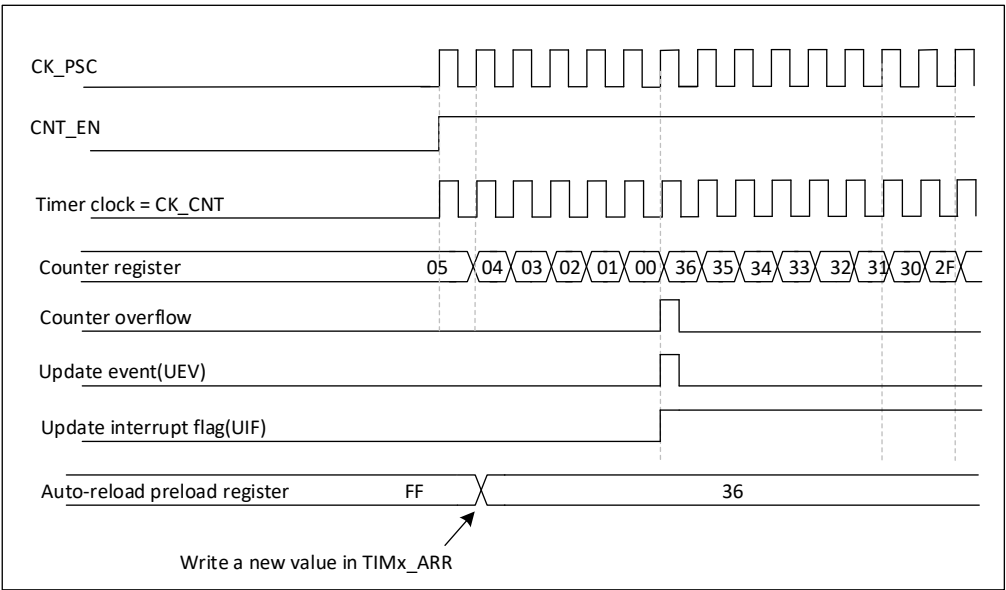


Figure 21-14 Counter Timing Diagram, Update Events When Cycle Counter is Not Used

Central alignment mode (counting up/down)

In center-aligned mode, the counter counts from 0 to the auto-loaded value (TIMx_ARR register) -1, generates a counter overflow event, then counts down to 1 and generates a counter underflow event; then counts from 0 again.

Central alignment mode is valid when CMS in the TIMx_CR1 register is not equal to zero. The output compare interrupt flag will be set when the channel is configured in output mode when: counting down (central alignment mode 1, CMS="01"), counting up (central alignment mode 2, CMS="10") counting up counting down (Central Alignment Mode 3, CMS="11").

In this mode, the DIR direction bit in TIMx_CR1 cannot be written. It is updated by hardware and indicates the current count direction.

Update events can be generated on every count overflow and every count underflow; they can also be generated by setting (in software or using a slave mode controller) the UG bit in the TIMx_EGR register. The counter then resumes counting from 0, and the prescaler resumes counting from 0 as well.

Setting the UDIS bit in the TIMx_CR1 register disables UEV events. This avoids updating the shadow registers when writing new values to the preloaded registers. Therefore no update event is generated until the UDIS bit is cleared to zero. However, the counter will still continue to count up or down depending on the current auto-reload value.

In addition, if the URS bit in the TIMx_CR1 register is set (selecting an update request), setting the UG bit will generate an update event UEV but not set the UIF flag (and therefore not generate an interrupt and a DMA request), this is to avoid generating an update and capture interrupt at the same time when a capture event occurs and clears the counter.

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

- The repeat counter is reset to the contents of the TIMx_RCR register
- The prescaler buffer is loaded with the value of the preload (TIMx_PSC register).
- The current auto-reload register is updated to the preloaded value (within the TIMx_ARR register)

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

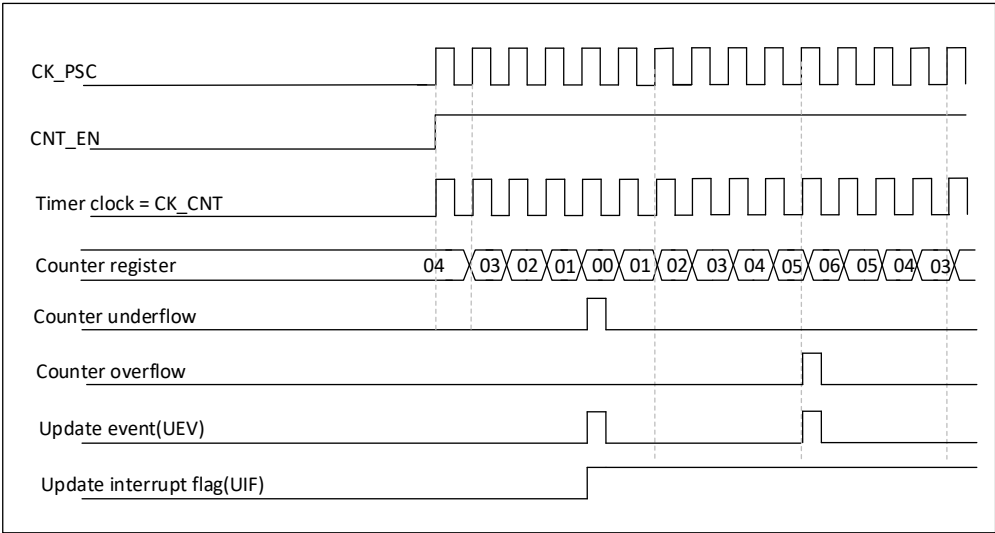


Figure 21-15 Counter timing diagram with internal clock divider factor of 1 and $TIMx_ARR = 0x6$

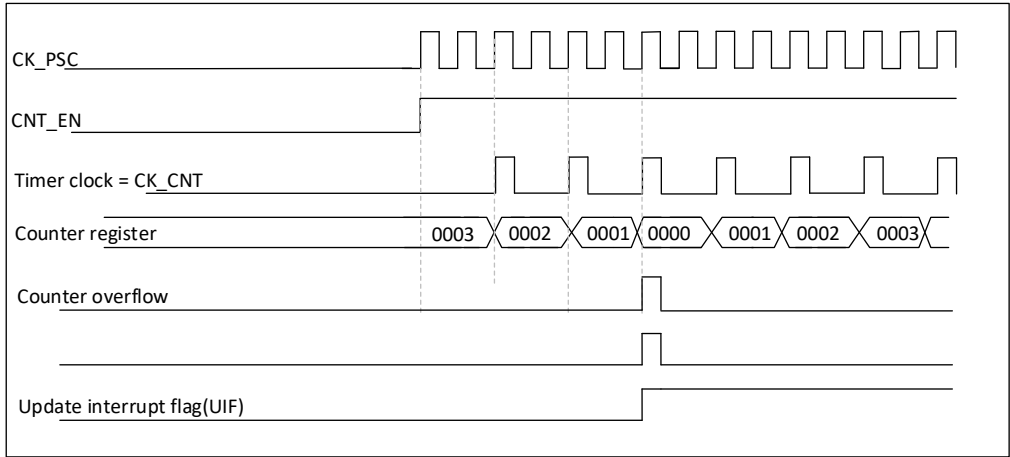


Figure 21-16 Counter timing diagram with internal clock divider factor of 2 and $TIMx_ARR=0x36$

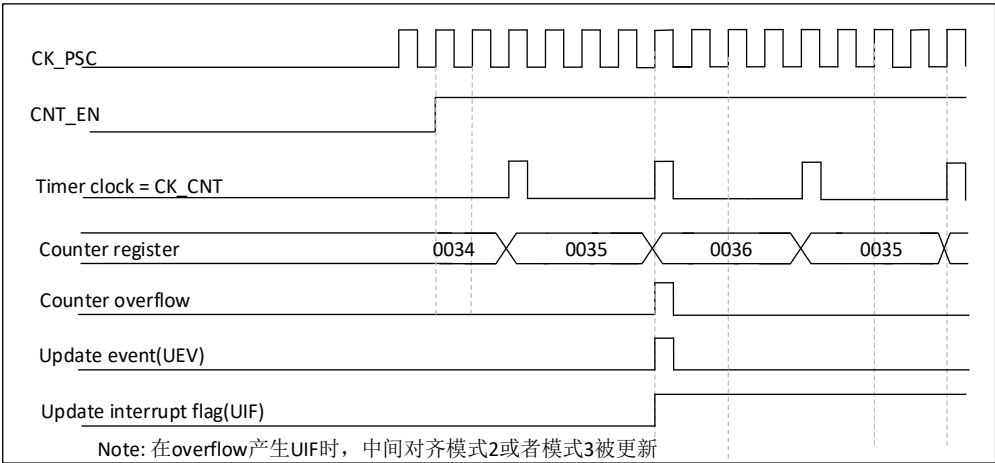


Figure 21-17 Counter timing diagram with internal clock divider factor of 4 and $TIMx_ARR=0x36$

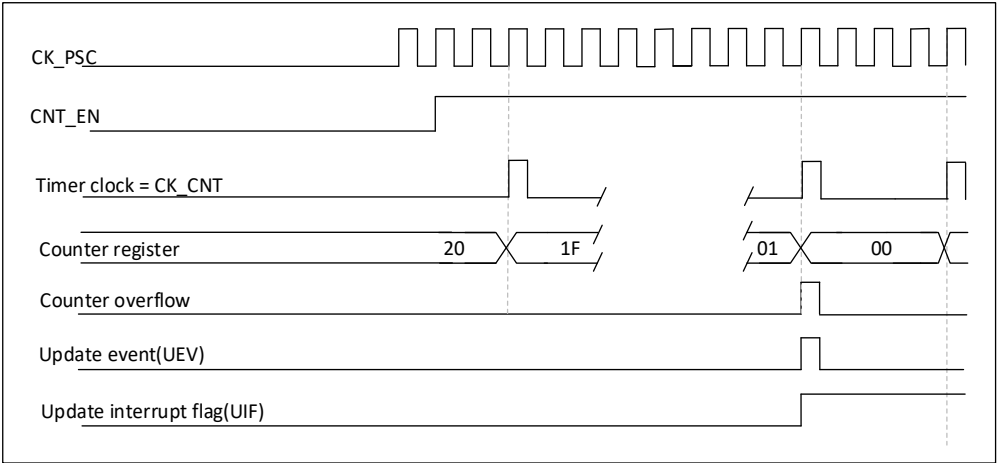


Figure 21-18 Timing diagram of the counter with internal clock dividing factor N

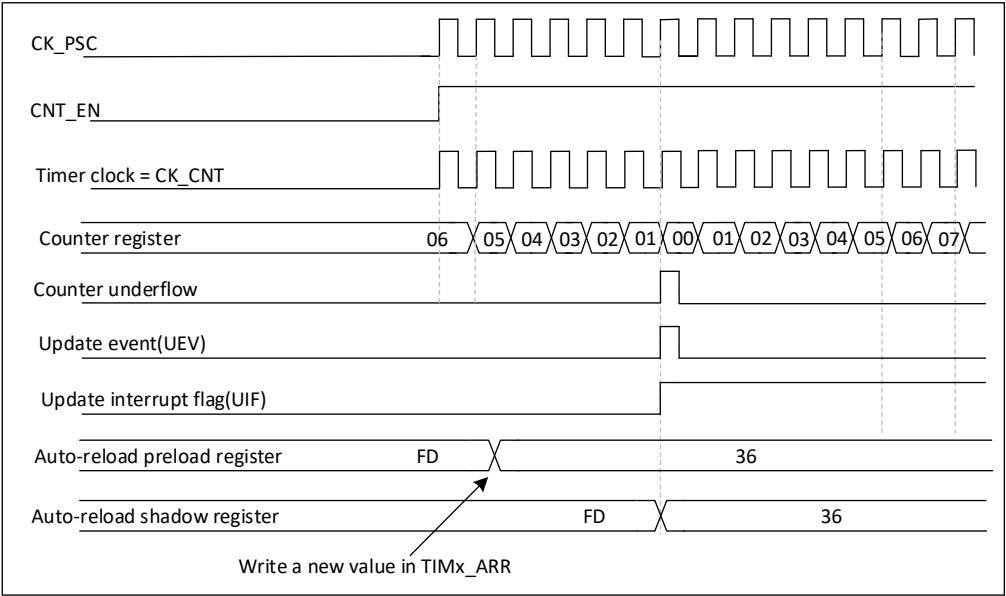


Figure 21-19 Counter timing diagram, update event at ARPE=1 (counter underflow)

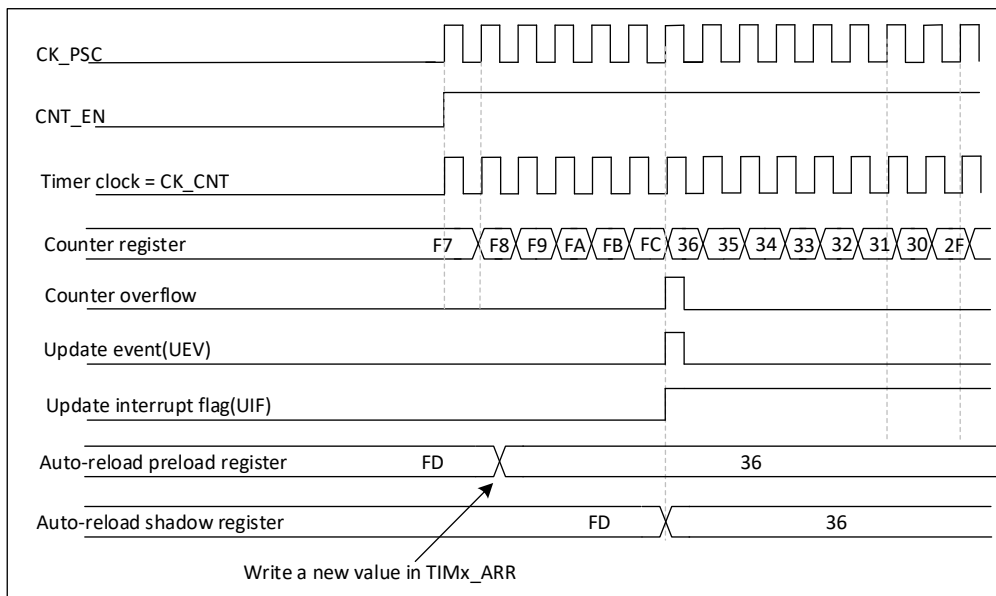


Figure 21-20 Counter timing diagram, update event at ARPE=1 (counter overflow)

21.3.3. Repetition counter

The time base unit describes how update events (UEVs) are generated regarding counter upward and downward overflow. It is actually generated only when the repeat counter counts to zero. This feature is useful for generating PWM signals.

This means that data is transferred from the preload registers to the shadow registers (TIMx_ARR auto-reload register, TIMx_PSC preload register, and also the capture/compare register TIMx_CCRx in compare mode) every N+1 counts of overflows or underflows, with N being the value in the TIMx_RCR repeat count register.

The repeat counter decreases when any of the following conditions hold:

- Each time the counter overflows in count-up mode
- Every time the count overflows in down count mode
- Central alignment mode at each overflow and at each underflow.

In the center-aligned mode, it is able to update the duty cycle 2 times per PWM cycle, although this limits the PWM to a maximum cycle period of 128 bits. In center-aligned mode, since the waveform is symmetrical, the maximum resolution is 2xTck if the compare register is only refreshed once in each PWM cycle.

The repeat counter is automatically loaded and the repeat rate is defined by the value of the TIMx_RCR register. When the update event is generated by software (by setting the UG bit in TIMx_EGR) or by the hardware slave mode controller, the update event occurs immediately regardless of the value of the repeat counter, and the contents of the TIMx_RCR register are reloaded as to the repeat counter.

In central alignment mode, an update event is generated for odd values of RCR depending on when the RCR register is written to and when the counter starts and there is an overflow, or underflow. If

the RCR is written before starting the counter, an update event is generated on overflow. For example, for $RCR = 3$, the update event is generated at the 4th overflow or underflow event (depending on the value that RCR was written to).

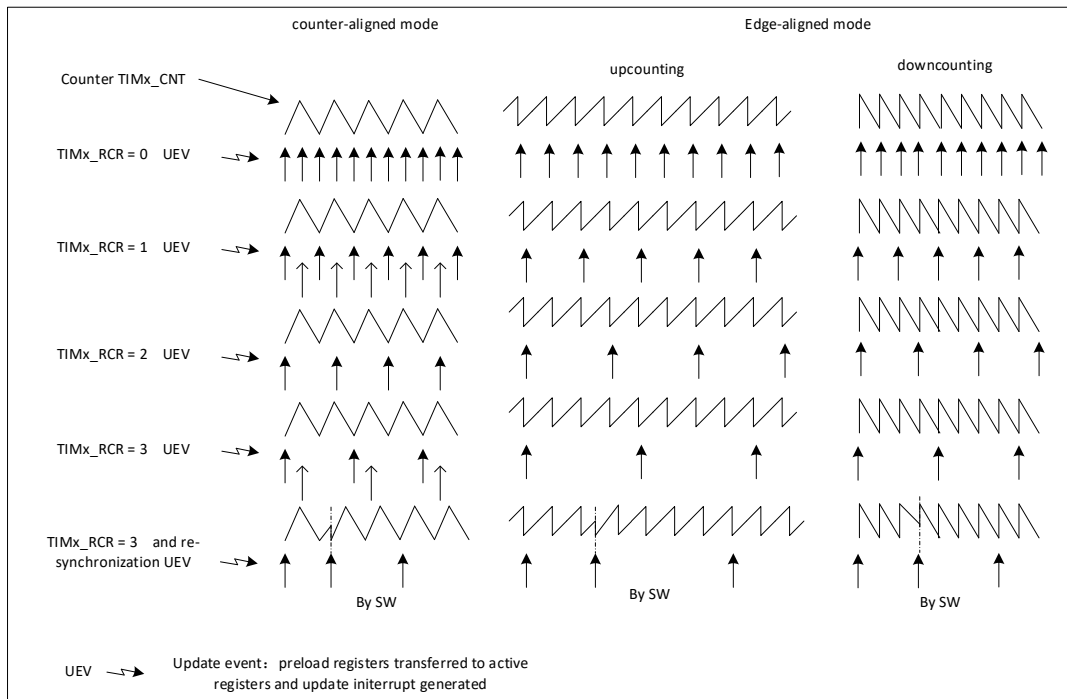


Figure 21-21 Example of update rate in different modes and register setting of TIMx_RCR

21.3.4. Clock source

The counter can be clocked by the following clock sources:

- Internal clock (CK_INT)
- External Clock Mode 1: External Input Pin
- External clock mode 2: External trigger input ETR
- Internal Trigger Input (ITRx): uses one timer as a prescaler for another timer. For example, one timer Timer1 can be configured as a prescaler for another timer Timer2.

Internal clock source (CK_INT)

If the slave mode controller is disabled, the CEN, DIR (TIMx_CR1 register) and UG bits (TIMx_EGR register) are de facto control bits and can only be modified by software. As long as the CEN bit is written to 1, the prescaler clock is provided by the internal clock CK_INT.

The following diagram shows the operation of the control circuit and up counter in general mode without prescaler

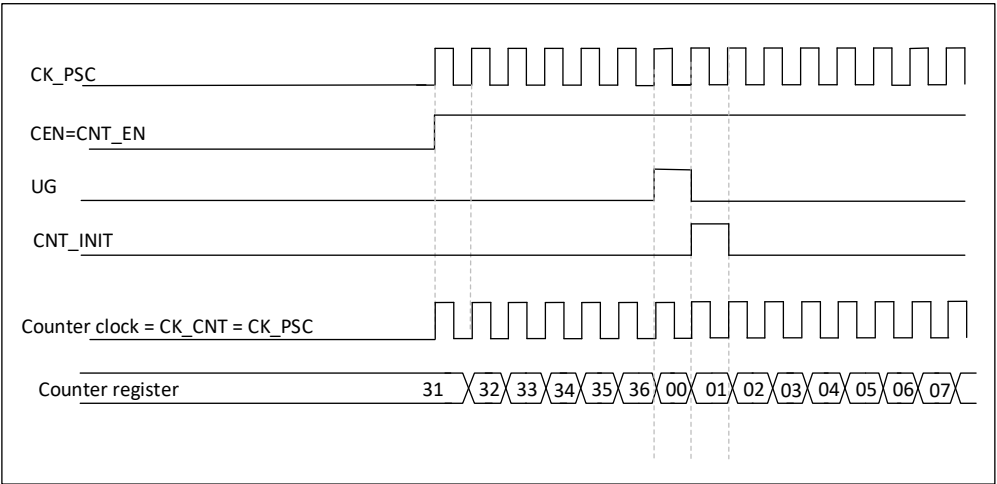


Figure 21-22 Control circuit in general mode with internal clock division factor of 1

External clock source mode 1

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

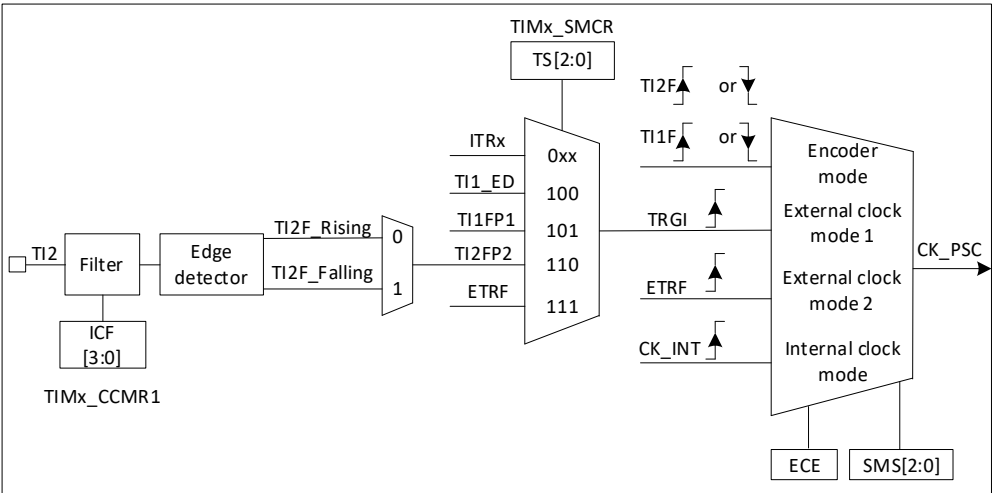


Figure 21-23 TI2 External Clock Connection Example

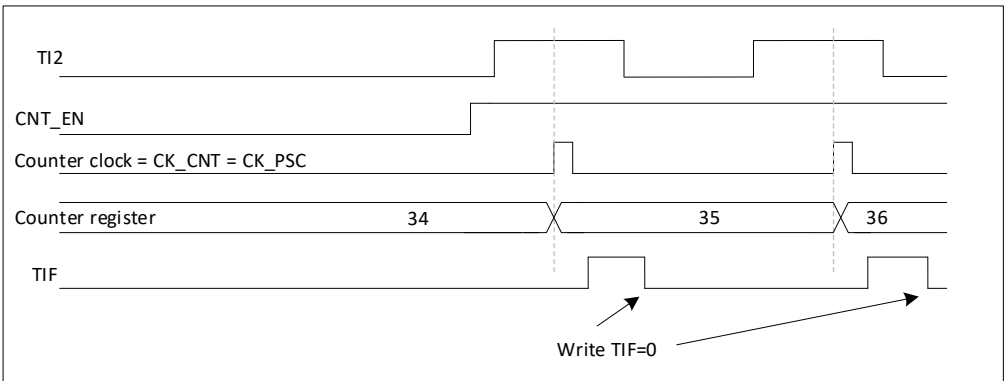


Figure 21-24 Control circuit in external clock mode 1

External clock source mode 2

This mode is selected by writing ECE of the TIMx_SMCR register to 1. The counter is capable of counting on every rising or falling edge of an externally triggered ETR.

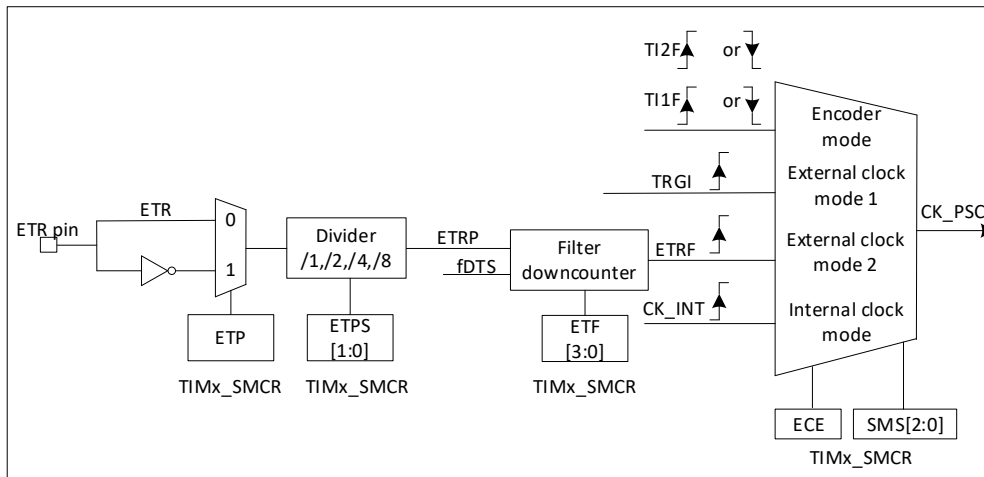


Figure 21-25 TI2 External Trigger Input Block Diagram

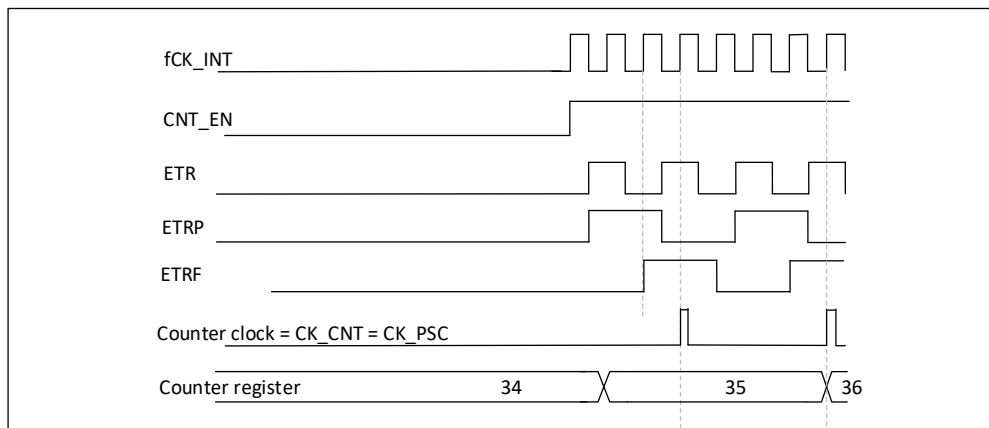


Figure 21-26 Control circuit in external clock mode 2

21.3.5. Capture/Compare Channel

Each capture/compare channel is organized around a capture/compare register (containing shadow registers), including the input portion of the capture (input filtering, multiplexing, and prescaler), and the output portion (comparator and output control).

The input section samples the corresponding TIx input signal and produces a filtered signal, TIxF. An edge monitor with polarity selection then generates a signal (TIxFPx) that can be used as an input trigger from the pattern controller or as a capture control. This signal enters the capture register (ICxPS) through pre-divided frequency.

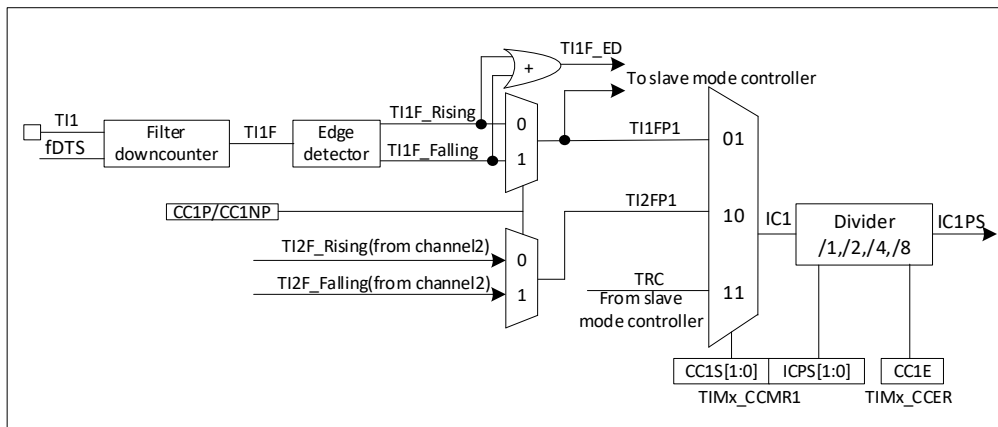


Figure 21-27 Capture/compare channels (e.g., channel 1 input section)

The output section generates an intermediate waveform OCxREF (highly active) as a reference, and the end of the chain determines the polarity of the final output signal.

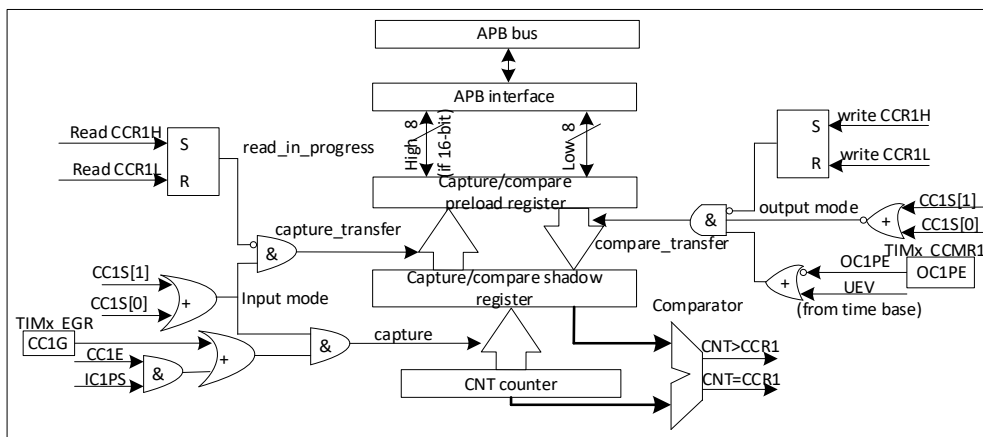


Figure 21-28 Capture/Compare Channel 1 Main Circuitry

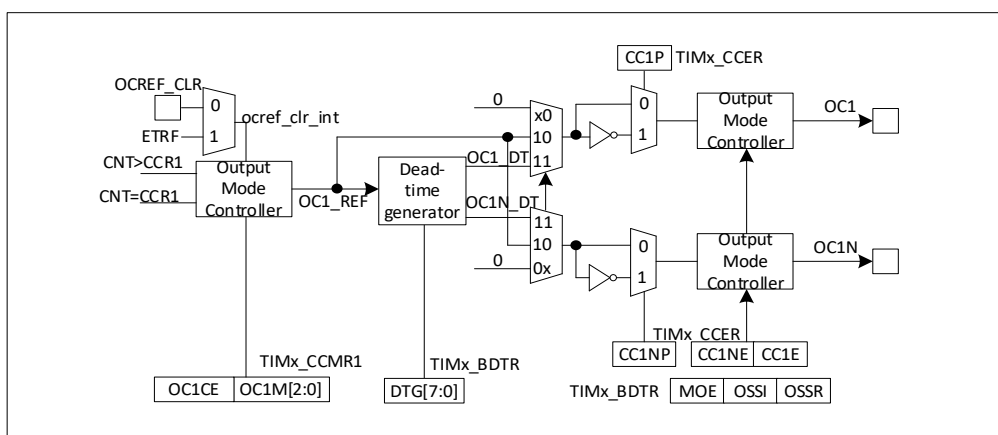


Figure 21-29 Output section of capture/compare channel (channels 1 to 3)

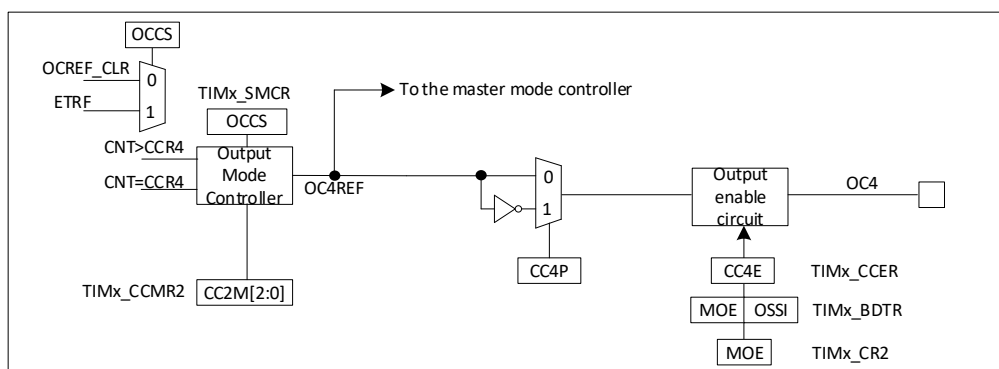


Figure 21-30 Output section of capture/compare channel (channel 4)

The capture/compare module consists of a preloaded register and a shadow register. The read/write process operates only on the preloaded registers.

In capture mode, the capture occurs on the shadow registers, which are then copied to the preloaded registers.

In comparison mode, the contents of the preloaded registers are copied to the shadow registers, and then the contents of the shadow registers are compared to the counter.

21.3.6. Input Capture Mode

In input capture mode, the current value of the counter is latched into the capture/compare register when the corresponding edge on the ICx signal is detected. When a capture event occurs, the corresponding CCxIF flag (TIMx_SR register) is set to 1, and an interrupt or DMA request will be generated if interrupt and DMA operations are turned on. If the CCxIF flag is already high when the capture event occurs, the repeat capture flag CCxOF (TIMx_SR register) is set to 1. Writing CCxIF=0 clears CCxIF, or reading the capture data stored in the TIMx_CCRx register also clears CCxIF. Writing CCxOF=0 clears CCxOF.

The following example shows how to capture the value of the counter into the TIMx_CCR1 register on the rising edge of the TI1 input as follows:

1. Select valid inputs: TIMx_CMR1 must be connected to the TI1 input, so write CC1S=01 to the TIMx_CMR1 register, as long as CC1S is not '00', the channel is configured as an input and the TIMx_CCR1 register becomes read-only.
2. Configure the input filter for the desired bandwidth based on the characteristics of the input signal (i.e., when the input is TIx, the input filter control bit is the ICxF bit in the TIMx_CMRx register). Assuming that the input signal is dithered for a maximum of 5 internal clock cycles, we have to configure the filter with a bandwidth longer than 5 clock cycles; we can therefore (at the fDTS frequency) sample 8 consecutive times in order to confirm a true edge shift on TI1, i.e., by writing IC1F=0011 in the TIMx_CMR1 register.
3. Select the active conversion edge of the TI1 channel by writing CC1P=0 (rising edge) in the TIMx_CCER register

4. Configure the input prescaler. In this example, we want the capture to occur at every valid level-transition moment, so the prescaler is disabled (write IC1PS=00 to the TIMx_CMR1 register).
5. Setting CC1E=1 in the TIMx_CCER register allows the value of the capture counter to be captured into the capture register.
6. If required, allow related interrupt requests by setting the CC1IE bit in the TIMx_DIER register and DMA requests by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:

- When a valid level transition is generated, the counter value is transferred to the TIMx_CCR1 register.
- The CC1IF flag is set (interrupt flag). When at least 2 consecutive captures occur without CC1IF having been cleared, CC1OF is also set to 1.
- If the CC1IE bit is set, an interrupt is generated.
- If the CC1DE bit is set, a DMA request is also generated.

In order to handle capture overflows, it is recommended that data be read before the capture overflow flag is read; this is to avoid losing capture overflow information that may be generated after the capture overflow flag is read and before the data is read.

Note: Setting the corresponding CCxG bit in the TIMx_EGR register allows you to generate input capture interrupts and/or DMA requests through software.

21.3.7. PWM Input Mode

This mode is a special case of the Input Capture mode and operates the same as the Input Capture mode except for the following differences:

- The two ICx signals are mapped to the same TIx input.
- These 2 ICx signals are edge valid, but of opposite polarity.
- One of the TIxFP signals is used as the trigger input signal, while the slave mode controller is configured in reset mode.

For example, when it is necessary to measure the length (TIMx_CCR1 register) and duty cycle (TIMx_CCR2 register) of a PWM signal input to TI1, proceed as follows (depending on the frequency of CK_INT and the value of the prescaler)

- To select the valid input for TIMx_CCR1: Set CC1S=01 of the TIMx_CMR1 register (TI1 selected).
- Select the active polarity of TI1FP1 (used to capture data into TIMx_CCR1 and clear the counter): set CC1P=0 (active on rising edge).
- To select the valid input for TIMx_CCR2: Set CC2S=10 in the TIMx_CMR1 register (TI1 selected).
- Select the active polarity of TI1FP2 (capture data to TIMx_CCR2): set CC2P=1 (active on falling edge).

- To select a valid trigger input signal: set TS=101 in the TIMx_SMCR register (select TI1FP1).
- Configure the slave mode controller for reset mode: set SMS=100 in TIMx_SMCR.
- Enable Capture: Set CC1E=1 and CC2E=1 in TIMx_CCER register.

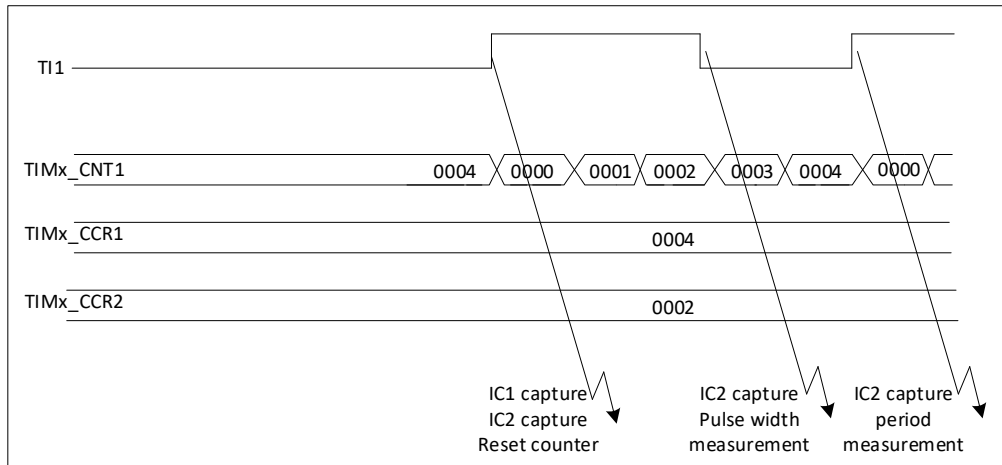


Figure 21-31 PWM Input Mode Timing

21.3.8. Forced Output Mode

In output mode (CCxS=00 in the TIMx_CMRx register), the output compare signals (OCxREF and the corresponding OCx/OCxN) can be directly forced by software to a valid or invalid state, independently of the result of the comparison between the Output Comparison Register and the counter. Setting the corresponding OCxM=101 in the TIMx_CMRx register will force the output compare signal (OCxREF/OCx) to be active. In this way OCxREF is forced high (OCxREF is always active high), while OCx gets a signal with the opposite polarity of CCxP.

For example, if CCxP=0 (OCx active high), then OCx is forced high. Setting OCxM=100 in the TIMx_CMRx register forces the OCxREF signal low.

In this mode, the comparison between the TIMx_CCRx shadow register and the counter is still in progress and the corresponding flags are modified. Therefore corresponding interrupts and DMA requests are still generated. This will be covered in the Output Comparison Mode section below.

21.3.9. Output Comparison Mode

This function is used to control an output waveform or to indicate that a given period of time has elapsed. When the contents of the counter and the capture/compare register are the same, the output compare function does the following:

- Outputs the values defined by the output comparison mode (OCxM bit in the TIMx_CMRx register) and output polarity (CCxP bit in the TIMx_CCER register) to the corresponding pins. When comparing matches, the output pin can hold its level (OCxM=000), be set to an active level (OCxM=001), be set to an inactive level (OCxM=010), or be flipped (OCxM=011).
- Set the flag bit in the interrupt status register (CCxIF bit in the TIMx_SR register).
- An interrupt is generated if the corresponding interrupt mask (CCxIE bit in the TIMx_DIER register) is set.

- If the corresponding enable bits are set (CCxDE bit in the TIMx_DIER register and CCDS bit in the TIMx_CR2 register selects the DMA request function), a DMA request is generated.

The OCxPE bit in TIMx_CMRx selects whether or not the TIMx_CCRx register requires the use of a preloaded register. In output comparison mode, the update event UEV has no effect on the OCxREF and OCx outputs.

The synchronization can be accurate up to one counting cycle of the counter. Output compare mode (in single pulse mode) can also be used to output a single pulse.

Outputs the configuration steps for the compare mode:

1. Select the counter clock (internal, external, prescaler).
2. Write the corresponding data to the TIMx_ARR and TIMx_CCRx registers.
3. If an interrupt request is to be generated, set the CCxIE bit.
4. Select the output mode, for example:
 - Require the counter to flip the output pin of OCx when it matches CCRx, set OCxM=011.

Set OCxPE = 0 to disable preloaded registers.

Set CCxP = 0 to select polarity as active high.

Set CCxE = 1 to enable the output.

5. Set the CEN bit of the TIMx_CR1 register to start the counter

The TIMx_CCRx registers can be updated by software at any time to control the output waveform, provided that the preloaded registers are not used (OCxPE='0', otherwise the shadow registers of TIMx_CCRx can only be updated when the next update event occurs). An example is given in the figure below.

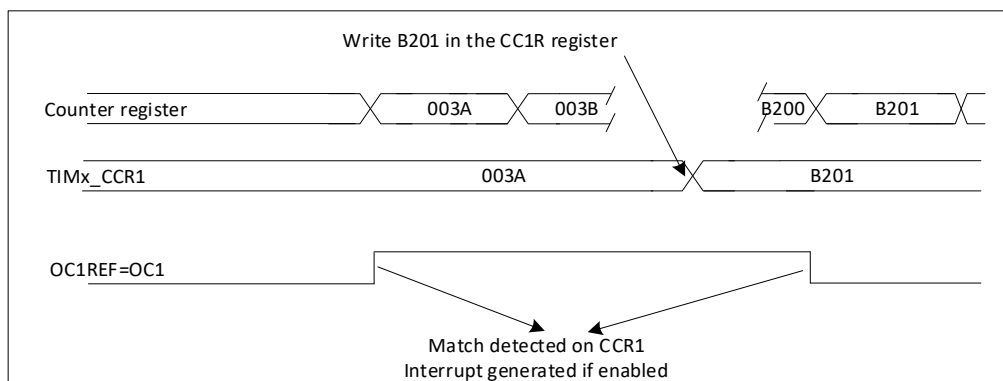


Figure 21-32 Output Compare Mode, Flip OC1

21.3.10. PWM mode

Pulse width modulation mode can allow the generation of a signal with a frequency determined by the TIMx_ARR register and a duty cycle determined by the TIMx_CCRx register.

Writing "110" (PWM mode 1) or "111" (PWM mode 2) to the OCxM bit in the TIMx_CMRx register can independently set each OCx output channel to generate one way PWM. The corresponding pre-load registers must be enabled by setting the OCxPE bit of the TIMx_CMRx register, and finally by setting the ARPE bit of the TIMx_CR1 register, which (in up-counting or center-symmetric modes) enables the auto-reloading preload registers.

Preloaded registers are transferred to the shadow registers only when an update event occurs, so all registers must be initialized by setting the UG bit in the TIMx_EGR register before the counter starts counting.

The polarity of the OCx can be set by software in the CCxP bit in the TIMx_CCER register, which can be set to active high or active low. The output enable of the OCx is controlled by a combination of the CCxE, CCxNE, MOE, OSSI, and OSSR bits (in the TIMx_CCER and TIMx_BDTR registers). See the description of the TIMx_CCER register for details.

In PWM mode (Mode 1 or Mode 2), TIMx_CNT and TIMx_CCRx are always being compared, (based on the counter count direction) to determine if $\text{TIMx_CCRx} \leq \text{TIMx_CNT}$ or $\text{TIMx_CNT} \leq \text{TIMx_CCRx}$ is met.

Depending on the state of the CMS bit in the TIMx_CR1 register, the timer is able to generate either an edge-aligned PWM signal or a center-aligned PWM signal.

PWM Edge Alignment Mode

■ Up Count Configuration

Performs an upward count when the DIR bit in the TIMx_CR1 register is low. See below for an example of PWM mode 1. The PWM reference signal OCxREF is high when $\text{TIMx_CNT} < \text{TIMx_CCRx}$ and low otherwise. If the comparison value in TIMx_CCRx is greater than the auto-reload value (TIMx_ARR), OCxREF is held to '1'. If the comparison value is 0, OCxREF remains '0'. The following figure shows an example of an edge-aligned PWM waveform with TIMx_ARR=8.

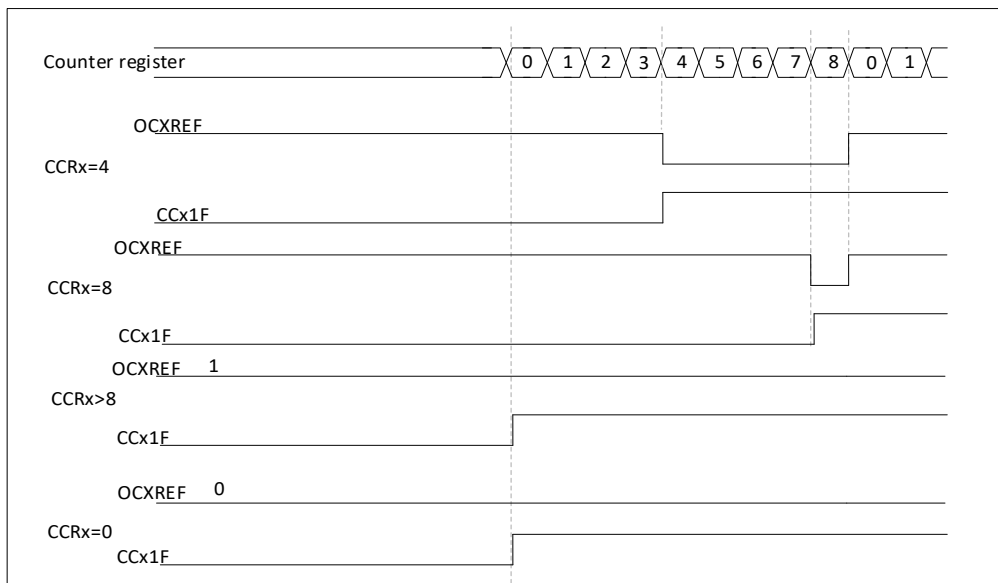


Figure 21-33 Edge-aligned PWM output, up (ARR=8)

■ Down Count Configuration

Performs a down count when the DIR bit of the TIMx_CR1 register is high.

In PWM mode 1, the reference signal OCxREF is low when $\text{TIMx_CNT} > \text{TIMx_CCRx}$, otherwise it is high. If the comparison value in TIMx_CCRx is greater than the auto-reload value in TIMx_ARR, OCxREF is held to '1'. A 0% PWM waveform cannot be generated in this mode.

PWM Central Alignment Mode

Central alignment mode when the CMS bit in the TIMx_CR1 register is not '00' (all other configurations have the same effect on the OCxREF/OCx signals). Depending on the CMS bit setting, the compare flag may be set to 1 when the counter is counting up, set to 1 when the counter is counting down, or set to 1 when the counter is counting up and down. The count direction bit (DIR) in the TIMx_CR1 register is updated by hardware; do not modify it with software.

The following figure gives some examples of centrally aligned PWM waveforms

- TIMx_ARR = 8
- PWM mode 1
- CMS=01 in the TIMx_CR1 register sets the compare flag when the counter counts down in central alignment mode

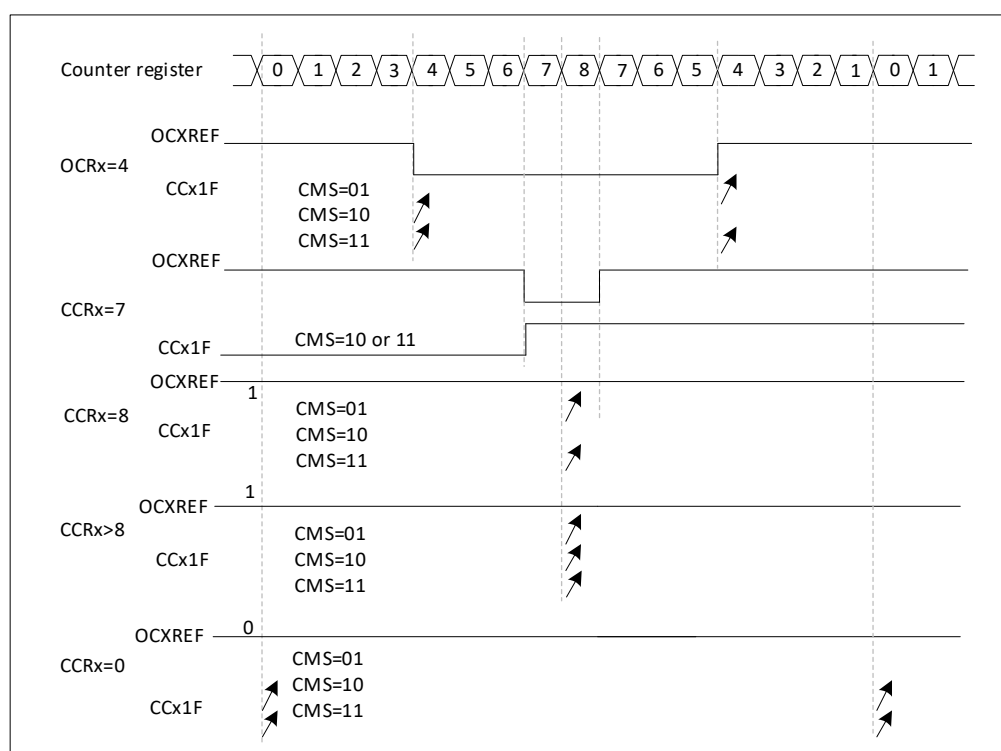


Figure 21-34 Centrally aligned PWM waveform (APR=8)

Tips for using the center alignment mode:

- The current count up/down configuration is used when entering central alignment mode; this means that whether the counter counts up or down depends on the current value of the DIR bit in the TIMx_CR1 register. In addition, the software cannot modify the DIR and CMS bits at the same time.
- It is not recommended to rewrite the counters when running in center-aligned mode, as this can produce unpredictable results. Specifically:— If the write counter value is greater than the auto-reload value (TIMx_CNT>TIMx_ARR), the direction will not be updated. For example, if the counter is counting up, it will continue to count up. — If a value of 0 or TIMx_ARR is written to the counter, the direction is updated but no update event UEV is generated.

- The safest way to use the central alignment mode is to generate a software update (set the UG bit in the TIMx_EGR bit) before starting the counter and not to modify the counter value while the count is in progress.

21.3.11. Complementary outputs and deadband insertion

The Advanced Control Timer (TIM1) is capable of outputting two complementary signals and manages the instantaneous turn-off and turn-on of the outputs. This period of time is often referred to as the deadband, and the user should adjust the deadband time according to the connected output devices and their characteristics (level shifting delays, power switching delays, etc.).

Configuring the CCxP and CCxNP bits in the TIMx_CCER register allows you to independently select the polarity (primary output OCx or complementary output OCxN) for each output.

The complementary signals OCx and OCxN are controlled by a combination of the following control bits: the CCxE and CCxNE bits in the TIMx_CCER register, and the MOE, OISx, OISxN, OSS1, and OSSR bits in the TIMx_BDTR and TIMx_CR2 registers, as detailed in Table 21-2 Complementary Output Channels with Brakes OCx and OCxN control bits. In particular, the deadband is activated upon transition to the IDLE state (MOE drops to 0).

Setting both the CCxE and CCxNE bits will insert a dead zone, and the MOE bit will also be set if a brake circuit is present. Each channel has an 8-bit deadband generator DTG[7:0]. The reference signal OCxREF generates 2 outputs OCx and OCxN. If OCx and OCxN are highly valid:

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

If the delay is greater than the currently valid output width (OCx or OCxN), the corresponding pulse is not generated.

The following graphs show the relationship between the output signal of the deadband generator and the current reference signal, OCxREF. (Assuming CCxP=0, CCxNP=0, MOE=1, CCxE=1 and CCxNE=1)

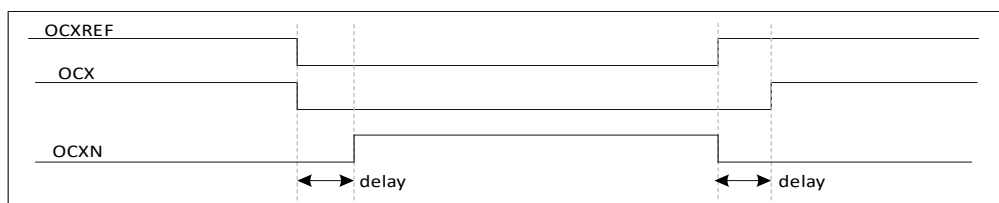


Figure 21-35 Complementary output with deadband insertion

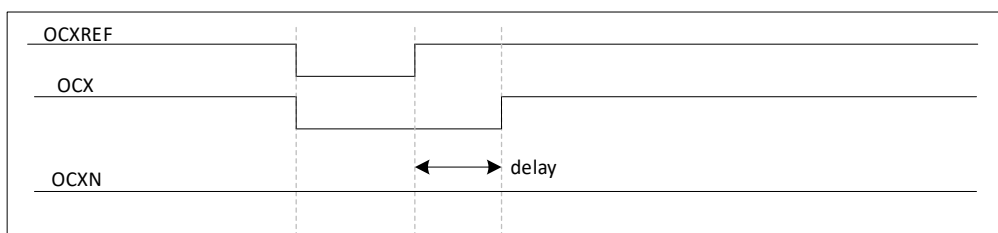


Figure 21-36 Deadband waveform delay greater than negative pulse

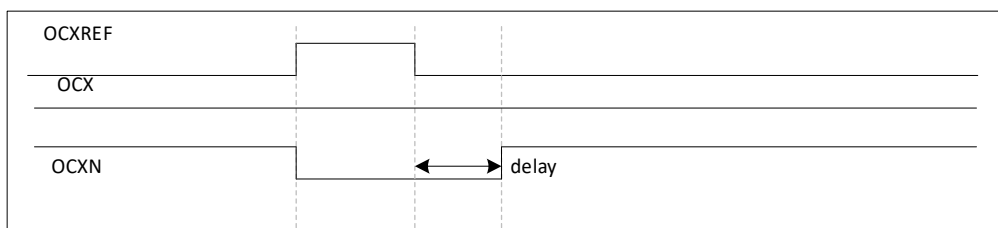


Figure 21-37 Deadband waveform delay is greater than positive pulse

The deadband delay is the same for each channel and is programmatically configured by the DTG bit in the TIMx_BDTR register.

Redirecting OCxREF to OCx or OCxN

In output mode (strong set, output compare, or PWM), OCxREF can be redirected to the output of OCx or OCxN by configuring the CCxE and CCxNE bits of the TIMx_CCER register. This function can send a special waveform (e.g. PWM or static active level) on one of the outputs when the complementary output is at an invalid level. Another effect is to have both outputs at the same time at an invalid level, or at an active level and a complementary output with deadband.

Note: When only OCxN is enabled (CCxE=0, CCxNE=1), it is not inverted and goes high immediately when OCxREF is active. *For example, if CCxNP = 0, then OCxN = OCxREF.* On the other hand, when both OCx and OCxN are enabled (CCxE=CCxNE=1), OCx is active when OCxREF is high; and OCxN, on the contrary, becomes active when OCxREF is low.

21.3.12. Using the brake function

When the brake function is used, both the output enable signal and the invalid level signal are modified based on additional control bits. In any case, the OCx and OCxN outputs cannot be on the active level at the same time at the same time.

The brake source can be either the brake input pin or the following internal source:

- CPU LOCKUP output
- PVD Output
- Clock failure events generated by CSS monitoring
- Output from comparator

After a system reset, the brake circuit is disabled and the MOE bit is low. Setting the BKE bit in the TIMx_BDTR register enables the brake function, and the polarity of the brake input signal can be selected by configuring the BKP bit in the same register. BKE and BKP can be modified at the same time. When writing

the BKE and BKP bits, there is a delay of 1 APB clock cycle before they are actually written, so it is necessary to wait one APB clock cycle before the written bits are read back correctly.

Because the MOE falling edge can be asynchronous, a resynchronization circuit is set up between the actual signal (acting on the output) and the synchronization control bit (in the TIMx_BDTR register). This resynchronization circuit creates a delay between the asynchronous and synchronous signals. In particular, if MOE=1 is written when it is low, a delay (null instruction) must be inserted before it is read to get the correct value. This is because writes are asynchronous signals while reads are synchronous signals.

When braking occurs (a selected level appears at the brake input), the following actions are available:

- The MOE bit is cleared asynchronously, placing the output in an invalid state, an idle state, or releasing control of the GPIO (selected by the OSSR bit). This feature remains in effect when the MCU's oscillator is turned off.
- Once MOE=0, each output channel outputs the level set by the OISx bit in the TIMx_CR2 register. If OSSR=0, the timer releases the enable output, otherwise the enable output is always high.
- When using complementary outputs:
 - The outputs are first placed in a reset state i.e. an invalid state (depending on the polarity). This is asynchronous operation and is valid even when the timer is not clocked.
 - If the timer clock is still present, the deadband generator will re-activate, driving the output port after the deadband according to the level indicated by the OISx and OISxN bits. Even in this case, OCx and OCxN cannot be driven to valid levels at the same time. Note that because of the resynchronization of the MOE, the dead time is a bit longer than usual (about 2 ck_tim clock cycles).
 - The timer releases the enable output if OSSR=0, otherwise it holds the enable output; or the enable output goes high once one of CCxE and CCxNE goes high.
- If the BIE bit in the TIMx_DIER register is set, an interrupt is generated when the brake status flag (BIF bit in the TIMx_SR register) is '1'.
- If the AOE bit in the TIMx_BDTR register is set, the MOE bit is automatically set at the next update event UEV; this can be used for shaping, for example. Otherwise, the MOE remains low until it is set to '1' again; at this point, this feature can be used for safety, where you can connect the brake input to a power-driven alarm output, thermal sensor, or other safety device.

Note: Brake input is level active. Therefore, when the brake input is active, the MOE cannot be set at the same time (automatically or via software). *At the same time, the status flag BIF cannot be cleared.*

The brake can be generated by the BRK input, which has a programmable active polarity and is turned on by the BG bit in the TIMx_BDTR register.

In addition to brake input and output management, write protection is implemented in the brake circuitry to ensure application security. It allows the user to freeze several configuration parameters (deadband length, OCx/OCxN polarity and disabled states, OCxM configuration, brake enable and polarity). The user can select one of the three levels of protection by using the LOCK bit in the TIMx_BDTR register. The LOCK bit can be modified only once after MCU reset.

The following figure shows an example of the output of the response brake.

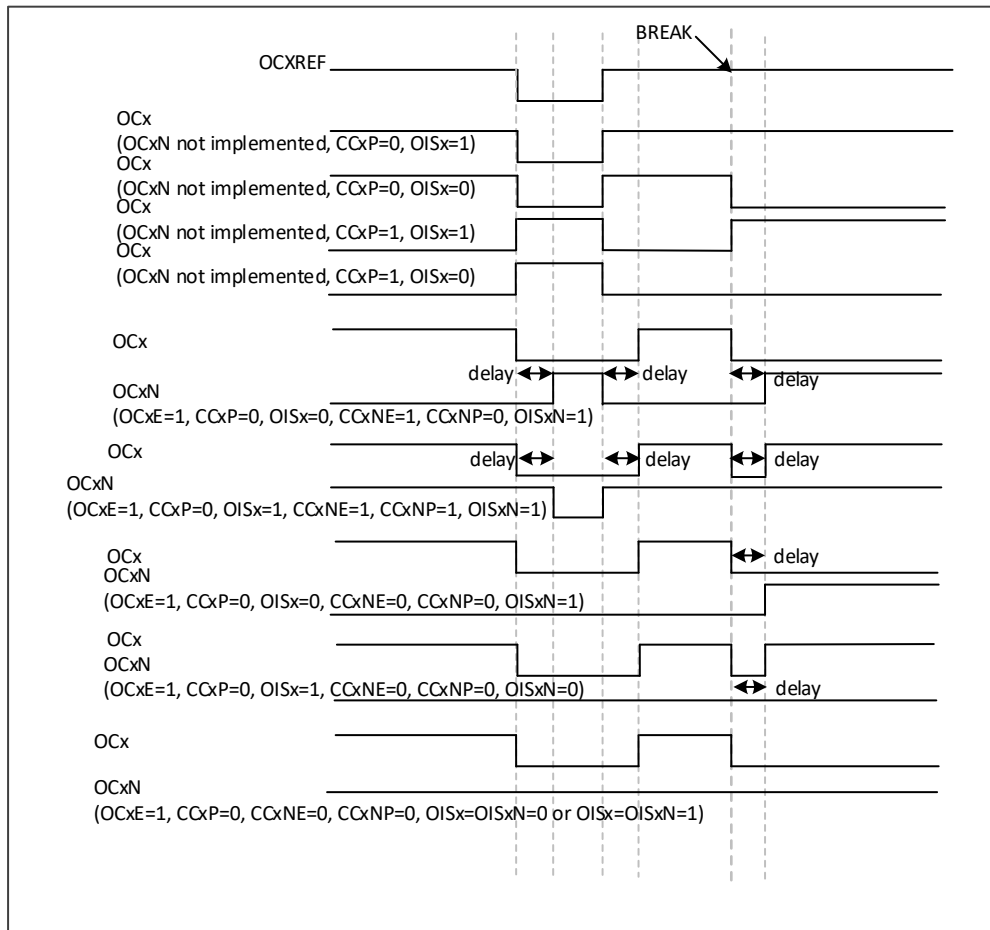


Figure 21-38 Output in response to braking

21.3.13. Clearing the OCxREF signal on an external event

For a given channel, setting the corresponding OCxCE bit in the TIMx_CMRx register to 1 enables the OCxREF signal to be pulled low with a high level on the ETRF input, and the OCxREF signal will remain low until the next update event UEV.

This function can only be used in Output Compare and PWM modes, not in Forced mode.

And OCREF_CLR_INPUT can be selected between OCREF_CLR and ETRF (after ETR filtering) by configuring the OCCS bit in the TIMx_SMCR register.

For example, the OCxREF signal can be coupled to the output of a comparator for current control. At this point, the ETR must be configured as follows:

1. The externally triggered prescaler must be off: ETPS[1:0]=00 in the TIMx_SMCR register.
2. External clock mode 2 must be disabled: ECE=0 in the TIMx_SMCR register.
3. External Trigger Polarity (ETP) and External Trigger Filter (ETF) can be configured as required.

The following figure shows the action of the OCxREF signal when the ETRF input becomes high, corresponding to different OCxCE values. In this example, timer TIMx is placed in PWM mode.

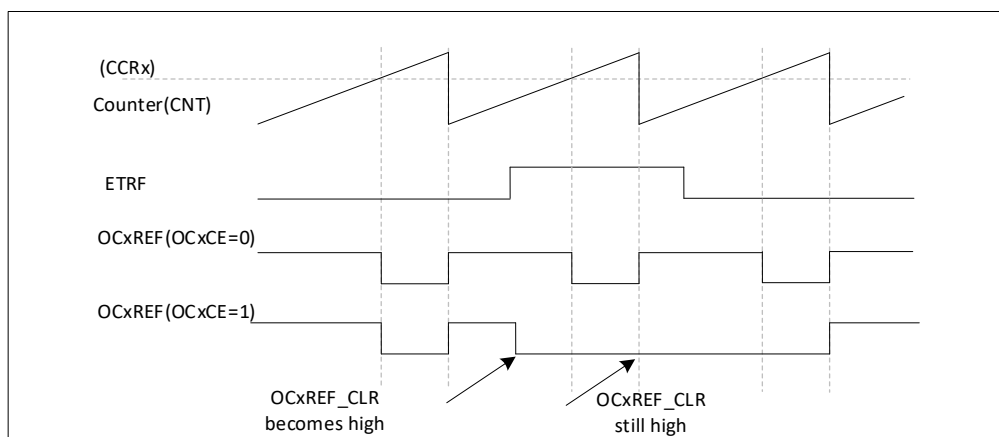


Figure 21-39 Clear OCxREF of TIM1

21.3.14. Six-step PWM generation

When complementary outputs are required on a channel, the preloaded bits are OCxM, CCxE, and CCxNE. On the occurrence of a COM phase change event, these preloaded bits are transferred to the shadow register bits. This makes it possible to pre-set the next step configuration and to modify the configuration of all channels simultaneously at the same moment. COM can be generated by software by setting the COM bit in the TIMx_EGR register, or by hardware on the rising edge of TRGI.

A flag bit (COMIF bit in the TIMx_SR register) is set when a COM event occurs, at which point an interrupt is generated if the COMIE bit in the TIMx_DIER register has been set, or a DMA request is generated if the COMDE bit in the TIMx_DIER register has been set.

The following figure shows the OCx and OCxN outputs for three different configurations when a COM event occurs.

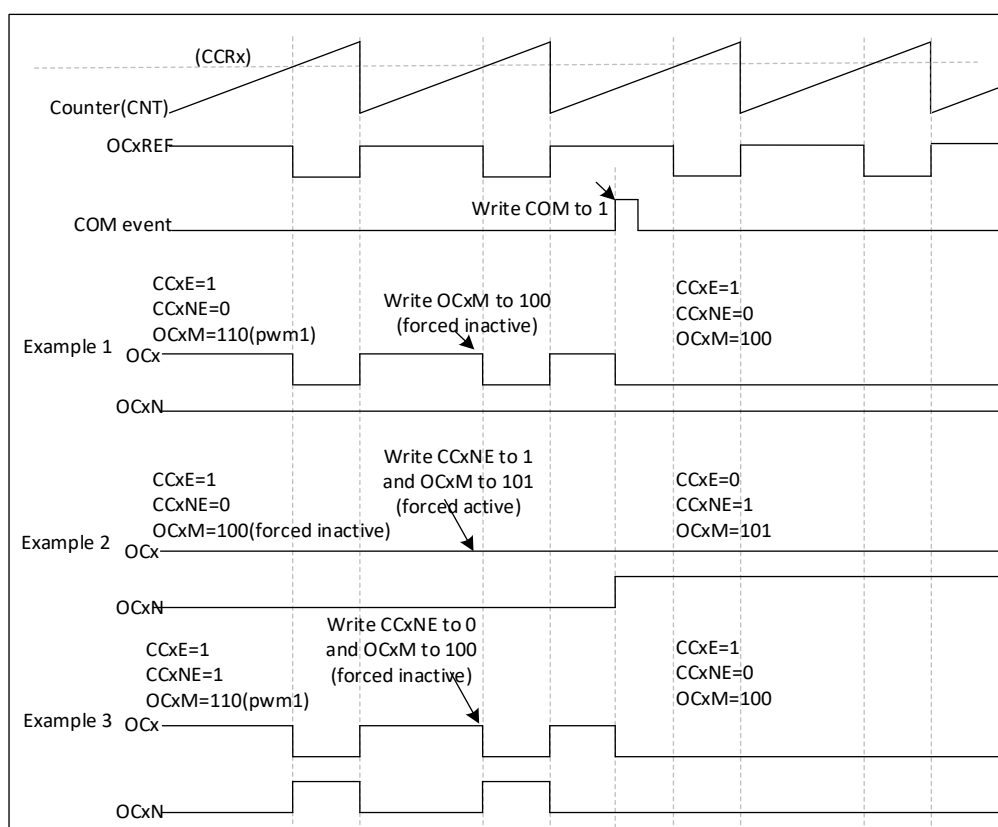


Figure 21-40 Six-step generation, COM example (OSSR=1)

21.3.15. Single Pulse Mode

The single pulse mode (OPM) is a special case of one of the many modes described previously. This mode allows the counter to respond to an excitation and, after a programmable delay, generate a pulse with a pulse width that can be controlled by the program.

The counter can be activated from the mode controller to generate waveforms in output comparison mode or PWM mode. Setting the OPM bit of the TIMx_CR1 register will select single-pulse mode, which allows the counter to automatically stop when the next counter overflow is generated.

A pulse is generated only when the comparison value is different from the initial value of the counter. Before starting (when the timer is waiting to be triggered), it must be configured as follows:

- Upward counting mode: counter $CNT < CCRx \leq ARR$ (in particular, $0 < CCRx$)
- Downward counting mode: Counter $CNT > CCRx$

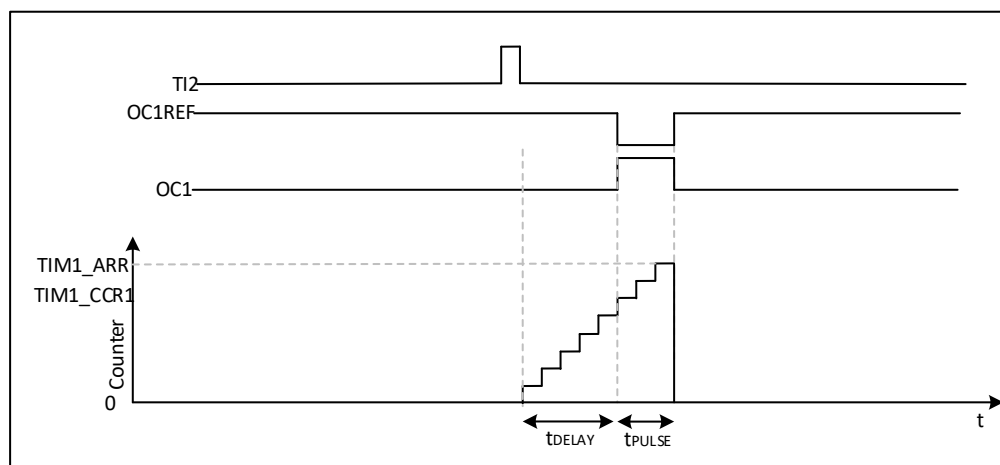


Figure 21-41 Example of single pulse mode

For example, when it is necessary to generate a positive pulse of length t_{PULSE} on OC1 after delaying t_{DELAY} at the beginning of a rising edge detected from the TI2 input pin.

Use the TI2FP2 as a trigger:

- Set CC2S=01 in the TIMx_CMR1 register to map TI2FP2 to TI2.
- Set CC2P=0 in the TIMx_CCER register to enable the TI2FP2 to detect the rising edge.
- Setting TS=110 in the TIMx_SMCR register triggers the TI2FP2 as a slave mode controller (TRGI).
- Setting SMS=110 (trigger mode) in the TIMx_SMCR register, the TI2FP2 is used to start the counter.

The OPM waveform is determined by the value written to the comparison register (taking into account the clock frequency and counter prescaler)

- t_{DELAY} is defined by the value in the TIMx_CCR1 register.
- t_{PULSE} is defined by the difference between the auto-reload value and the comparison value (TIMx_ARR - TIMx_CCR1).
- Assuming that a waveform from 0 to 1 is to be generated when a comparison match occurs, and a waveform from 1 to 0 is to be generated when the counter reaches the preloaded value; firstly, set OC1M=111 in TIMx_CMR1 register to enter PWM mode 2; selectively enable the preload registers as needed: set OC1PE=1 in TIMx_CMR1 and ARPE in TIMx_CR1 register; then fill the comparison value in TIMx_CCR1 register, and set the UG bit to generate an update event. then fill in the comparison value in the TIMx_CCR1 register and the auto-reload value in the TIMx_ARR register, set the UG bit to generate an update event, and wait for an external trigger event on TI2. In this example, CC1P = 0.

In this example, the DIR and CMS bits in the TIMx_CR1 register should be set low.

Since only one pulse is needed, OPM=1 in the TIMx_CR1 register must be set to stop counting at the next update event (when the counter flips from the auto-reload value to 0).

Special case: OCx fast enable:

In single pulse mode, the CEN bit is set by the edge detection logic at the TIx input pin to start the counter. The comparison operation between the counter and the comparison value then produces the conversion of the output. However, these operations require a certain number of clock cycles, so it limits the minimum delay tDELAY that can be obtained.

If you want to output the waveform with minimum delay, you can set the OCxFE bit in the TIMx_CMRx register; at this point, OCxREF (and OCx) responds directly to the excitation and no longer relies on the result of the comparison, and the output waveform is the same as the waveform when the comparison is matched. OCxFE only works when the channel is configured for PWM1 and PWM2 modes.

21.3.16. Encoder Interface Mode

The encoder interface mode is selected by setting SMS=001 in the TIMx_SMCR register if the counter counts only on the TI2 edge, SMS=010 if it counts only on the TI1 edge, or SMS=011 if the counter counts on both the TI1 and TI2 edges.

TI1 and TI2 polarity can be selected by setting the CC1P and CC2P bits in the TIMx_CCER register; the input filter can also be programmed if desired.

Two inputs TI1 and TI2 are used as interfaces to the incremental encoder. Referring to Table 21-1, assuming that the counter has been started (CEN=1 in the TIMx_CR1 register), the counter is driven by each valid hop on TI1FP1 or TI2FP2. TI1FP1 and TI2FP2 are the signals of TI1 and TI2 after passing them through the input filter and polarity control; without filtering and phasing, TI1FP1 = TI1 and TI2FP2 = TI2. Based on the hopping sequence of the two input signals, a counting pulse and a direction signal are generated. Depending on the hopping order of the two input signals, the counter counts up or down while the hardware sets the DIR bit of the TIMx_CR1 register accordingly. Whether the counter relies on TI1 for counting, on TI2 for counting, or on both TI1 and TI2 for counting, a trip on either input (TI1 or TI2) recalculates the DIR bit.

The encoder interface mode is basically equivalent to using an external clock with direction selection. This means that the counter only counts continuously between 0 and the auto-reload value of the TIMx_ARR register (either 0 to ARR count or ARR to 0 count, depending on the direction). So TIMx_ARR must be configured before counting starts; again, the captures, comparators, prescalers, repeat counters, trigger output characteristics, etc. still work as usual. Encoder mode and external clock mode 2 are not compatible and therefore cannot be operated simultaneously. In this mode, the counter is automatically modified according to the speed and direction of the incremental encoder, so that the contents of the counter always indicate the position of the encoder. The counting direction corresponds to the direction of rotation of the connected sensor. The following table lists all possible combinations, assuming that TI1 and TI2 do not transform simultaneously.

Table 21-1 Relationship between count direction and encoder signal

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

Counting on TI1 and TI2	Low	Up	Down	Down	Up
----------------------------	-----	----	------	------	----

An external incremental encoder can be connected directly to the MCU without external interface logic. However, a comparator is typically used to convert the differential output of the encoder to a digital signal, which greatly increases the resistance to noise interference. The third signal from the encoder output indicates the mechanical zero point and can be connected to an external interrupt input and trigger a counter reset.

The following figure is an example of counter operation showing the generation of counting signals and direction control. It also shows how input jitter is suppressed when a double edge is selected; jitter may be generated when the sensor's position is close to a transition point. In this example, we assume the following configuration:

- CC1S='01' (TIMx_CMR1 register, TI1FP1 mapped to TI1)
- CC2S='01' (TIMx_CMR2 register, TI2FP2 mapped to TI2)
- CC1P='0' (TIMx_CCER register, TI1FP1 not inverted, TI1FP1=TI1)
- CC2P='0' (TIMx_CCER register, TI2FP2 not inverted, TI2FP2=TI2)
- SMS='011' (TIMx_SMCR register, all inputs valid on rising and falling edges)
- CEN='1' (TIMx_CR1 register, counter enable)

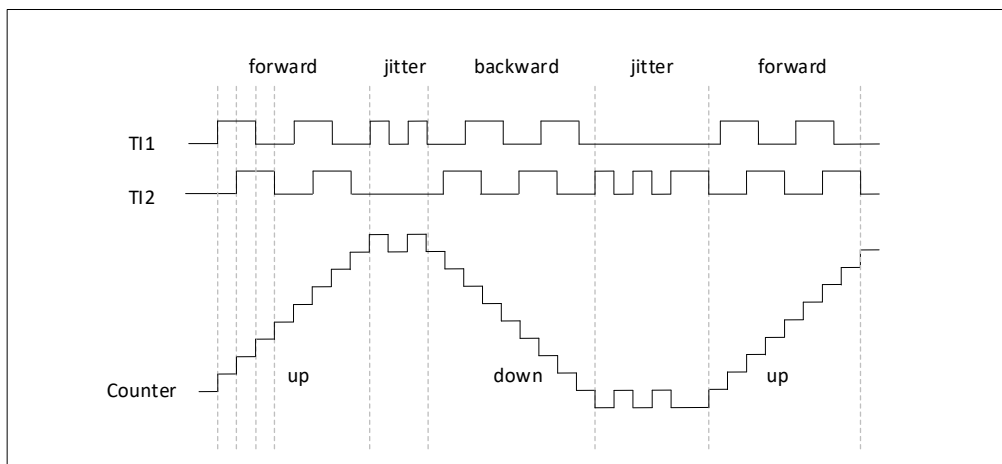


Figure 21-42 Example of counter operation in encoder mode

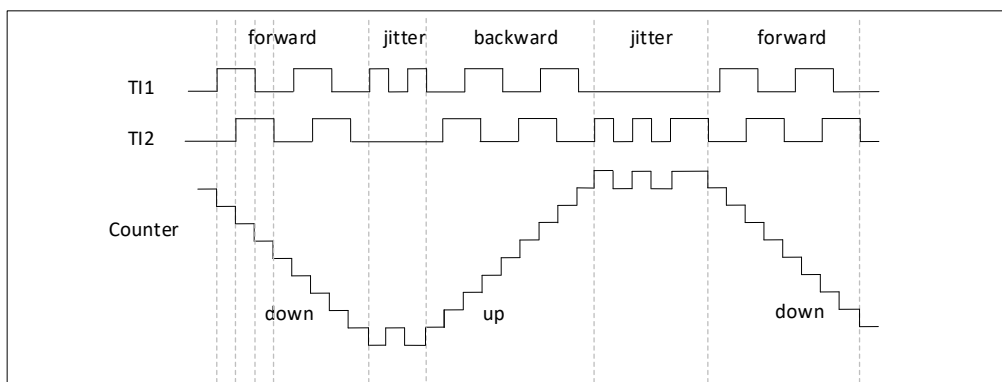


Figure 21-43 TI1FP1 Inverted Encoder Interface Mode Example

Provides information on the current position of the sensor when the timer is configured in encoder interface mode. Using a second timer configured in capture mode, it is possible to measure the interval between the two encoder events and obtain information about the dynamics (velocity, acceleration, deceleration). The

encoder output indicating the mechanical zero point can be used for this purpose. Depending on the interval between two events, the counter can be read out at a fixed time. If possible, the counter's value can be latched to a third input capture register (the capture signal must be periodic and can be generated by another timer); its value can also be read by a DMA request generated by the real-time clock.

21.3.17. Timer input heterodyne function

The TI1S bit of the TIM_CR2 register allows the input filter of channel 1 to be connected to the output of an iso-gate whose three inputs are TIMx_CH1, TIMx_CH2, and TIMx_CH3.

The heterosync outputs can be used for all timer input functions such as triggering or input capture.

21.3.18. Interfacing with Hall Sensors

When using the advanced timer (TIM1) to generate a PWM signal to drive the motor, another general-purpose timer (TIM3) can be used as an "interface timer" to connect the Hall sensor. The three timer input pins (CC1, CC2, CC3) are connected to the TI1 input channel through an iso-gate (selected by setting the TI1S bit in the TIMx_CR2 register), and the "interface timer" captures this signal.

The slave mode controller is configured to reset mode and the slave input is TI1F_ED. Whenever one of the 3 inputs changes, the counter starts counting from 0 again. This produces a time reference triggered by any change in the Hall input.

Capture/compare channel 1 on the interface timer is configured in capture mode with the capture signal being TRC (see Figure 21-27). The captured value reflects the time delay between two input changes, giving information about the motor speed.

The interface timer can be used to generate a pulse in output mode which can be used (by triggering a COM event) to change the attributes of the individual channels of the advanced timer TIM1, which generates a PWM signal to drive the motor. Therefore the interface timer channel must be programmed to generate a positive pulse after a specified delay (output comparison or PWM mode), and this pulse is sent to the advanced timer TIM1 via the TRGO output.

Example: A Hall input is connected to a TIMx timer that requires the PWM configuration of the advanced control timer TIMx to be changed at a specified moment after each change on either Hall input.

- Set the TI1S bit of the TIMx_CR2 register to '1' to configure the three timer inputs to logic-or to the TI1 input.
- Time base programming: set TIMx_ARR to its maximum value (the counter must be cleared by a change in TI1). Set the prescaler to get a maximum counter period that is longer than the time interval between two changes on the sensor.
- Set channel 1 to capture mode (TRC checked): set CC1S=01 in the TIMx_CMR1 register, and set the digital filter if desired.
- Set channel 2 to PWM2 mode with the requested delay: set OC2M=111 and CC2S=00 in the TIMx_CMR1 register.
- Select OC2REF as the trigger output on the TRGO: Set MMS=101 in the TIMx_CR2 register.

In the advanced control register TIM1, the correct ITR input must be a flip-flop input, the timer is programmed to generate a PWM signal, and the capture/compare control signal is preloaded (CCPC=1 in the

TIMx_CR2 register) while triggering the input control COM event (CCUS=1 in the TIMx_CR2 register). After a COM event, the next PWM control bits (CCxE, OCxM) are written, which can be implemented in the interrupt subroutine that handles the rising edge of OC2REF.

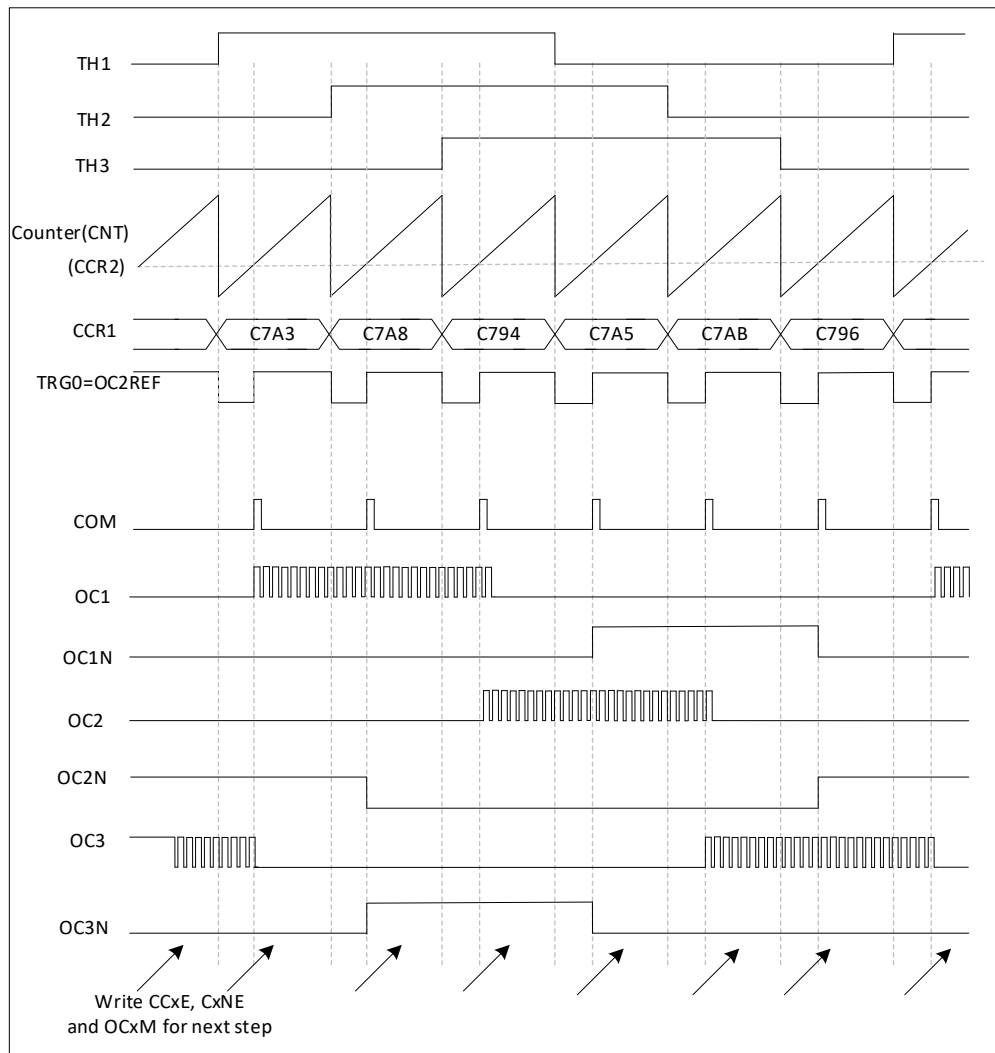


Figure 21-44 Example of Hall sensor interface

21.3.19. Timer and external trigger synchronization

The TIMx timer can be synchronized with an external trigger in a variety of modes: reset mode, gated mode, and triggered mode.

Slave mode: Reset mode

On the occurrence of a trigger input event, the counter and its prescaler are able to be reinitialized; at the same time, an update event UEV is also generated if the URS bit of the TIMx_CR1 register is low; all pre-loaded registers (TIMx_ARR, TIMx_CCRx) are then updated.

In the following example, the rising edge of the TI1 input causes the up counter to be cleared:

- Configure channel 1 to detect the rising edge of TI1. Configure the bandwidth of the input filter (in this example, no filter is required, so keep IC1F=0000). The capture prescaler is not used in the trigger operation, so no configuration is required. The CC1S bit selects only the input capture source, i.e.,

CC1S=01 in the TIMx_CMR1 register. Set CC1P=0 in the TIMx_CCER register to determine polarity (detects only the rising edge).

- Set SMS=100 in TIMx_SMCR register to configure the timer in reset mode; set TS=101 in TIMx_SMCR register to select TI1 as the input source.
- Set CEN=1 in the TIMx_CR1 register to start the counter.

The counter starts counting according to the internal clock and then operates normally until a rising edge occurs in TI1; at this point, the counter is cleared to zero and then restarts counting from zero. At the same time, the trigger flag (TIF bit in the TIMx_SR register) is set, generating either an interrupt request or a DMA request depending on the setting of the TIE (interrupt enable) bit and the TDE (DMA enable) bit in the TIMx_DIER register.

The following figure shows the action when the auto-reload register TIMx_ARR = 0x36. The delay between the rising edge of TI1 and the actual reset of the counter depends on the resynchronization circuit at the TI1 input.

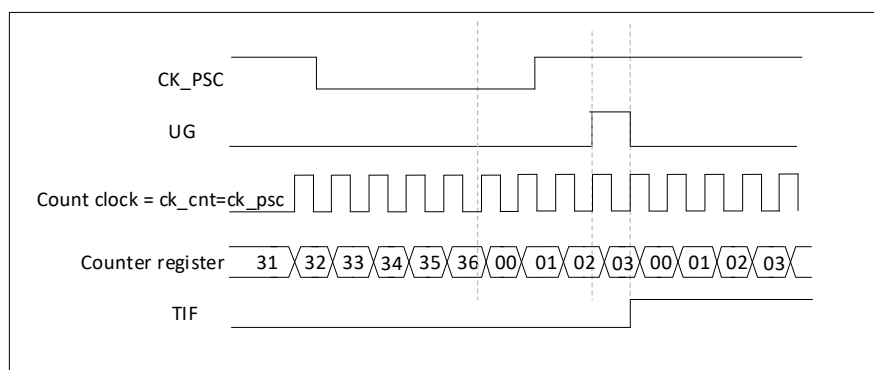


Figure 21-45 Control circuit in reset mode

From Mode: Gated Mode

Enable the counter according to the selected input level.

In the following example, the counter only counts up when TI1 is low:

- Configure channel 1 to detect a low level on TI1. Configure the input filter bandwidth (in this case, no filtering is required, so keep IC1F=0000). The capture prescaler is not used in the trigger operation, so no configuration is required. The CC1S bit is used to select the input capture source by setting CC1S=01 in the TIMx_CMR1 register. Set CC1P=1 in the TIMx_CCER register to determine polarity (detects only low levels).
- Set SMS=101 in the TIMx_SMCR register to configure the timer for gating mode; set TS=101 in the TIMx_SMCR register to select TI1 as the input source.
- Set CEN=1 in the TIMx_CR1 register to start the counter. In gated mode, if CEN=0, the counter cannot start, regardless of the trigger input level.

As long as TI1 is low, the counter starts counting based on the internal clock and stops counting once TI1 goes high. The TIF scaler in TIMx_SR is set when either the counter starts or stops.

The delay between the rising edge of TI1 and the actual stopping of the counter depends on the resynchronization circuit at the TI1 input.

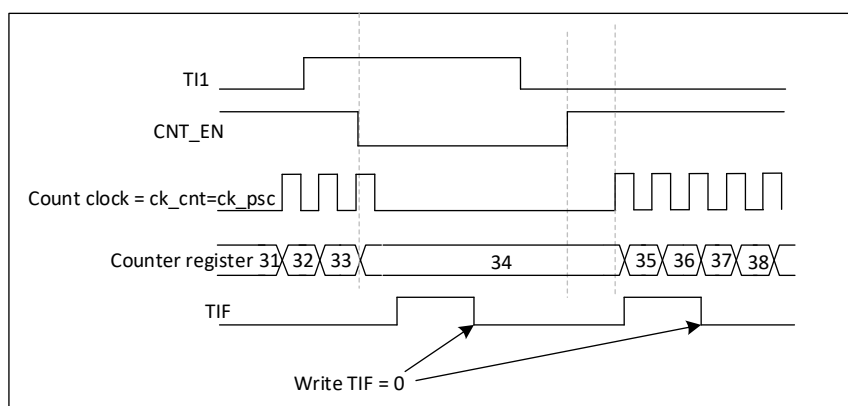


Figure 21-46 Control circuit in gated mode

The selected event on the input enables the counter.

In the following example, the counter starts counting up on the rising edge of the TI2 input:

1. Configure channel 2 to detect the rising edge of TI2. Configure the input filter bandwidth (in this example, no filter is required, keep IC2F=0000). The capture prescaler is not used in the trigger operation and does not require configuration. The CC2S bit is only used to select the input capture source by setting CC2S=01 in the TIMx_CMR1 register. Set CC2P=1 in the TIMx_CCER register to determine polarity (detects only low levels).
2. Set SMS=110 in TIMx_SMCR register to configure the timer in trigger mode; set TS=110 in TIMx_SMCR register to select TI2 as the input source.

When a rising edge of TI2 occurs, the counter starts counting driven by the internal clock and the TIF flag is set. The delay between the rising edge of TI2 and the counter starting to count depends on the resynchronization circuit at the TI2 input.

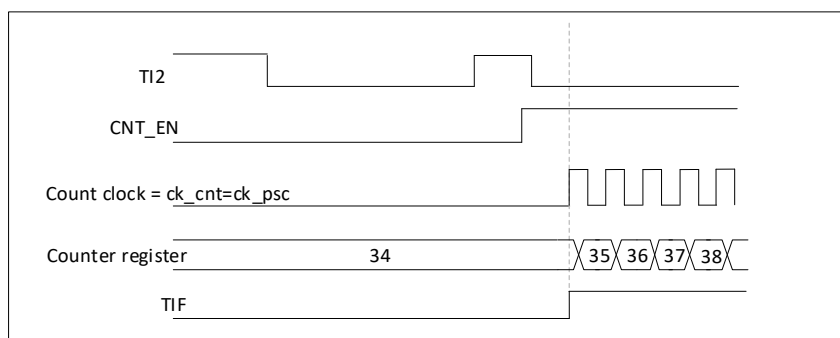


Figure 21-47 Control circuit in gated mode

Slave Mode: External Clock Mode 2 + Trigger Mode

External clock mode 2 can be used with another slave mode (except external clock mode 1 and encoder mode). In this case, the ETR signal is used as the input for the external clock, and another input can be selected as the trigger input in reset mode, gated mode, or trigger mode. It is not recommended to use the TS bit of the TIMx_SMCR register to select ETR as the TRGI.

In the following example, once a rising edge occurs on TI1, the counter counts up on each rising edge of ETR:

- Configure the external trigger input circuit through the TIMx_SMCR register:

ETF=0000: no filtering

— ETPS=00: no prescaler used

– ETP=0: detect the rising edge of ETR, set ECE=1 to enable external clock mode 2.

- Configure channel 1 as follows to detect the rising edge of TI:

IC1F=0000: No filtering.

-The capture prescaler is not used in the trigger operation and does not need to be configured.

Set CC1S=01 in the TIMx_CMR1 register to select the input capture source.

– Set CC1P=0 in the TIMx_CCER register to determine polarity (detects only rising edges)

- Set SMS=110 in the TIMx_SMCR register to configure the timer for trigger mode. Set TS=101 in the TIMx_SMCR register to select TI1 as the input source.

When a rising edge occurs on TI1, the TIF flag is set and the counter starts counting on the rising edge of ETR. The delay between the rising edge of the ETR signal and the actual reset of the counter depends on the resynchronization circuit at the ETRP input.

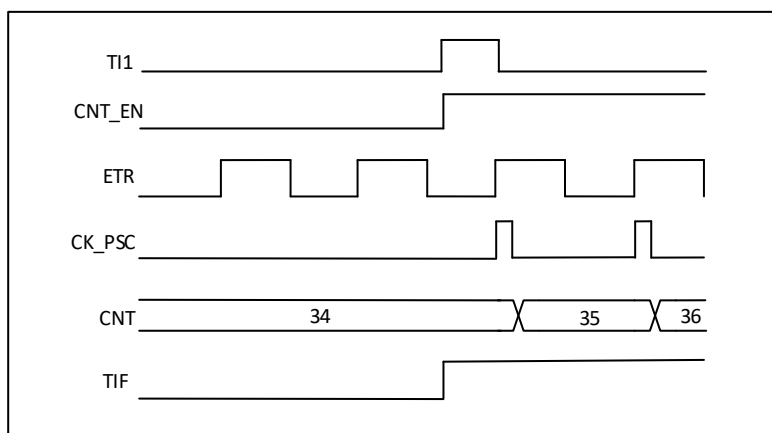


Figure 21-48 Control circuit in external clock mode 2 + trigger mode

21.3.20. Timer synchronization

The TIM timer is connected internally for timer synchronization or linking functions. When a timer is in master mode, it can reset, start, and stop the counter of another timer in slave mode. Refer to Section 22.3.14 Timer Synchronization Description for details.

21.3.21. Debug mode

When the chip enters the debug mode, the TIMx counter can continue to work normally or stop working according to the setting of DBG_TIMx_STOP in the DBG module.

21.4. TIM1 Register Description

21.4.1. TIM1 Control Register 1 (TIM1_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

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

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

Bit	Name	R/W	Reset Value	Function
31: 10	Res	-	-	Res
9: 8	CKD [1:0]	RW	00	<p>Clock Dividing Factor These 2 bits define the ratio between the frequency of the timer clock (CK_INT), the dead time and the dividing ratio between the dead time generator and the sampling clock used by the digital filter (ETR, Tix)</p> <p>00: $t_{DTS} = t_{CK_INT}$</p> <p>01: $t_{DTS} = 2 \times t_{CK_INT}$</p> <p>10: $t_{DTS} = 4 \times t_{CK_INT}$</p> <p>11: reserved, do not use this!</p> <p>Configuration</p>
7	ARPE	RW	0	<p>Auto Reload Preload Allowed</p> <p>Bit 0: TIMx_ARR register is not buffered</p> <p>1: TIMx_ARR register is buffered</p>
6: 5	CMS [1:0]	RW	00	<p>Selecting Central Alignment Mode</p> <p>00: Edge alignment mode. The counter counts up or down based on the direction bit (DIR).</p> <p>01: Central Alignment Mode 1. The counter alternately counts up and down. Configured as an output channel (The output compare interrupt flag bit for (CCxS=00 in the TIMx_CMRx register) is set only when the counter counts down.</p> <p>10: Central Alignment Mode 2. The counter alternately counts up and down. Configured as an output compare interrupt flag bit for the output channel (CCxS=00 in the TIMx_CMRx register), which is only set when the counter counts up.</p> <p>11: Central Alignment Mode 3. The counter alternately counts up and down. Configured as an output compare interrupt flag bit for the output channel (CCxS=00 in the TIMx_CMRx register) that is set when the counter counts up and down.</p>

Bit	Name	R/W	Reset Value	Function
				Note: While the counter is on (CEN=1), switching from edge-aligned mode to center-aligned mode is not allowed.
4	DIR	RW	0	<p>direction of counting</p> <p>0: Counter counts up</p> <p>1: Counter counts down</p> <p>Note: This bit is read-only when the counter is configured for center-aligned mode or encoder mode</p>
3	OPM	RW	0	<p>Single Pulse Mode</p> <p>0: counter does not stop when an update event occurs</p> <p>1: The counter stops when the next update event (clearing the CEN bit) occurs.</p>
2	URS	RW	0	<p>Update request source</p> <p>The software selects the source of UEV events with this bit</p> <p>0: If generating an update interrupt or DMA request is allowed, either of the following events generates an update interrupt or DMA request:</p> <ul style="list-style-type: none"> - Counter overflow/underflow - Setting the UG position - Updates generated from the schema controller <p>1: If generating update interrupts or DMA requests is allowed, only counter overflow/underflow generates an update interrupt or DMA request</p>
1	UDIS	RW	0	<p>Update disable</p> <p>Software allows/disallows the generation of UEV events with this bit</p> <p>0: UEV is allowed. Update (UEV) events are generated by any of the following events:</p> <ul style="list-style-type: none"> - Counter overflow/underflow - Setting the UG position - Updates generated from the schema controller <p>Registers with caches are loaded with their preloaded values.</p> <p>1: UEV is prohibited. No update events are generated and the shadow registers (ARR,PSC,CCRx) hold their values.</p> <p>If the UG bit is set or a hardware reset is issued from the mode controller, the counter and prescaler are reinitialized.</p>
0	CEN	RW	0	<p>Enable Counter</p> <p>0: Disable counter</p> <p>1: Turn on the counter</p>

Bit	Name	R/W	Reset Value	Function
				Note: The external clock, gated mode and encoder mode will not work until the CEN bit is set in software. Trigger mode can be automatically set in hardware with the CEN bit.

21.4.2. TIM1 Control Register 2 (TIM1_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Re s	OIS 4	OIS3 N	OIS 3	OIS2 N	OIS 2	OIS1 N	OIS 1	TI1 S	MMS [2:0]			CC DS	CCU S	Re s	CCP C
-	RW	RW	RW	RW	RW	RW	RW	RW	RW			RW	RW	-	RW

Bit	Name	R/W	Reset Value	Function
31: 15	Res	-	-	Res
14	OIS4	RW	0	Output idle state 4 (OC4 output). See the OIS1 bit.
13	OIS3N	RW	0	Output idle state 3 (OC3N output). See OIS1N bit
12	OIS3	RW	0	Output idle state 3 (OC3 output). See the OIS1 bit.
11	OIS2N	RW	0	Output idle state 2 (OC2N output). See the OIS1N bit.
10	OIS2	RW	0	Output idle state 2 (OC2 output). See OIS1 bit
9	OIS1N	RW	0	Output idle state 1 (OC1N output). 0: OC1N=0 after deadband when MOE=0 1: When MOE=0, OC1N=1 after the deadband Note: This bit cannot be modified after LOCK (TIMx_BKR register) level 1, 2 or 3 has been set.
8	OIS1	RW	0	Output idle state 1 (OC1 output). 0: When MOE=0, if OC1N is realized, then OC1=0 after dead zone 1: When MOE=0, if OC1N is realized, OC1=1 after the deadband Note: This bit cannot be modified after LOCK (TIMx_BKR register) level 1, 2 or 3 has been set.
7	TI1S	RW	0	TI1 Selection 0: The TIMx_CH1 pin is connected to the TI1 input.

Bit	Name	R/W	Reset Value	Function
				1: The TIMx_CH1, TIMx_CH2 and TIMx_CH3 pins are connected to the TI1 inputs via heterodyne.
6: 4	MMS [2:0]	RW	000	<p>Master Mode Selection</p> <p>These two bits are used to select the synchronization message (TRGO) sent to the slave timer in master mode. The possible combinations are as follows:</p> <p>000: Reset - The UG bit of the TIMx_EGR register is used as a trigger output (TRGO). If the trigger input (slave mode controller in reset mode) generates a reset, the signal on the TRGO is delayed relative to the actual reset.</p> <p>001: Allowed - The counter enable signal CNT_EN is used as a trigger output (TRGO). Sometimes it is necessary to start multiple timers at the same time or to control a window from a timer. The counter enable signal is generated by a logical or of the CEN control bits and the trigger input signal in gated mode. When the counter enable signal is controlled on the trigger input, there is a delay on TRGO unless master/slave mode is selected (see description of the MSM bit in the TIMx_SMCR register).</p> <p>010: Update - The update event is selected as a trigger input (TRGO). For example, a master timer clock may be used as a prescaler for a slave timer.</p> <p>011: Compare Pulse - Once a capture has occurred or a successful comparison has occurred, the trigger output sends a positive pulse (TRGO) when the CC1IF flag is to be set (i.e. is it already high).</p> <p>100: Compare - The OC1REF signal is used as a trigger output (TRGO).</p> <p>101: Compare - The OC2REF signal is used as a trigger output (TRGO).</p> <p>110: Compare - The OC3REF signal is used as a trigger output (TRGO).</p> <p>111: Compare - The OC4REF signal is used as a trigger output (TRGO).</p> <p>Attention:</p> <p>1. The clocks of the slave timer and ADC must first be enabled to receive signals from the master timer and should not be changed while receiving.</p>

Bit	Name	R/W	Reset Value	Function
				2. If the master and slave timers are not on the same bus, the master mode should be configured to a width that can be picked up by the slave timer.
3	CCDS	RW	0	DMA selection for capture/compare 0: DMA request for CCx is sent when a CCx event occurs. 1: Send a DMA request for CCx when an update event occurs.
2	CCUS	RW	0	Capture/comparison control update selection 0: If the capture/compare control bits are preloaded (CCPC=1), they can only be updated by setting the COM bit. 1: If the capture/compare control bits are preloaded (CCPC=1), they can be updated by setting the COM bit or a rising edge on TRGI. Note: This bit only works for channels with complementary outputs.
1	Res	-	0	Reserved, always reads 0.
0	CCPC	RW	0	Capture/compare preload control bits 0: CCxE, CCxNE and OCxM bits are not preloaded. 1: The CCxE, CCxNE, and OCxM bits are preloaded; when this bit is set, they are updated only when the COM bit is set. Note: This bit only works for channels with complementary outputs.

21.4.3. TIM1 Slave Mode Control Register (TIM1_SMCR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS [1:0]		ETF [3:0]				MSM	TS [2:0]			OCCS	SMS [2:0]		
RW	RW	RW		RW				RW	RW			RW	RW		

Bit	Name	R/W	Reset Value	Function
31: 16	Res	-	-	Res
15	ETP	RW	0	External trigger polarity. This bit selects whether ETR or the reverse of ETR is used as the trigger operation.

Bit	Name	R/W	Reset Value	Function
				<p>0: ETR is not inverted, high or rising edge active</p> <p>1: ETR reverse, active low or falling edge</p>
14	ECE	RW	0	<p>External clock enable bit. This bit enables external clock mode 2</p> <p>0: Disable external clock mode 2;</p> <p>1: Enable external clock mode 2. The counter is driven by any valid edge on the ETRF signal.</p> <p>Note 1: Setting the ECE bit has the same effect as selecting External Clock Mode 1 and connecting TRGI to ETRF (SMS=111 and TS=111).</p> <p>Note 2: The following slave modes can be used in conjunction with External Clock Mode 2: Reset Mode, Gated Mode, and Trigger Mode; however, TRGI cannot be connected to ETRF at this time (TS bit cannot be '111').</p> <p>Note 3: When External Clock Mode 1 and External Clock Mode 2 are both enabled, the external clock input is ETRF.</p>
13: 12	ETPS [1:0]	RW	00	<p>Externally triggered prescaler. The external trigger input signal ETR frequency must be at most 1/4 of the TIM1CLK frequency. The prescaler can be enabled to reduce the frequency of the ETRP.</p> <p>00: Prescaler off</p> <p>01: 2 crossover of ETRP frequency</p> <p>10: 4 divisions of the ETRP frequency</p> <p>11: ETRP frequency in 8 divisions</p>
11: 8	ETF [3:0]	RW	0000	<p>External trigger filtering. These bits define the frequency of the sampled ETRP signal and the length of the digital filter applied to the ETRP. This digital filtering consists of an event counter where N consecutive events are required to make the output edge valid.</p> <p>0000: no filter, sampled at fDTS</p> <p>0001: fSAMPLING=fCK_INT, N=2</p> <p>0010: fSAMPLING=fCK_INT, N=4</p> <p>0011: fSAMPLING=fCK_INT, N=8</p> <p>0100: fSAMPLING=fDTS/2, N=6</p> <p>0101: fSAMPLING=fDTS/2, N=8</p> <p>0110: fSAMPLING=fDTS/4, N=6</p> <p>0111: fSAMPLING=fDTS/4, N=8</p> <p>1000: fSAMPLING=fDTS/8, N=6</p> <p>1001: fSAMPLING=fDTS/8, N=8</p> <p>1010: fSAMPLING=fDTS/16, N=5</p>

Bit	Name	R/W	Reset Value	Function
				1011: fSAMPLING=fDTS/16, N=6 1100: fSAMPLING=fDTS/16, N=8 1101: fSAMPLING=fDTS/32, N=5 1110: fSAMPLING=fDTS/32, N=6 1111: fSAMPLING=fDTS/32, N=8
7	MSM	RW	0	Master/Slave mode 0: No effect 1: Events on the trigger input (TRGI) are delayed to allow for synchronization (via TRGO) between the current timer (via TRGO) and its current and slave timers. This is useful when requiring synchronization of several timers to a single external event
6: 4	TS [2:0]	RW	000	Trigger Select, these 3 bits select the trigger input used to synchronize the counter. 000: TIM15 (ITR0) 001: TIM2 (ITR1) 010: TIM3 (ITR2) 011: TIM17 (ITR3) 100: Edge detector for TI1 (TI1F_ED) 101: Filtered Timer Input 1 (TI1FP1) 110: Filtered Timer Input 2 (TI2FP2) 111: External Trigger Input (ETRF) Note: To avoid incorrect edge detection during signal transitions, these bits must be modified when they are not used (e.g. SMS=000)
3	OCCS	RW	0	OCREF clears the select bit. This bit is used to select the clear source for OCREF. 0: OCREF_CLR_INT connected to OCREF_CLR input 1: OCREF_CLR_INT connected to ETRF
2: 0	SMS [2:0]	RW	000	Select from mode. When an external signal is selected, the active edge of the trigger signal (TRGI) is related to the selected external input polarity (see description of the Input Control Register and Control Register) 000: Disable slave mode If CEN=1, the prescaler is driven directly by the internal clock. 001: Encoder mode 1 Depending on the level of TI1FP1, the counter counts up/down on the edge of TI2FP2. 010: Encoder mode 2

Bit	Name	R/W	Reset Value	Function
				<p>Depending on the level of TI2FP2, the counter counts up/down on the edge of TI1FP1.</p> <p>011: Encoder mode 3</p> <p>Synthesis of encoder modes 1 and 2</p> <p>100: Reset mode</p> <p>The rising edge of the selected trigger input (TRGI) reinitializes the counter and generates a signal to update the register.</p> <p>101: Gated mode</p> <p>When the trigger input (TRGI) is high, the counter's clock turns on. Once the trigger input goes low, the counter stops (but does not reset). The start and stop of the counter is controlled.</p> <p>110: Trigger mode</p> <p>The counter is started (but not reset) on the rising edge of the trigger input TRGI, and only the start of the counter is controlled.</p> <p>111: External clock mode 1</p> <p>The rising edge of the selected trigger input (TRGI) drives the counter.</p> <p>Note: Do not use gating mode if TI1F_ED is selected as the trigger input (TS=100). This is because TI1F_ED outputs a pulse each time TI1F changes, however the gated mode is to check the level of the trigger input.</p> <p>Note: Do not use UEV as the TRGO output signal in encoder mode, (i.e. MMS cannot be configured to 010)</p>

Table 21-2 TIM1 internal trigger connection

Slave TIM	ITR0 (TS=000)	ITR1 (TS=001)	ITR2 (TS=010)	ITR3 (TS=011)
TIM1	TIM15_TRGO	TIM2_TRGO	TIM3_TRGO	TIM17_OC

21.4.4. TIM1 DMA/Interrupt Enable Register (TIM1_DIER)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Re	TD	COM	CC4D	CC3D	CC2D	CC1D	UD	BI	TI	COMI	CC4I	CC3I	CC2I	CC1I	UI
s	E	DE	E	E	E	E	E	E	E	E	E	E	E	E	E
-	RW														

Bit	Name	R/W	Reset Value	Function
31: 15	Res	-	-	Res
14	TDE	RW	0	TDE: Allow triggering of DMA requests 0: Disable triggering of DMA request 1: Allow triggering of DMA requests
13	COMDE	RW	0	COMDE: Allow COM's DMA request 0: DMA request from COM is disabled 1: Allow COM's DMA request
12	CC4DE	RW	0	CC4DE: Allows capture/comparison of DMA requests for 4 0: Disable capture/compare 4 DMA requests 1: Allow capture/comparison of DMA requests for 4
11	CC3DE	RW	0	CC3DE: Allows capture/comparison of DMA requests for 3 0: Disable capture/compare 3 DMA requests 1: Allow capture/comparison of 3 DMA requests
10	CC2DE	RW	0	CC2DE: Allows capture/comparison of 2 DMA requests 0: Disable capture/compare 2 DMA requests 1: Allow capture/compare 2 DMA requests
9	CC1DE	RW	0	CC1DE: Allow capture/compare 1 DMA requests 0: Capture/compare 1 DMA request disabled 1: Allow capture/compare 1 for DMA requests
8	UDE	RW	0	UDE: DMA request to allow updates 0: DMA requests for updates are disabled 1: Allow updated DMA requests
7	BIE	RW	0	BIE: Allow brake interruption 0: Brake interrupt prohibited 1: Allow brake interruptions
6	TIE	RW	0	TIE: Allowed to trigger an interrupt 0: Disable interrupt triggering 1: Allow triggering of interrupts
5	COMIE	RW	0	COMIE: Allow COM interrupt 0: Disable COM interrupt 1: Allow COM interrupt
4	CC4IE	RW	0	CC4IE: Allow capture/compare 4 interrupts 0: Capture/compare 4 interrupt disabled 1: Capture/compare 4 interrupts allowed
3	CC3IE	RW	0	CC3IE: Allow capture/compare 3 interrupts 0: Capture/compare 3 interrupt disabled 1: Capture/compare 3 interrupts allowed
2	CC2IE	RW	0	CC2IE: Capture/Compare 2 interrupts allowed 0: Capture/compare 2 interrupt disabled

Bit	Name	R/W	Reset Value	Function
				1: Capture/compare 2 interrupts allowed
1	CC1IE	RW	0	CC1IE: Capture/Compare 1 interrupt allowed 0: Capture/compare 1 interrupt disabled 1: Allow capture/compare 1 interrupt
0	UIE	RW	0	UIE: Allow update interruptions 0: Disable update interrupt 1: Allow updates to be interrupted

21.4.5. TIM1 status register (TIM1_SR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res			Res					ic4i f	ic3i f	ic2if	ic1if	ic4i r	ic3ir	ic2ir	ic1i r
-								RC_W0							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res			CC4O F	CC3O F	CC2O F	CC1O F	Re s	BI F	TIF	COMI F	CC4I F	CC 3IF	CC2 IF	CC1I F	UI F
-			RC_W0				-	RC_W0							

Bit	Name	R/W	Reset Value	Function
31: 13	Res	-	-	Res
23	IC4IF	RC_W0	0	Capture 4 flags on falling edge See IC1IF description.
22	IC3IF	RC_W0	0	Capture 3 flag on falling edge See IC1IF description.
21	IC2IF	RC_W0	0	Capture 2 flag on falling edge See IC1IF description.
20	IC1IF	RC_W0	0	Capture 1 flag on falling edge This flag can be set to 1 by hardware only if the corresponding channel is configured for input capture and the capture event is triggered by a falling edge. It is cleared '0' by software or by reading TIMx_CCR1. 0: No falling edge capture is generated; 1: A falling edge capture event occurs.
19	IC4IR	RC_W0	0	Rising edge capture 4 flags See IC1IR for description.
18	IC3IR	RC_W0	0	Rising edge capture 3 flag

Bit	Name	R/W	Reset Value	Function
				See IC1IR for description.
17	IC2IR	RC_W0	0	Rising edge capture 2 flag See IC1IR for description.
16	IC1IR	RC_W0	0	Rising edge capture 1 flag This flag can be set to 1 by hardware only if the corresponding channel is configured for input capture and the capture event is triggered by a rising edge. It is cleared '0' by software or by reading TIMx_CCR1. 0: No rising edge capture is generated; 1: A rising edge capture event occurs.
15: 13	Res	-	-	Res
12	CC4OF	RC_W0	0	Capture/Compare4 Overcapture Marker See CC1OF description
11	CC3OF	RC_W0	0	Capture/Compare3 Overcapture Marker See CC1OF description
10	CC2OF	RC_W0	0	Capture/Compare2 Overcapture Marker See CC1OF description
9	CC1OF	RC_W0	0	Capture/Compare1 Overcapture Marker This flag can be set to 1 by hardware only if the corresponding channel is configured for input capture. Writing a 0 clears the bit. 0: No overcapture is generated; 1: When CC1OF is set to 1, the value of the counter is captured to the TIMx_CCR1 register when CC1IF is 1.
8	Res	-	-	Res
7	BIF	RC_W0	0	Brake interrupt marker Once the brake input is valid, by the hardware for that position 1. If the brake input is invalid, this bit can be cleared to 0 by software. 0: No brake events are generated; 1: A valid level is detected on the brake input.
6	TIF	RC_W0	0	Trigger interrupt flag When a trigger event occurs (when the slave mode controller is in a mode other than the gated mode, a trigger event is detected at the TRGI input), a trigger event is generated. Effective edge, or or either edge in gated mode) by hardware for this position 1. It is cleared 0 by the software. 0: No trigger event is generated;

Bit	Name	R/W	Reset Value	Function
				1: Trigger interrupt waiting for response
5	COMIF	RC_W0	0	<p>COM interrupt marker</p> <p>Once a COM event is generated (when CCxE, CCxNE, OCxM have been updated) this bit is set to 1 by hardware. It is cleared 0 by the software.</p> <p>0: No COM events are generated; 1: COM interrupt waiting for response</p>
4	CC4IF	RC_W0	0	<p>Capture/Compare4 Interrupt Marker</p> <p>Refer to CC1IF description</p>
3	CC3IF	RC_W0	0	<p>Capture/Compare3 Interrupt Marker</p> <p>Refer to CC1IF description</p>
2	CC2IF	RC_W0	0	<p>Capture/Compare2 Interrupt Marker</p> <p>Refer to CC1IF description</p>
1	CC1IF	RC_W0	0	<p>Capture/Compare1 Interrupt Marker</p> <p>If channel CC1 is configured for output mode: This bit is set to 1 by hardware when the counter value matches the comparison value. However, CMS needs to be judged in centrosymmetric mode; when the comparison value is greater than the overloaded value, it will be set to 1 on upflow (upward counting mode, center-aligned counting mode) or downflow (downward counting mode). It is cleared 0 by the software. 0: No match occurred; 1: The value of TIMx_CNT matches the value of TIMx_CCR1.</p> <p>If channel CC1 is configured for input mode: This bit is set to 1 by hardware when a capture event occurs, and it is cleared to 0 by software or by reading TIMx_CCR1. 0: No input capture is generated; 1: Input capture is generated and the counter value has been loaded into TIMx_CCR1 (an edge of the same polarity as selected is detected on IC1). Note: When CEN is turned on, this bit is also set.</p>
0	UIF	RC_W0	0	<p>Updating the interrupt flag</p> <p>This bit is set to 1 by hardware when an update event is generated. It is cleared 0 by the software.</p> <p>0: No update events are generated;</p>

Bit	Name	R/W	Reset Value	Function
				<p>1: Update event waiting for response. This bit is set to 1 by hardware when the register is updated.</p> <p>- If UDIS=0 in the TIMx_CR1 register, an update event is generated when REP_CNT=0 (when the repeat down counter overflows or underflows);</p> <p>- If UDIS=0 and URS=0 in the TIMx_CR1 register, the update event is generated when UG=1 in the TIMx_EGR register.</p> <p>pieces (software reinitialization of the CNT);</p> <p>- If UDIS=0 and URS=0 in the TIMx_CR1 register, the update event is generated when the CNT is reinitialized by the trigger event</p> <p>Pieces. (Reference Slave Mode Control Register (TIMx_SMCR))</p>

21.4.6. TIM1 Event Generation Register (TIM1_EGR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
-	-	-	--	-	-	-	-	W	W	W	W	W	W	W	W

Bit	Name	R/W	Reset Value	Function
31: 8	Res	-	-	Res
7	BG	W	0	<p>Generate a braking event</p> <p>This bit is set to 1 by software to generate a brake event, which is automatically cleared to 0 by hardware.</p> <p>0: No action;</p> <p>1: Generate a brake event. At this time, MOE=0, BIF=1, and the corresponding interrupt and DMA will be generated if they are turned on.</p>
6	TG	W	0	<p>generate a trigger event</p> <p>This bit is set to 1 by software to generate a trigger event that is automatically cleared to 0 by hardware.</p> <p>0: No action;</p>

Bit	Name	R/W	Reset Value	Function
				1: TIF=1 in the TIMx_SR register generates the corresponding interrupt and DMA if they are turned on.
5	COMG	W	0	<p>Capture/compare events to generate control updates</p> <p>This bit is set to 1 by software and cleared to 0 automatically by hardware.</p> <p>0: No action;</p> <p>1: When CCPC = 1, the CCxE, CCxNE, and OCxM bits are allowed to be updated.</p> <p>Note: This bit is only valid for channels with complementary outputs.</p>
4	CC4G	W	0	<p>Generate capture/compare 4 events</p> <p>Refer to CC1G description</p>
3	CC3G	W	0	<p>Generate Capture/Compare 3 events</p> <p>Refer to CC1G description</p>
2	CC2G	W	0	<p>Generate Capture/Compare 2 events</p> <p>Refer to CC1G description</p>
1	CC1G	W	0	<p>Generate Capture/Compare 1 events</p> <p>This bit is set to 1 by software to generate a capture/compare event that is automatically cleared to 0 by hardware.</p> <p>0: No action;</p> <p>1: Generate a capture/compare event on channel CC1:</p> <p>If channel CC1 is configured as an output:</p> <p>Set CC1IF=1 to generate the corresponding interrupt and DMA if it is turned on.</p> <p>If channel CC1 is configured as an input:</p> <p>The current counter value is captured to the TIMx_CCR1 register, CC1IF=1 is set, and the corresponding interrupt and DMA are generated if they are turned on. If CC1IF is already 1, set CC1OF=1.</p>
0	UG	W	0	<p>Generate update events. This bit is set to 1 by software and cleared to 0 automatically by hardware.</p> <p>0: No action;</p> <p>1: Reinitialize the counter and generate an update event.</p> <p>Note: The prescaler counter is also cleared to 0 (but the prescaler</p> <p>(The coefficients are unchanged). The counter is cleared to 0 if in centrosymmetric mode or DIR=0 (counting up), and the counter is loaded with the value of TIMx_ARR if DIR=1 (counting down).</p>

21.4.7. TIM1 Capture/Compare Mode Register 1 (TIM1_CM1)

Address offset: 0x18

Reset value: 0x0000 0000

Output comparison mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2: 0]			OC2PE	CO2FE	CC2S[1: 0]		OC1CE	OC1M[2: 0]			OC1PE	OC1FE	CC1S[1: 0]	
RW															

Bit	Name	R/W	Reset Value	Function
31: 16	Res	-	-	Res
15	OC2CE	RW	0	Output compare 2 clear 0 enable
14: 12	OC2M[2: 0]	RW	000	Output Compare 2 Mode Selection
11	OC2PE	RW	0	Output compare 2 preload enable
10	OC2FE	RW	0	Output compare 2 fast enable
9: 8	CC2S[1: 0]	RW	00	<p>Capture/Compare 2 Selection.</p> <p>This bit defines the direction of the channel (input/output), and the selection of the input pin:</p> <p>00: The CC2 channel is configured as an output;</p> <p>01: CC2 channel is configured as an input and IC2 is mapped on TI2;</p> <p>10: The CC2 channel is configured as an input and IC2 is mapped on TI1;</p> <p>11: The CC2 channel is configured as an input and IC2 is mapped on TRC. This mode works only when the internal trigger input is selected.</p> <p>(selected by the TS bit of the TIMx_SMCR register).</p> <p>Note: CC2S is writable only when the channel is closed (CC2E=0 in the TIMx_CCER register).</p>
7	OC1CE	RW	0	<p>Output compare 1 clear 0 enable</p> <p>0: OC1REF is not affected by the ETRF input;</p> <p>1: Clear OC1REF=0 as soon as the ETRF input is detected high.</p>
6: 4	OC1M[2: 0]	RW	00	Output Comparison 1 Mode

Bit	Name	R/W	Reset Value	Function
				<p>These 3 bits define the action of the output reference signal OC1REF, which determines the values of OC1, OC1N. OC1REF is active high, while the active levels of OC1 and OC1N depend on the CC1P and CC1NP bits.</p> <p>000: Frozen. The comparison between the output comparison register TIMx_CCR1 and the counter TIMx_CNT does not work for OC1REF;</p> <p>001: Set channel 1 to active level when matching. Forces OC1REF high when counter TIMx_CNT has the same value as Capture/Compare Register 1 (TIMx_CCR1).</p> <p>010 : Set channel 1 to invalid level when matching. Force OC1REF low when the value of counter TIMx_CNT is the same as Capture/Compare Register 1 (TIMx_CCR1).</p> <p>011: Flip. Flips the level of OC1REF when TIMx_CCR1 = TIMx_CNT.</p> <p>100: Forced to an invalid level. Force OC1REF to be low.</p> <p>101: Forced to active level. Force OC1REF to be high.</p> <p>110: PWM Mode 1- In upward counting, channel 1 is valid once TIMx_CNT<TIMx_CCR1, otherwise it is invalid; in downward counting, channel 1 is invalid once TIMx_CNT>TIMx_CCR1 (OC1REF=0), otherwise it is valid (OC1REF=1). 1).</p> <p>111: PWM Mode 2- In up count, channel 1 is invalid once TIMx_CNT<TIMx_CCR1, otherwise it is valid; in down count, channel 1 is valid once TIMx_CNT>TIMx_CCR1, otherwise it is invalid.</p> <p>Note 1: This bit cannot be modified once the LOCK level is set to 3 (LOCK bit in the TIMx_BDTR register) and CC1S=00 (the channel is configured as an output).</p> <p>Note 2: In PWM mode 1 or PWM mode 2, the OC1REF level changes only when the comparison result is changed or when switching from freeze mode to PWM mode in the output comparison mode.</p>
3	OC1PE	RW	0	<p>Output compare 1 preload enable</p> <p>0: Disables the preload function of the TIMx_CCR1 register, which can be written to the TIMx_CCR1 register at any time and the new value takes effect immediately.</p> <p>1: Enable the preload function of TIMx_CCR1 register, the read and write operation only operates on the preloaded</p>

Bit	Name	R/W	Reset Value	Function
				<p>register, the preloaded value of TIMx_CCR1 is loaded into the current register when the update event arrives.</p> <p>Note 1: This bit cannot be modified once the LOCK level is set to 3 (LOCK bit in the TIMx_BDTR register) and CC1S=00 (the channel is configured as an output).</p>
2	OC1FE	RW	0	<p>Output Compare 1 Fast Enable</p> <p>This bit is used to speed up the response of the CC output to trigger input events.</p> <p>0: Depending on the value of the counter and CCR1, CC1 operates normally, even if the trigger is open. The minimum delay to activate the CC1 output is 5 clock cycles when the input of the flip-flop has an active edge.</p> <p>1: The active edge of the input to the trigger acts as if a comparison match has occurred. Therefore, OC is set to the comparison level and the</p> <p>Irrelevant to the comparison of results. The delay between the active edge of the sampling flip-flop and the CC1 output is reduced to 3 clock cycles.</p> <p>OCFE's only work when the channel is configured for PWM1 or PWM2 mode.</p>
1: 0	CC1S[1: 0]	RW	00	<p>Capture/Compare1 Select.</p> <p>These 2 bits define the direction of the channel (input/output), and the selection of the input pin:</p> <p>00: CC1 channel is configured as an output;</p> <p>01: CC1 channel is configured as an input and IC1 is mapped on TI1;</p> <p>10: CC1 channel is configured as an input and IC1 is mapped on TI2;</p> <p>11: CC1 channel is configured as an input and IC1 is mapped on TRC. This mode works only when the internal trigger input is selected.</p> <p>(selected by the TS bit of the TIMx_SMCR register).</p> <p>Note: CC1S is writable only when the channel is closed (CC1E=0 in the TIMx_CCER register).</p>

Input Capture Mode:

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

-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC2F[3: 0]				IC2PSC[1: 0]		CC2S[1: 0]		IC1F[3: 0]				IC1PSC[1: 0]		CC1S[1: 0]	
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 16	Res	-	-	Res
15: 12	IC2F	RW	0000	Input Capture 2 Filter
11: 10	IC2PSC[1: 0]	RW	00	Capture/Compare 2 Prescaler
9: 8	CC2S[1: 0]	RW	0	<p>Capture/Compare 2 Selection.</p> <p>These 2 bits define the direction of the channel (input/output), and the selection of the input pin:</p> <p>00: The CC2 channel is configured as an output;</p> <p>01: CC2 channel is configured as an input and IC2 is mapped on TI2;</p> <p>10: The CC2 channel is configured as an input and IC2 is mapped on TI1;</p> <p>11: The CC2 channel is configured as an input and IC2 is mapped on TRC. This mode works only when the internal trigger input is selected.</p> <p>(selected by the TS bit of the TIMx_SMCR register).</p> <p>Note: CC2S is writable only when the channel is closed (CC2E=0 in the TIMx_CCER register).</p>
7: 4	IC1F[3: 0]	RW	0000	<p>Input Capture 1 Filter</p> <p>These bits define the sampling frequency of the TI1 input and the digital filter length. The digital filter consists of an event counter that</p> <p>It records N events and then produces a jump in output:</p> <p>0000: no filter, sampled at fDTS</p> <p>0001: fSAMPLING=fCK_INT, N=2</p> <p>0010: fSAMPLING=fCK_INT, N=4</p> <p>0011: fSAMPLING=fCK_INT, N=8</p> <p>0100: fSAMPLING=fDTS/2, N=6</p> <p>0101: fSAMPLING=fDTS/2, N=8</p> <p>0110: fSAMPLING=fDTS/4, N=6</p> <p>0111: fSAMPLING=fDTS/4, N=8</p> <p>1000: fSAMPLING=fDTS/8, N=6</p> <p>1001: fSAMPLING=fDTS/8, N=8</p> <p>1010: fSAMPLING=fDTS/16, N=5</p>

Bit	Name	R/W	Reset Value	Function
				1011: fSAMPLING=fDTS/16, N=6 1100: fSAMPLING=fDTS/16, N=8 1101: fSAMPLING=fDTS/32, N=5 1110: fSAMPLING=fDTS/32, N=6 1111: fSAMPLING=fDTS/32, N=8
3: 2	IC1PSC[1: 0]	RW	00	Capture/Compare 1 prescaler 00: No prescaler, each edge detected on the capture input triggers a capture; 01: Capture is triggered every 2 events; 10: Capture is triggered every 4 events; 11: Capture is triggered every 8 events.
1: 0	CC1S[1: 0]	RW	00	CC1S[1:0]: capture/compare 1 selection. These 2 bits define the direction of the channel (input/output), and the selection of the input pin: 00: CC1 channel is configured as an output; 01: CC1 channel is configured as an input and IC1 is mapped on TI1; 10: CC1 channel is configured as an input and IC1 is mapped on TI2; 11: CC1 channel is configured as an input and IC1 is mapped on TRC. This mode works only when the internal trigger input is selected. (selected by the TS bit of the TIMx_SMCR register). Note: CC1S is writable only when the channel is closed (CC1E=0 in the TIMx_CCER register).

21.4.8. TIM1 Capture/Compare Mode Register 2 (TIM1_CMR2)

Address offset: 0x1C

Reset value: 0x0000 0000

Output comparison mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4C E	OC4M[2: 0]			OC4P E	CO4F E	CC4S[1 : 0]		OC3C E	OC3M[2: 0]			OC3P E	OC 3FE	CC3S[1: 0]	
IC4F[3: 0]				IC4PSC[1: 0]				IC3F[3: 0]			IC3PSC[1: 0]				
RW															

Bit	Name	R/W	Reset Value	Function
31: 16	Res	-	-	Res
15	OC4CE	RW	0	Output compare 4 clear 0 enable
14: 12	OC4M[2: 0]	RW	000	Output Comparison 4 Mode
11	OC4PE	RW	0	Output Compare 4 Preload Enable
10	OC4FE	RW	0	Output Compare 4 Fast Enable
9: 8	CC4S[1: 0]	RW	00	<p>Capture/Compare 4 Selection.</p> <p>This bit defines the direction of the channel (input/output), and the selection of the input pin:</p> <p>00: The CC4 channel is configured as an output;</p> <p>01: CC4 channel is configured as an input and IC4 is mapped on TI4;</p> <p>10: The CC4 channel is configured as an input and IC4 is mapped on TI3;</p> <p>11: The CC4 channel is configured as an input and IC4 is mapped on TRC. This mode operates only when the internal trigger input is selected (by the TS bit of the TIMx_SMCR register).</p> <p>Note: CC4S is writable only when the channel is closed (CC4E=0 in the TIMx_CCER register).</p>
7	OC3CE	RW	0	Output compare 3 clear 0 enable
6: 4	OC3M[2: 0]	RW	00	Output Comparison 3 Mode
3	OC3PE	RW	0	Output Compare 3 Preload Enable
2	OC3FE	RW	0	Output Compare 3 Fast Enable
1: 0	CC3S[1: 0]	RW	00	<p>Capture/Compare 3 options.</p> <p>These 2 bits define the direction of the channel (input/output), and the selection of the input pin:</p> <p>00: The CC3 channel is configured as an output;</p> <p>01: CC3 channel is configured as an input and IC3 is mapped on TI3;</p> <p>10: CC3 channel is configured as an input and IC3 is mapped on TI4;</p> <p>11: CC3 channel is configured as an input and IC3 is mapped on TRC. This mode works only when the internal trigger input is selected.</p> <p>(selected by the TS bit of the TIMx_SMCR register).</p> <p>Note: CC3S is writable only when the channel is closed (CC3E=0 in the TIMx_CCER register).</p>

Input Capture Mode:

Bit	Name	R/W	Reset Value	Function
-----	------	-----	-------------	----------

31: 16	Res	-	-	Res
15: 12	IC4F[3: 0]	RW	0000	Input Capture 4 Filter
11: 10	IC4PSC[1: 0]	RW	00	Capture/compare 4 prescalers
9: 8	CC4S[1: 0]	RW	00	<p>Capture/Compare 4 Selection.</p> <p>These 2 bits define the direction of the channel (input/output), and the selection of the input pin:</p> <p>00: The CC4 channel is configured as an output;</p> <p>01: CC4 channel is configured as an input and IC4 is mapped on TI4;</p> <p>10: The CC4 channel is configured as an input and IC4 is mapped on TI3;</p> <p>11: The CC4 channel is configured as an input and IC4 is mapped on TRC. This mode operates only when the internal trigger input is selected (by the TS bit of the TIMx_SMCR register).</p> <p>Note: CC4S is writable only when the channel is closed (CC4E=0 in the TIMx_CCER register).</p>
7: 4	IC3F[3: 0]	RW	0000	Capture/Compare 3 Filter
3: 2	IC3PSC[1: 0]	RW	00	Capture/Compare 3 Prescaler
1: 0	CC3S[1: 0]	RW	00	<p>Capture/Compare 3 options.</p> <p>These 2 bits define the direction of the channel (input/output), and the selection of the input pin:</p> <p>00: The CC3 channel is configured as an output;</p> <p>01: CC3 channel is configured as an input and IC3 is mapped on TI3;</p> <p>10: CC3 channel is configured as an input and IC3 is mapped on TI4;</p> <p>11: CC3 channel is configured as an input and IC1 is mapped on TRC. This mode works only when the internal trigger input is selected.</p> <p>(selected by the TS bit of the TIMx_SMCR register).</p> <p>Note: CC3S is writable only when the channel is closed (CC3E=0 in the TIMx_CCER register).</p>

21.4.9. TIM1 Capture/Compare Enable Register (TIM1_CCER)

Address offset: 0x20

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
-	-	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 14	Res	-	0	Reserved, always 0
13	CC4P	RW	0	Capture/compare 4 output polarity. Refer to the description of CC1P.
12	CC4E	RW	0	Capture/Compare 4 output enable. Refer to the description of CC1E.
11	CC3NP	RW	0	Capture/compare 3 complementary output polarities. Refer to the description of CC1NP.
10	CC3NE	RW	0	Capture/compare 3 complementary output enable. Refer to the description of CC1NE.
9	CC3P	RW	0	Capture/compare 3 output polarity. Refer to the description of CC1P.
8	CC3E	RW	0	Capture/Compare 3 output enable. Refer to the description of CC1E.
7	CC2NP	RW	0	Capture/compare 2 complementary output polarities. Refer to the description of CC1NP.
6	CC2NE	RW	0	Capture/Compare 2 complementary output enable. Refer to the description of CC1NE.
5	CC2P	RW	0	Capture/Compare 2 Output Polarity. Refer to the description of CC1P.
4	CC2E	RW	0	Capture/Compare 2 output enable. Refer to the description of CC1E.
3	CC1NP	RW	0	Capture/compare 1 complementary output polarity 0: OC1N high level active 1: OC1N active-low Note: This bit cannot be modified once the LOCK level (LCK bit in the TIMx_BDTR register) is set to 3 or 2 and CC1S=00 (channel configured for output).
2	CC1NE	RW	0	Capture/compare 1 complementary output enable 0: OC1N prohibits output 1: OC1N signal output to the corresponding output pin When the CC1 channel is configured as an output, the OC1N output level is determined by a combination of the

Bit	Name	R/W	Reset Value	Function
				<p>MOE, OSS1, OSSR, OIS1, OIS1N, CC1E, and CC1NE bits, as shown in the table below</p> <p>For complementary output channels, this bit is preloaded. If the CCPC bit in the TIMx_CR2 register is set, the actual valid bits of CC1E are loaded with the preloaded value only when a com event occurs.</p>
1	CC1P	RW	0	<p>Capture/Compare 1 Output Polarity</p> <p>The CC1 channel is configured as an output:</p> <p>0: OC1 active high</p> <p>1: OC1 active low</p> <p>The CC1 channel is configured as an input:</p> <p>The CC1NP/CC1P bits select the polarity of TI1FP1 and TI2FP1 used as trigger or capture signals.</p> <p>00: No inversion/rising edge:</p> <p>TIxFP1 rising edge active (capture, trigger in reset mode, external clock, or trigger mode);</p> <p>TIxFP1 is not inverted (gated mode, encoder mode).</p> <p>01: Inverted/falling edge:</p> <p>TIxFP1 falling edge active (capture, trigger in reset mode, external clock, or trigger mode);</p> <p>TIxFP1 inversion (gated mode, encoder mode).</p> <p>10: Reserved, do not use this configuration.</p> <p>11: Non-inverting/double-edge</p> <p>TIxFP1 is valid on both rising and falling edges (capture, triggered in reset mode, externally clocked, or in trigger mode);</p> <p>TIxFP1 is not inverted (gated mode). This configuration cannot be applied in encoder mode.</p> <p>Notes:</p> <p>1. For complementary output channels, this bit is preloaded. If the CCPC bit in the TIMx_CR2 register is set, the actual valid bits of CC1P are loaded with the preloaded value only when a com event occurs.</p> <p>2. Once the LOCK level (LCKK bit in the TIMx_BDTR register) is set to 3 or 2, this bit cannot be modified</p>
0	CC1E	RW	0	<p>Capture/compare 1 output enable</p> <p>0: Capture mode off/OC1 output disabled</p>

Bit	Name	R/W	Reset Value	Function
				<p>1: Capture mode on/OC1 signal output to corresponding output pin</p> <p>When the CC1 channel is configured as an output, the OC1 output level is determined by a combination of the MOE, OSSI, OSSR, OIS1, OIS1N, CC1E, and CC1NE bits, as shown in the following table</p> <p>Notes:</p> <p>For complementary output channels, this bit is preloaded.</p> <p>If the CCPC bit in the TIMx_CR2 register is set, the actual valid bits of CC1E are loaded with the preloaded value only when a com event occurs.</p>

Table 21-3 Output Control for Complementary OCx and OCxN Channels with Interrupt Functions

control bits					output state	
MOE	OSSI	OSSR	CCxE	CCxNE	OCx output state	OCxN output state
1	x	0	0	0	Output inhibit (disconnected from timer) OCx=0, OCx_EN=0	Output inhibit (disconnected from timer) OCxN=0. OCxN_EN=0
		0	0	1	Output inhibit (disconnected from timer) OCx=0, OCx_EN=0	OCxREF+ polarity. OCxN=OCREF xor CCxNP, the OCxN_EN=1
		0	1	0	OCxREF+ polarity. OCx = OCREF xor CCxP, OCx_EN=1	Output inhibit (disconnected from timer) OCxN=0. OCxN_EN=0
		0	1	1	OCxREF+Polarity+Deadband,, OCx_EN=1	OCxREF+Polarity+Deadband. OCxN_EN=1
		1	0	0	Output inhibit (disconnected from timer) OCx=CCxP. OCx_EN=0	Output inhibit (disconnected from timer) OCxN=CCxNP. OCxN_EN=0
		1	0	1	Off state (output enabled and invalid level) OCx=CCxP. OCx_EN=1	OCxREF+ polarity. OCxN=OCREF xor CCxNP, the OCxN_EN=1
		1	1	0	OCxREF+ polarity. OCx = OCREF xor CCxP,	Off state (output enabled and invalid level)

control bits					output state	
MOE	OSSI	OSSR	CCxE	CCxNE	OCx output state	OCxN output state
					OCx_EN=1	OCxN=CCxNP. OCxN_EN=1
		1	1	1	OCxREF+Polarity+Deadband,, OCx_EN=1	OCxREF+Polarity+Deadband. OCxN_EN=1
0	0	x	0	0	Output inhibit (disconnected from timer)	
	0		0	1		
	0		1	0		
	0		1	1		
	1		0	0		
	1		0	1	Off state (output enabled and invalid level) Asynchronous: OCx=CCxP with OCx_EN=1 and OCxN=CCxNP with OCx_EN=1; If the clock exists: after a dead time OCx = OISx and OCxN = OISxN, assuming that OISx and OISxN do not both correspond to valid levels of OCx and OCxN	
	1		1	0		
	1		1	1		

21.4.10. TIM1 Counter (TIM1_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 16	Res	-	-	Res
15: 0	CNT [15:0]	RW	0	Counter value

21.4.11. TIM1 prescaler (TIM1_PSC)

Address offset: 0x28

Reset value: 0x0000 0000

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

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

Bit	Name	R/W	Reset Value	Function
31: 16	Res	-	-	Res
15: 0	PSC [15:0]	RW	0	<p>Prescaler value</p> <p>The counter clock frequency (CK_CNT) is equal to $f_{CK_PSC}/(PSC[15:0]+1)$.</p> <p>The PSC contains the value loaded into the current pre-scaler register when the update event is generated; the update event includes the counter</p> <p>Cleared to 0 by the UG bit of TIM_EGR or by a slave controller operating in reset mode.</p>

21.4.12. TIM1 Auto-Reload Register (TIM1_ARR)

Address offset: 0x2C

Reset value: 0x0000 FFFF

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

Bit	Name	R/W	Reset Value	Function
31: 16	Res	-	-	Res
15: 0	ARR [15:0]	RW	0xFFFF	<p>Auto-reload values</p> <p>The ARR contains the value that will be loaded into the actual auto-reload register.</p> <p>The counter does not work when the value of Auto-Reload is empty.</p>

21.4.13. TIM1 Repeat Counter Register (TIM1_RCR)

Address offset: 0x30

Reset value: 0x0000 0000

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

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

Bit	Name	R/W	Reset Value	Function
31: 8	Res	-	-	Res
7: 0	REP [7:0]	RW	0	<p>Cycle counter value</p> <p>With preload enabled, these bits allow the user to set the update rate of the comparison register (i.e., periodically register transfer to the current register); if generating update interrupts is allowed, it will also affect the rate at which update interrupts are generated.</p> <p>Each time the down counter REP_CNT reaches 0, an update event is generated and the counter REP_CNT starts counting again from the REP value. Since REP_CNT only reloads the REP value when the cycle update event U_RC occurs, the new value written to the TIMx_RCR register only acts when the next cycle update event occurs.</p> <p>This means that in PWM mode, (REP+1) corresponds to:</p> <ul style="list-style-type: none"> - The number of PWM cycles in edge-aligned mode; - Number of PWM half-cycles in centrosymmetric mode;

21.4.14. TIM1 Capture/Compare Register 1 (TIM1_CCR1)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15: 0]															
RW															

Bit	Name	R/W	Reset Value	Function
31: 16	Res	-	-	Res
15: 0	CCR1[15: 0]	RW	0	<p>Capture/compare the value of 1</p> <p>If the CC1 channel is configured as an output:</p>

Bit	Name	R/W	Reset Value	Function
				<p>CCR1 contains the value loaded into the current Capture/Compare 1 register (preloaded value).</p> <p>If the preload function is not selected in the TIMx_CMR1 register (OC1PE bit), the written value is immediately transferred to the current register. Otherwise this preloaded value is transferred to the current Capture/Compare 1 register only when an update event occurs.</p> <p>The current capture/comparison register participates in the comparison with counter TIMx_CNT and generates an output signal on port OC1.</p> <p>If the CC1 channel is configured as an input:</p> <p>CCR1 contains the counter value transmitted by the last input capture 1 event (IC1).</p>

21.4.15. TIM1 Capture/Compare Register 2 (TIM1_CCR2)

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15: 0]															
RW															

Bit	Name	R/W	Reset Value	Function
31: 16	Res	-	-	Res
15: 0	CCR2[15: 0]	RW	0	<p>Capture/compare channel 2 values</p> <p>If the CC2 channel is configured as an output:</p> <p>CCR2 contains the value loaded into the current Capture/Compare 2 register (preloaded value).</p> <p>If the preload function is not selected in the TIMx_CMR1 register (OC2PE bit), the written value is immediately transferred to the current register. Otherwise this preloaded value is transferred to the current Capture/Compare 2 register only when an update event occurs.</p> <p>The current capture/comparison register participates in the comparison with counter TIMx_CNT and generates an output signal on port OC2.</p> <p>If the CC2 channel is configured as an input:</p>

				CCR2 contains the counter value transmitted by the last input capture 2 event (IC2).
--	--	--	--	--

21.4.16. TIM1 Capture/Compare Register 3 (TIM1_CCR3)

Address offset: 0x3C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15: 0]															
RW															

Bit	Name	R/W	Reset Value	Function
31: 16	Res	-	-	Res
15: 0	CCR3[15: 0]	RW	0	<p>Capture/compare the value of 3</p> <p>If the CC3 channel is configured as an output: CCR3 contains the value loaded into the current Capture/Compare 3 register (preloaded value).</p> <p>If the preload feature is not selected in the TIMx_CMR2 register (OC3PE bit), it is always loaded into the current register.</p> <p>Otherwise, this preloaded value is loaded into the current Capture/Compare 3 register only when an update event occurs.</p> <p>The current capture/compare register contains the value to be compared with counter TIMx_CNT and signaled on the OC port.</p> <p>If the CC3 channel is configured as an input: CCR3 contains the counter value transmitted by the last input capture 3 event (IC3).</p>

21.4.17. TIM1 Capture/Compare Register 4 (TIM1_CCR4)

Address offset: 0x40

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15: 0]															

RW

Bit	Name	R/W	Reset Value	Function
31: 16	Res	-	-	Res
15: 0	CCR4[15: 0]	RW	0	<p>Capture/compare the value of 4</p> <p>If the CC4 channel is configured as an output: CCR4 contains the value loaded into the current Capture/Compare 4 register (preloaded value).</p> <p>If the preload feature is not selected in the TIMx_CMR2 register (OC4PE bit), it is always loaded into the current register.</p> <p>Otherwise, this preloaded value is loaded into the current Capture/Compare 4 register only when an update event occurs.</p> <p>The current capture/compare register contains the value to be compared with counter TIMx_CNT and signaled on the OC port.</p> <p>If the CC4 channel is configured as an input: CCR4 contains the counter value transmitted by the last input capture 4 event (IC4).</p>

21.4.18. TIM1 Brake and Deadband Register (TIM1_BDTR)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG [7:0]							
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 16	Res	-	-	Res
15	MOE	RW	0	<p>Master Output Enable</p> <p>This bit is cleared to 0 by hardware asynchronously once the brake input is valid. Depending on the value of the AOE bit, it can be cleared by software to 0 or automatically set to 1. It is only valid for channels configured as outputs.</p> <p>0: Disable OC and OCN output or force to idle state;</p>

Bit	Name	R/W	Reset Value	Function
				1: Turn on the OC and OCN outputs if the corresponding enable bits (CcxE, CcxNE bits of the TIMx_CCER register) are set.
14	AOE	RW	0	<p>Auto Output Enable</p> <p>0: MOE can only be set to 1 by the software;</p> <p>1: The MOE can be set to 1 by the software or automatically set to 1 at the next update event (if the brake input is not valid).</p> <p>Note: Once the LOCK level (LOCK bit in the TIMx_BDTR register) is set to 1, this bit cannot be modified.</p>
13	BKP	RW	0	<p>Brake Input Polarity</p> <p>0: Brake input active low;</p> <p>1: Brake input active high.</p> <p>Notes:</p> <p>1. Once the LOCK level (LOCK bit in the TIMx_BDTR register) is set to 1, this bit cannot be modified.</p> <p>2. Any write operation to this bit requires a delay of one APB clock before it works.</p>
12	BKE	RW	0	<p>Brake function enable</p> <p>(0: Disable the brake function);</p> <p>1: Turn on the brake function.</p> <p>Notes:</p> <p>1. Once the LOCK level (LOCK bit in the TIMx_BDTR register) is set to 1, this bit cannot be modified.</p> <p>2. Any write operation to this bit requires a delay of one APB clock before it works.</p>
11	OSSR	RW	0	<p>Off state" selection in operation mode</p> <p>This bit is used when MOE=1 and the channel is a complementary output. The OSSR bit is not present in timers without complementary outputs.</p> <p>Refer to the detailed description of OC/OCN enable (Capture/Compare Enable Register (TIMx_CCER)).</p> <p>0: Disable OC/OCN output (release control of GPIO);</p> <p>1: Turns on the OC/OCN output and outputs an invalid level once CCxE=1 or CCxNE=1.</p> <p>OC/OCN enable output signal = 1.</p> <p>Note: Once the LOCK level (LOCK bit in the TIMx_BDTR register) is set to 2, this bit cannot be modified.</p>
10	OSSI	RW	0	Idle mode "off state" selection

Bit	Name	R/W	Reset Value	Function
				<p>This bit is used when MOE=0 and the channel is set to output.</p> <p>Refer to the detailed description of OC/OCN enable (Capture/Compare Enable Register (TIMx_CCER)).</p> <p>0: Disable OC/OCN output (release control of GPIO);</p> <p>1: once CCxE=1 or CCxNE=1, OC/OCN outputs its idle level.</p> <p>Note: Once the LOCK level (LOCK bit in the TIMx_BDTR register) is set to 2, this bit cannot be modified.</p>
9: 8	LOCK[1:0]	RW	00	<p>Locked Settings</p> <p>This bit provides write protection against software errors.</p> <p>00: Lock off, registers are not write-protected;</p> <p>01: Lock level 1, cannot write the DTG/BKE/BKP/AOE bits of the TIMx_BDTR register and the OISx/OISxN bits of the TIMx_CR2 register;</p> <p>10: Latch level 2, the bits in Latch level 1 cannot be written, nor can the CC polarity bits (once the channel in question has been set to output via the CCxS bit, the CCxP/CCNxP bits of the TIMx_CCER register) and the OSSR/OSSI bits;</p> <p>11: Latch Level 3, cannot write to bits in Latch Level 2, nor to the CC control bits (OCxM/OCxPE bits of the TIMx_CMRx register once the channel in question has been set to output via the CCxS bit);</p> <p>Note: The LOCK bit can only be written once after a system reset, and once written to the TIMx_BDTR register, its contents are frozen until reset.</p>
7: 0	DTG [7:0]	RW	0000 0000	<p>Deadband generator setting</p> <p>These bits define the deadband duration between insertion of complementary outputs. Assume that DT denotes its duration:</p> <p>$DTG[7:5] = 0xx \Rightarrow DT = DTG[7:0] \times Tdtg, Tdtg = TDTS;$</p> <p>$DTG[7:5] = 10x \Rightarrow DT = (64 + DTG[5:0]) \times Tdtg, Tdtg = 2 \times TDTS;$</p> <p>$DTG[7:5] = 110 \Rightarrow DT = (32 + DTG[4:0]) \times Tdtg, Tdtg = 8 \times TDTS;$</p> <p>$DTG[7:5] = 111 \Rightarrow DT = (32 + DTG[4:0]) \times Tdtg, Tdtg = 16 \times TDTS;$</p>

Bit	Name	R/W	Reset Value	Function
				<p>Example: if TDTS = 125ns (8MHZ), the possible dead time is:</p> <p>0 to 15875ns if the step time is 125ns;</p> <p>16us to 31750ns if the step time is 250ns;</p> <p>32us to 63us if the step time is 1us;</p> <p>64us to 126us if the step time is 2us;</p> <p>Note: Once the LOCK level (LOCK bit in the TIMx_BDTR register) is set to 1, 2, or 3, these bits cannot be modified.</p>

21.4.19. TIM1 DMA Control Register (TIM1_DCR)

Address offset: 0x48

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 13	Res	-	-	Res
12: 8	DBL [4:0]	RW	0 0000	<p>DMA sequential transfer length</p> <p>These bits define the transfer length of the DMA in continuous mode (when reading or writing to the address of the TIMx_DMAR register)</p> <p>(when the timer then performs one consecutive transmission), i.e.: defines the number of times it is transmitted:</p> <p>00000: 1 transmission</p> <p>00001: 2 transmissions</p> <p>00010: 3 transmissions</p> <p>.....</p> <p>.....</p> <p>10001: 18 transmissions</p>
7: 5	Res	RW	0	Reserved, always reads 0
4: 0	DBA [4:0]	RW	0 0000	<p>DBA[4:0]: DMA base address</p> <p>These bits define the base address of the DMA in continuous mode (when the address of the TIMx_DMAR register is read or written)</p>

				time), DBA is defined as the offset from the address where the TIMx_CR1 register is located: 00000: TIMx_CR1, 00001: TIMx_CR2, 00010: TIMx_SMCR.
--	--	--	--	--

21.4.20. TIM1 continuous mode DMA address (TIM1_DMAR)

Address offset: 0x4C

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 16	Res	-	-	Res
15: 0	DMAB [15:0]	RW	0	<p>DMA Sequential Transfer Register</p> <p>A read or write to the TIMx_DMAR register results in an access operation to the register at the following address: TIMx_CR1 address + (DBA + DMA pointer) x4, where:</p> <p>"TIMx_CR1 Address" is the address of control register 1;</p> <p>"DBA" is the base address defined in the TIMx_DCR register;</p> <p>The "DMA Pointer" is an offset that is automatically controlled by the DMA and depends on the DBL defined in the TIMx_DCR register.</p>

22. General purpose timer (TIM2/3)

22.1. Introduction to TIM2/TIM3

The TIM2 general-purpose timer consists of a 32-bit auto-reload counter driven by a programmable prescaler.

The TIM3 general-purpose timer consists of a 16-bit auto-reload counter driven by a programmable prescaler.

Suitable for a variety of applications, including measuring the pulse length of an incoming signal (input capture) or generating an output waveform (output comparison and PWM).

Using the timer prescaler and the RCC clock controller prescaler, the pulse length and waveform period can be adjusted from a few microseconds to a few milliseconds.

Each timer is completely independent and does not share any resources with each other. They can be synchronized together.

22.2. TIM2/3 Main Features

General purpose TIM2/3 timer functions include:

- 16-bit (TIM3), 32-bit (TIM2) up, down, up-down auto-reload counter
- 16-bit programmable prescaler, counter clock frequency division factor of any value between 1 and 65536
- 4 independent channels
 - Input Capture
 - Output Comparison
 - PWM generation (edge or center aligned mode)
 - Single pulse mode output
- Synchronization circuits can be used to synchronize the control timer with other timers connected internally using external signals.
- Interrupt/DMA is generated when the following events occur
 - Updates: Counter overflow up/down, counter initialization (via software or internal/external trigger)
 - Trigger events (counter starts, stops, initializes or counts triggered internally/externally)
 - Input Capture
 - Output Comparison
- Supports incremental (quadrature) encoder and Hall sensor circuits for positioning
- Trigger input as external clock or per-cycle current management

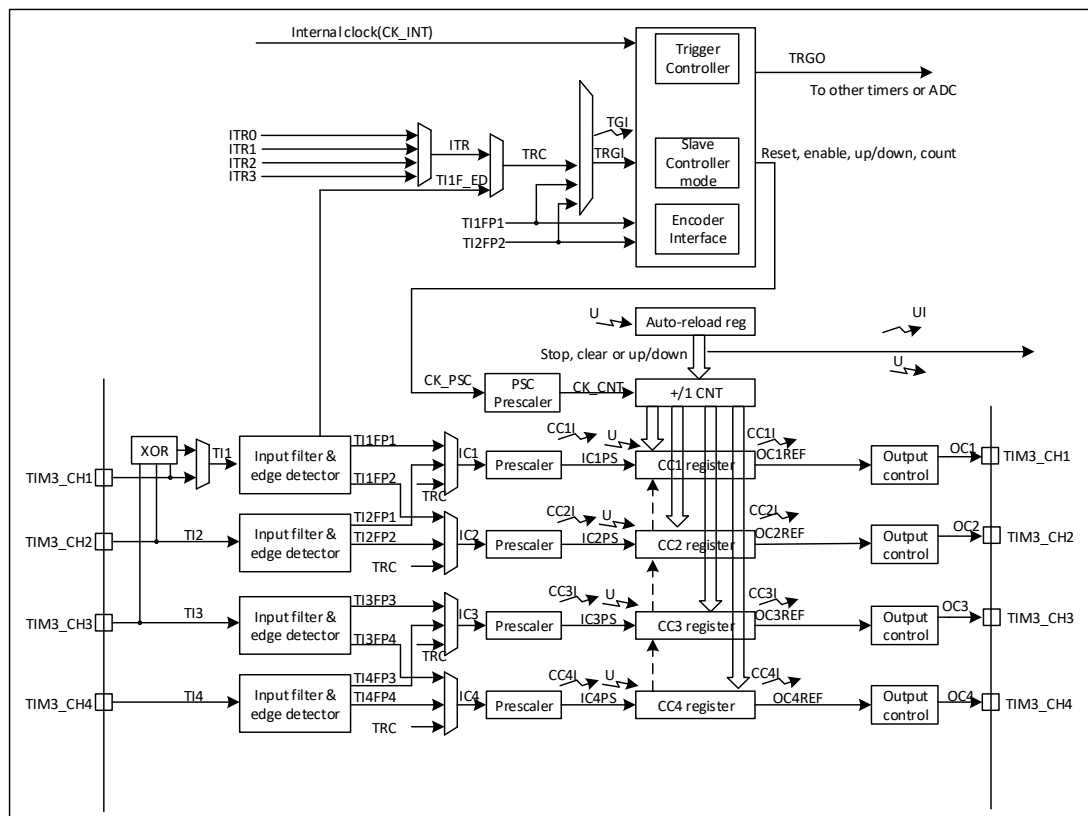


Figure 22-1 General Purpose Timer Architecture

22.3. TIM2/3 Functional Description

22.3.1. time base unit (in computing)

The main part of the TIM2/3 timer is a 16-bit (TIM3) or 32-bit (TIM2) counter and its associated auto-reload register. This counter can count up, count down or count up and down in both directions. This counter clock is divided by a prescaler.

The counter, auto-reload registers, and prescaler registers can be read and written by software, and reads and writes remain valid even if the counter is still running.

The time base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Automatic Reload Register (TIMx_ARR)

The auto-reload registers are preloaded, and writing or reading the auto-reload registers will access the preloaded registers. Depending on the setting of the auto-reload preload enable bit (ARPE) in the TIMx_CR1 register, the contents of the preload register are transferred to the shadow register either all the time or at each update event UEV. The update event is generated when the counter reaches upflow (downflow condition in case of down counter) and when the UDIS bit in the TIMx_CR1 register is equal to zero. Update events can also be generated by software.

The counter is driven by the clock output of the prescaler, CK_CNT, which is valid only when the counter enable bit (CEN) in the counter TIMx_CR1 register is set.

Note that the counter starts counting one clock cycle after the CEN bit of the TIMx_CR register is set.

Prescaler Description

The prescaler divides the counter clock by any value between 1 and 65536. It is a 16-bit counter based on 16-bit register control (in the TIMx_PSC register). Because this control register has a buffer, it can be changed at runtime. The parameters of the new prescaler are adopted on the arrival of the next update event.

The following figure gives an example of changing the counter parameters while the prescaler is running.

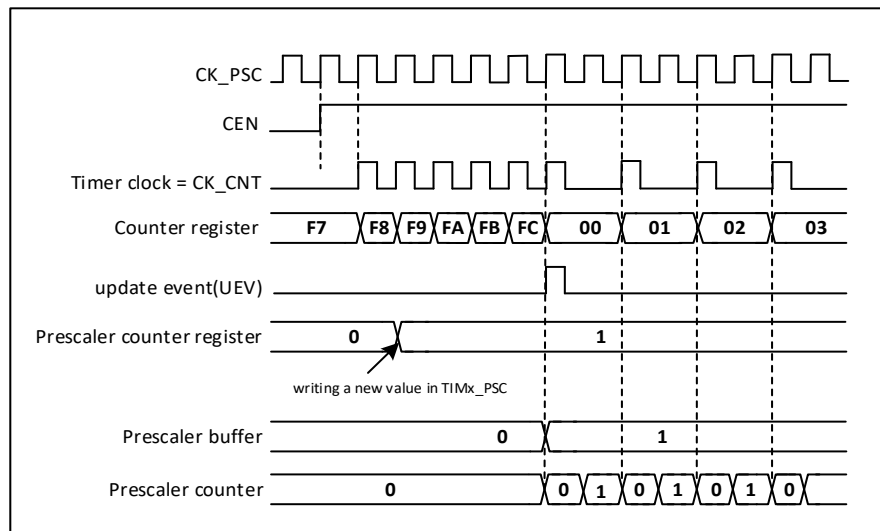


Figure 22-2 Timing diagram of the counter when the prescaler parameter is changed from 1 to 2

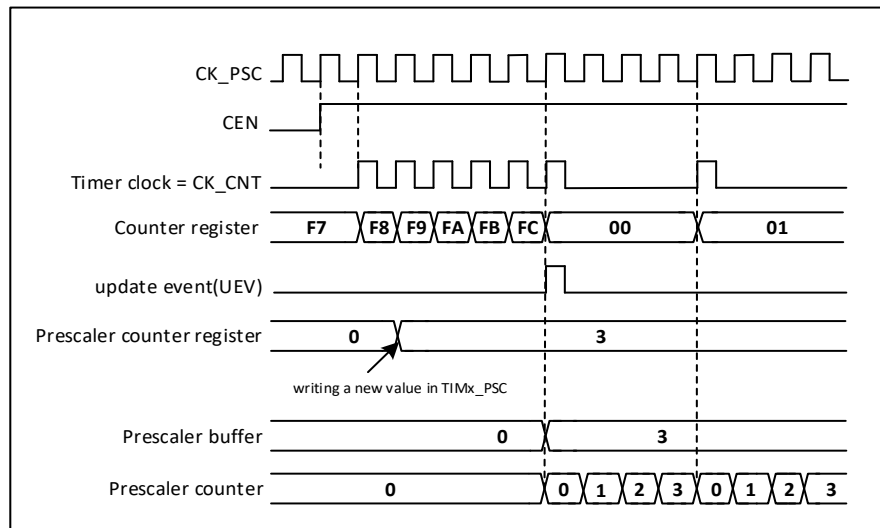


Figure 22-3 Timing diagram of the counter when the prescaler parameter is changed from 1 to 4

22.3.2. Counter Mode

22.3.2.1. Up Count Mode

Upward counting mode, which is a counter that counts from 0 to an auto-reloaded value, then re-starts counting from 0 again and generates a count overflow event.

An update event can be generated each time the count overflows, and setting the UG bit in the TIMx_EGR register (either by software or using a slave mode controller) can similarly generate an update event.

The update event can be disabled by setting the UDIS bit in the TIMx_CR1 register; this also prevents the shadow register from being updated when a new value is written to the preload register. No update event will be generated until the UDIS bit is cleared. However, when an update event should be generated, the counter is still cleared '0' and the prescaler count is also invited to 0 (but the prescaler value remains unchanged). In addition, if the URS bit in the TIMx_CR1 register is set (selecting the source of the update request), setting the UG bit generates an update event UEV, but the UIF flag is not set by the hardware (i.e., no interrupt or DMA request is generated). This is to avoid clearing the counter when an event is captured and generating both update and capture interrupts.

When an update event occurs, all of the following registers are updated and the hardware simultaneously (based on the URS bit) sets the update flag bit (UIF bit in the TIMx_SR register).

- The auto-reload shadow register is reset to the value of the preload register (TIMx_ARR).
- The prescaler buffer is set to the value of the preload register (the contents of the TIMx_PSC register).

The following figure gives some examples of how the counter acts at different clock frequencies when TIMx_ARR=0x36.

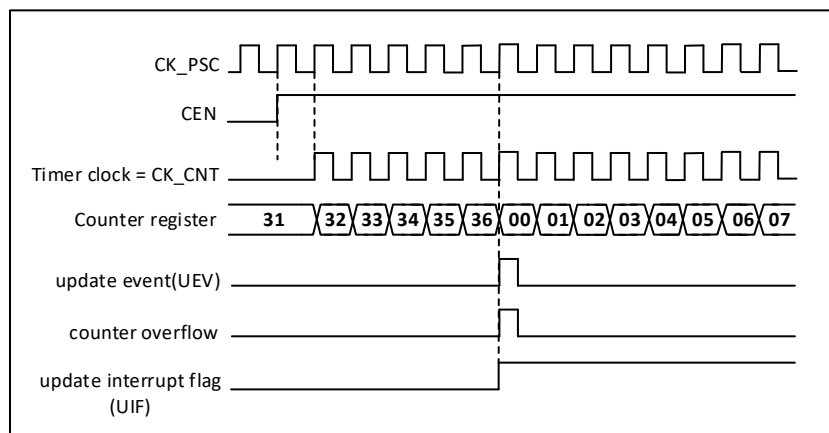


Figure 22-4 Timing diagram of the counter with internal clock division factor of 1

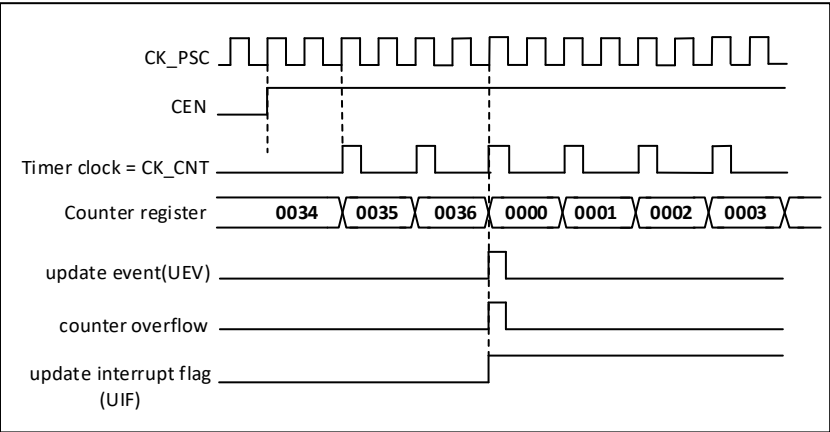


Figure 22-5 Timing diagram of the counter with internal clock division factor of 2

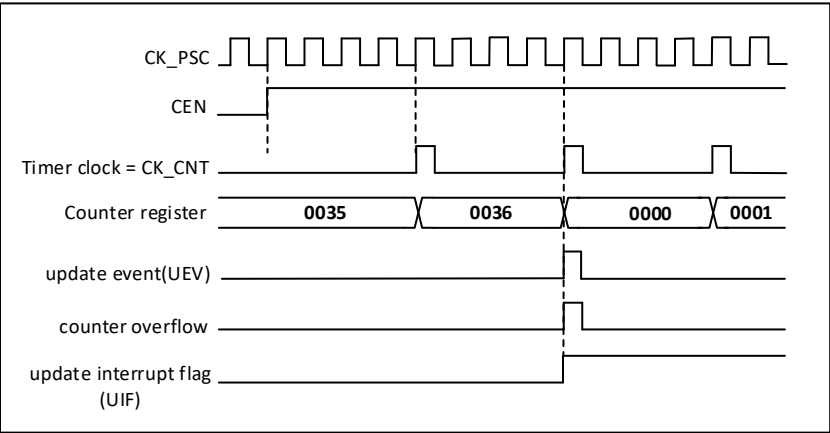


Figure 22-6 Timing diagram of the counter with internal clock divider factor of 4

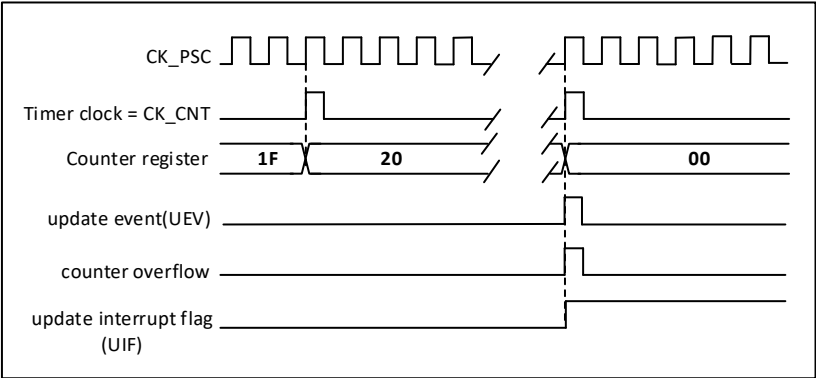


Figure 22-7 Timing diagram of the counter with internal clock dividing factor N

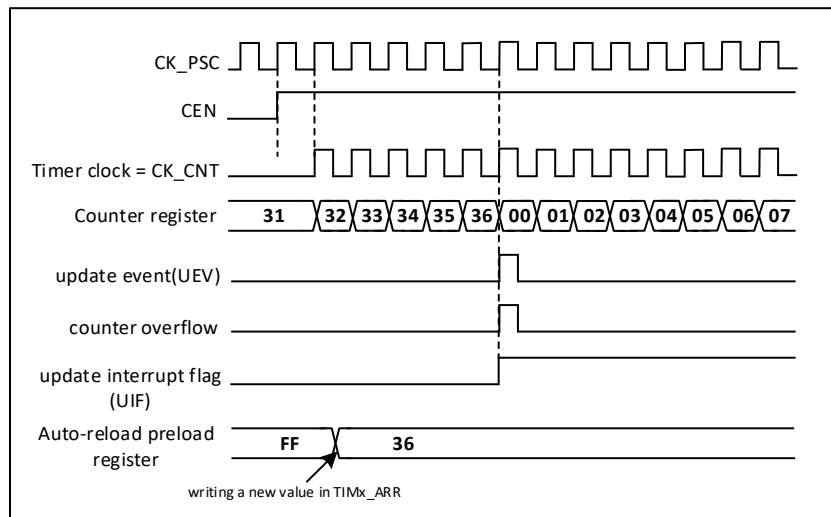


Figure 22-8 Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)

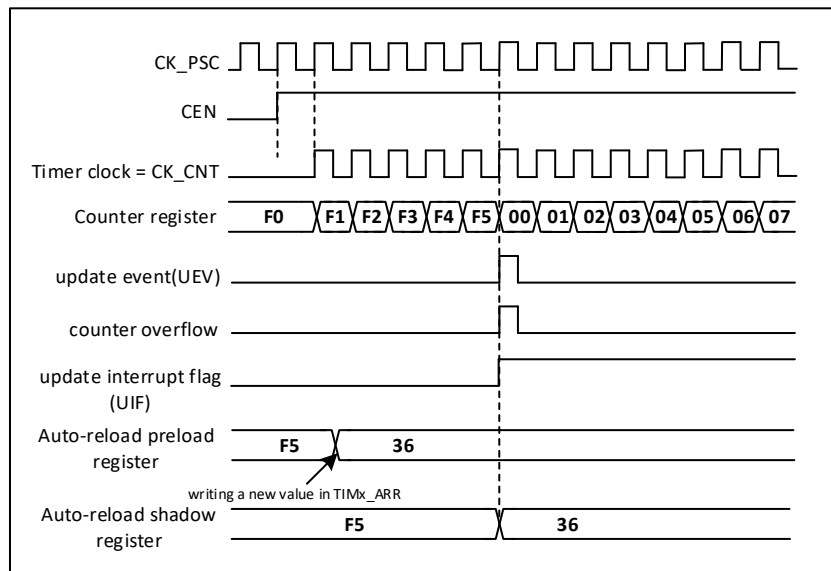


Figure 22-9 Counter timing diagram, update event when ARPE=1 (preloaded with TIMx_ARR)

22.3.2.2. Down Count Mode

Downward counting mode, which starts counting down to 0 from the auto-reloaded value, then restarts counting down from the auto-reloaded value and generates a downward overflow event.

An update event can be generated each time the count overflows, and an update event can be similarly generated by setting the UG bit in the TIMx_EGR register (either by software or using a slave mode controller).

Setting the UDIS bit of the TIMx_CR1 register disables update events. This avoids updating the registers when writing new values to the preloaded registers. Therefore no update event is generated until the UDIS bit is cleared to zero. However, the counter still restarts counting from the current auto-load value and the prescaler counter is cleared to zero (but the prescaler coefficient remains unchanged).

In addition, if the URS bit in the TIMx_CR1 register is set (selecting the source of the update request) , setting the UG bit generates an update event UEV but does not set the UIF flag (and therefore does not generate an interrupt and a DMA request), this is to avoid generating an update and a capture interrupt at the same time when a capture event occurs and clears the counter.

When an update event occurs, all of the following registers are updated and (depending on the setting of the URS bit) the update flag bit (the UIF bit in the TIMx_SR register) is also set.

- The prescaler buffer is loaded with the preloaded value (the value of the TIMx_PSC register).
- The current auto-reload register is updated to the preloaded value (the contents of the TIMx_ARR register).

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

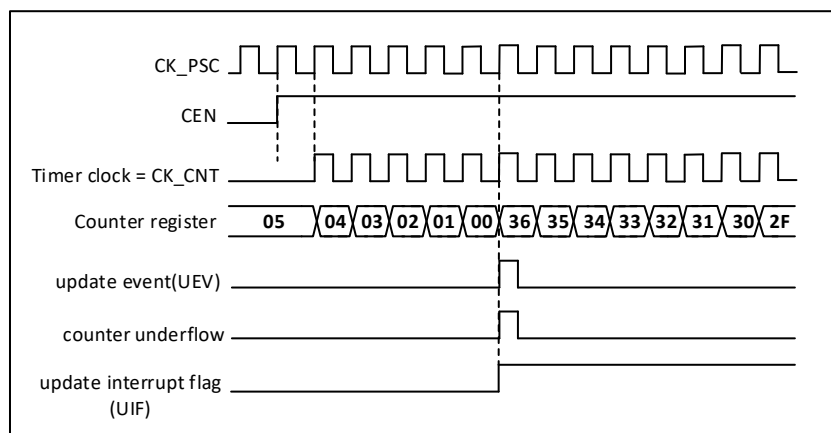


Figure 22-10 Timing diagram of the counter with internal clock division factor of 1

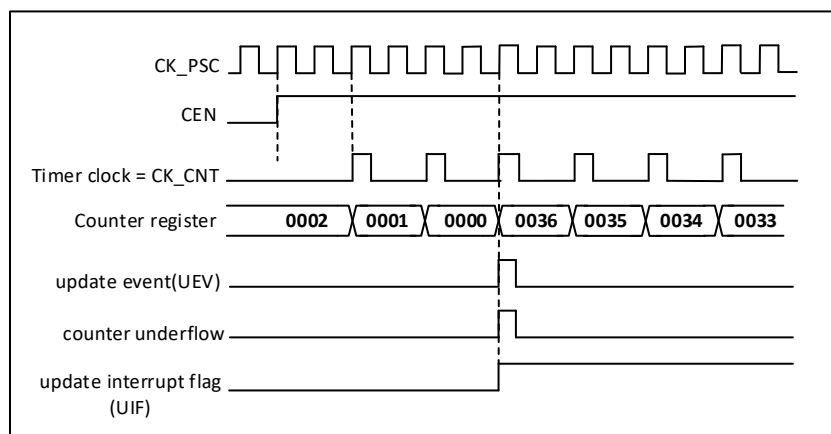


Figure 22-11 Timing diagram of the counter with internal clock divider factor of 2

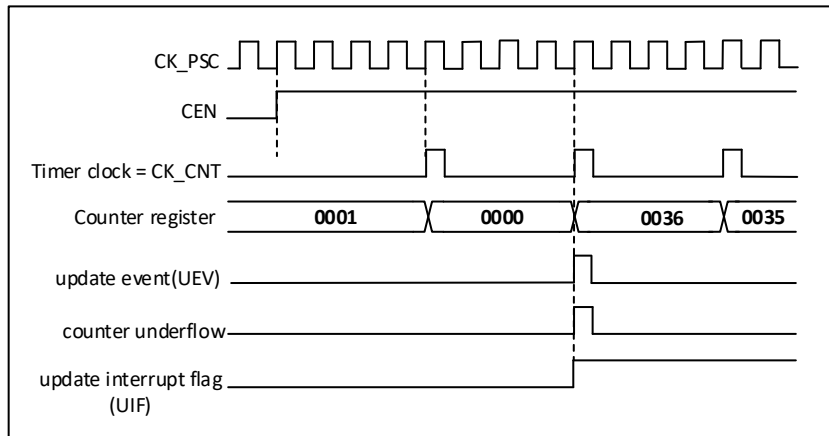


Figure 22-12 Timing diagram of the counter with internal clock divider factor of 4

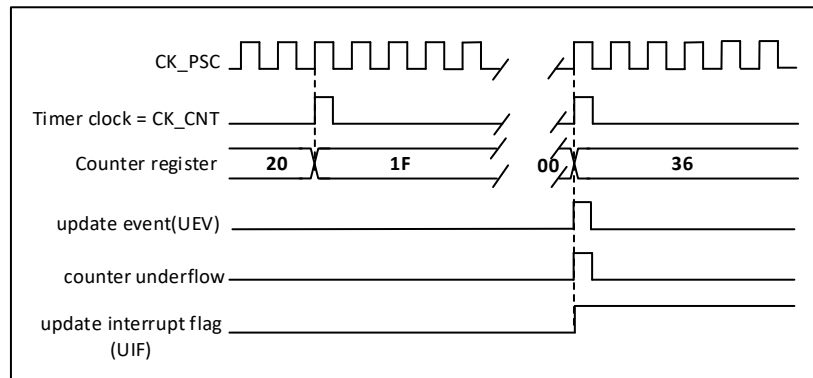


Figure 22-13 Timing diagram of the counter with internal clock dividing factor N

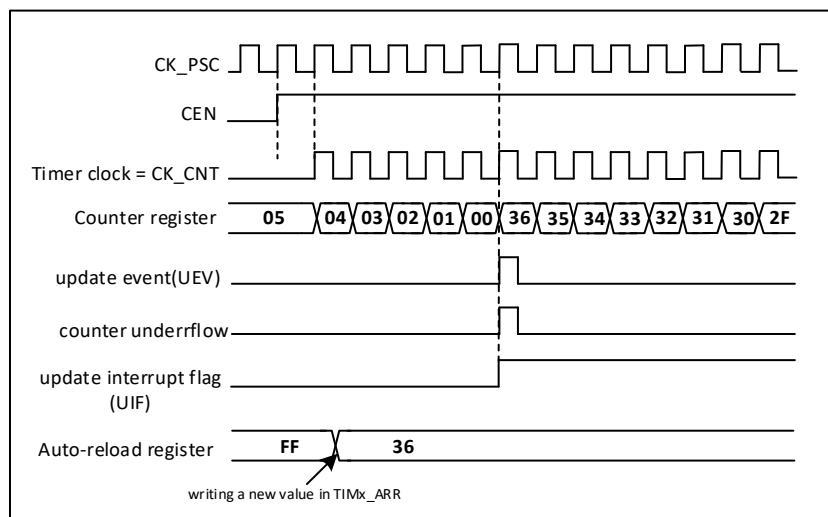


Figure 22-14 Counter Timing Diagram, Update Events When Cycle Counter is Not Used

Central alignment mode (counting up/down)

In center-aligned mode, the counter counts from 0 to the auto-loaded value (TIMx_ARR register) -1, generates a counter overflow event, then counts down to 1 and generates a counter underflow event before counting from 0 again.

Central alignment mode is valid when CMS in the TIMx_CR1 register is not equal to zero. When the channel is configured in output mode, the output compare interrupt flag will be set when: counting

down (center aligned mode 1, CMS="01"), counting up (center aligned mode 2, CMS="10") counting up counting down (Central Alignment Mode 3, CMS="11").

In this mode, the DIR direction bit in TIMx_CR1 cannot be written. It is updated by hardware and indicates the current count direction.

Update events can be generated on every count overflow and every count underflow; they can also be generated by setting (in software or using a slave mode controller) the UG bit in the TIMx_EGR register. The counter then resumes counting from 0, and the prescaler resumes counting from 0 as well.

Setting the UDIS bit in the TIMx_CR1 register disables UEV events. This avoids updating the shadow registers when writing new values to the preloaded registers. Therefore no update event is generated until the UDIS bit is cleared to zero. However, the counter will still continue to count up or down depending on the current auto-reload value.

In addition, if the URS bit in the TIMx_CR1 register is set (selecting update requests) , setting the UG bit will generate an update event UEV but not set the UIF flag (and therefore not generate interrupts and DMA requests), this is to avoid generating both update and capture interrupts when a capture event occurs and clears the counter.

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

- The prescaler buffer is loaded with the value of the preload (TIMx_PSC register).
- The current auto-reload register is updated to the preloaded value (TIMx_ARR)

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

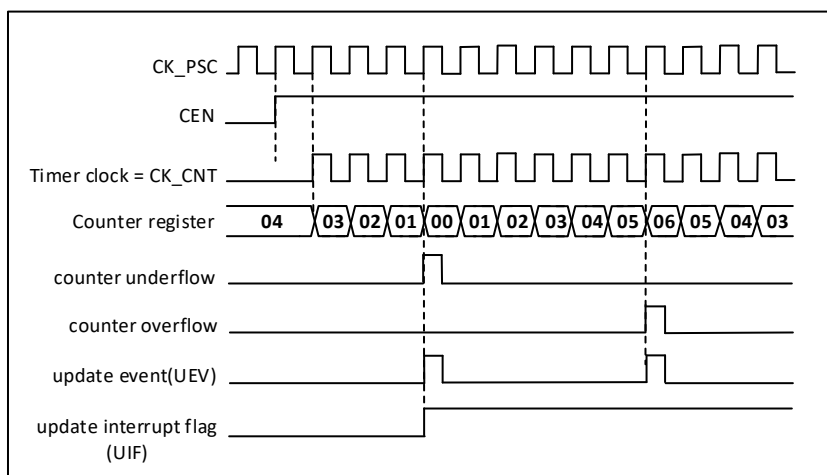


Figure 22-15 Counter timing diagram with internal clock divider factor of 1 and TIMx_ARR = 0x6

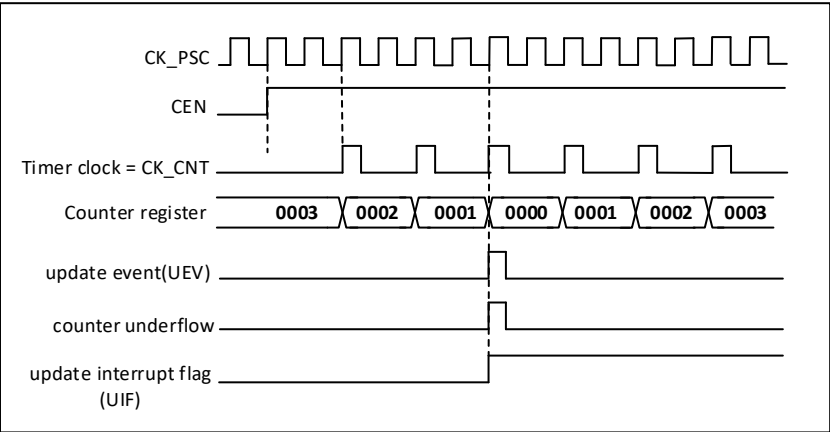


Figure 22-16 Timing diagram of the counter with internal clock divider factor of 2

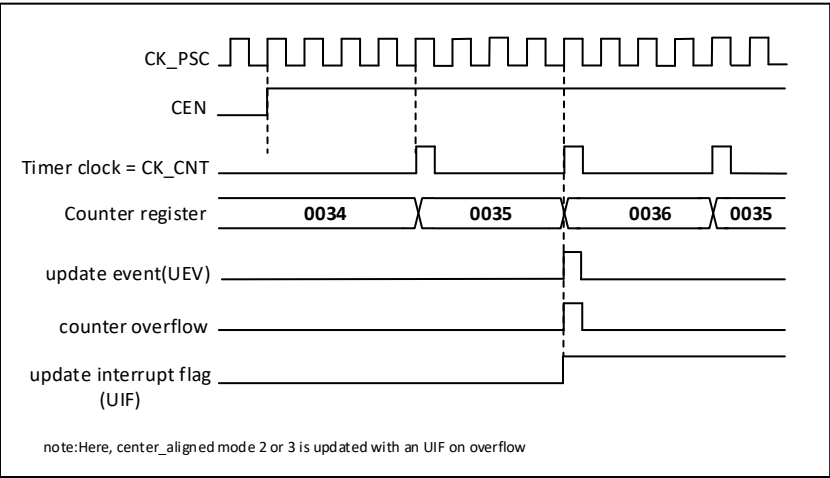


Figure 22-17 Counter timing diagram with internal clock divider factor of 4 and $TIMx_ARR=0x36$

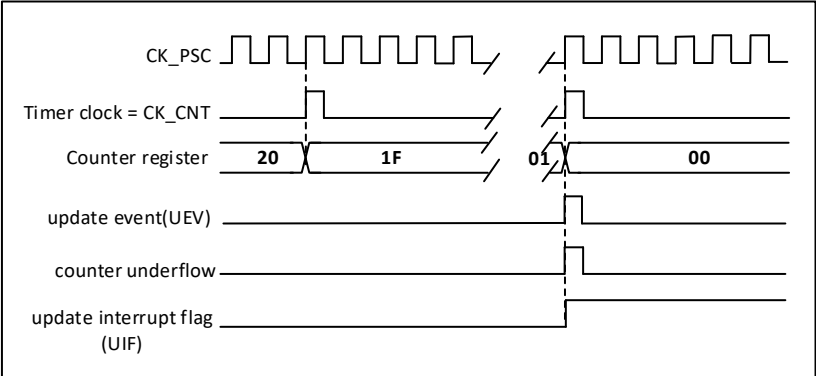


Figure 22-18 Timing diagram of the counter with internal clock dividing factor N

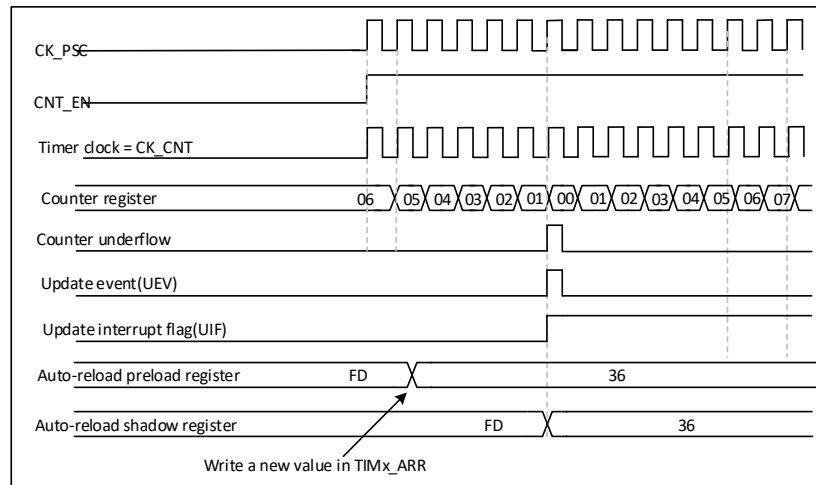


Figure 22-19 Counter timing diagram, update event at ARPE=1 (counter underflow)

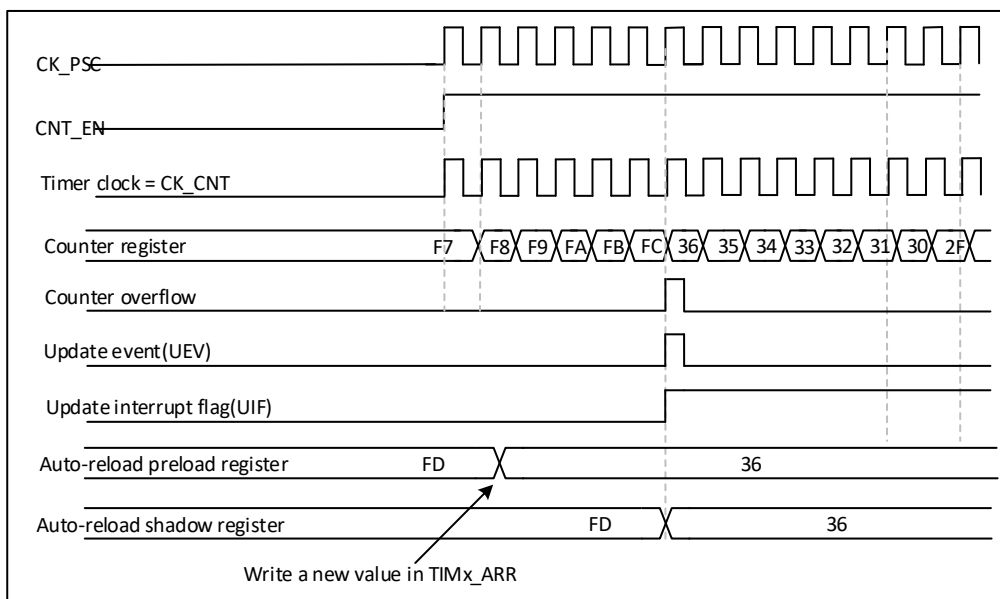


Figure 22-20 Counter timing diagram, update event at ARPE=1 (counter overflow)

22.3.3. Clock source

The counter can be clocked by the following clock sources:

- Internal clock (CK_INT)
- External Clock Mode 1: External Input Pin
- Internal Trigger Input (ITRx): uses one timer as a prescaler for another timer. For example, a timer Timer1 can be configured as a prescaler for another timer Timer3.

Internal clock source (CK_INT)

If the slave mode controller is disabled, the CEN, DIR (TIMx_CR1 register) and UG bits (TIMx_EGR register) are de facto control bits and can only be modified by software. As long as the CEN bit is written to 1, the prescaler clock is provided by the internal clock CK_INT.

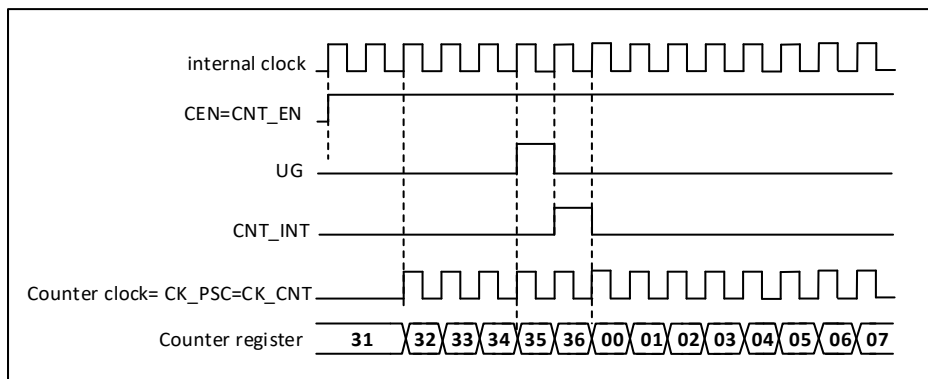


Figure 22-21 Control circuit in general mode with internal clock division factor of 1

External clock source mode 1

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

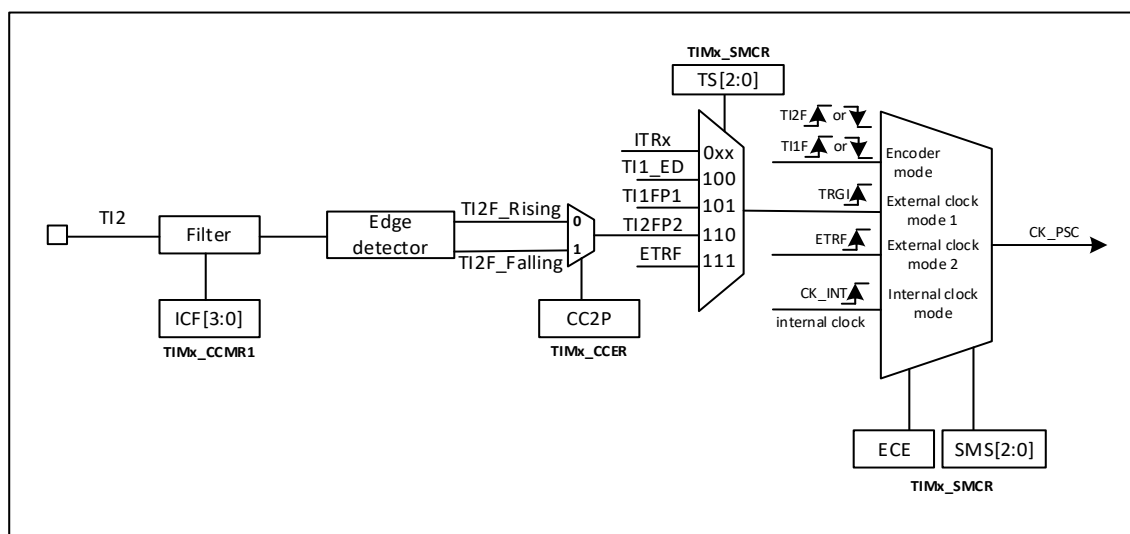


Figure 22-22 T12 Example of External Clock Connection

For example, to configure the up counter to count on the rising edge of the T12 input, use the following steps:

1. Configure TIMx_CMR1 register CC2S=01 to configure channel 2 to detect the rising edge of the T12 input
2. Configure TIMx_CMR1 register's IC2F[3:0] to select the input filter bandwidth (if no filter is required, keep IC2F= 0000)
3. Configure CC2P=0 in the TIMx_CCER register to select the rising edge polarity
4. Configure SMS=111 in the TIMx_SMCR register to select the external clock mode for the timer 15. Configure TS=110 in the TIMx_SMCR register to select T12 as the trigger input source
6. Set CEN= in the TIMx_CR1 register to 1 to start the counter

Note: The capture prescaler is not used as a trigger, so there is no need to configure it. When the rising edge occurs at T12, the counter counts once and the TIF flag is set. The delay between the rising edge of T12 and the actual clock of the counter depends on the resynchronization circuit at the T12 input.

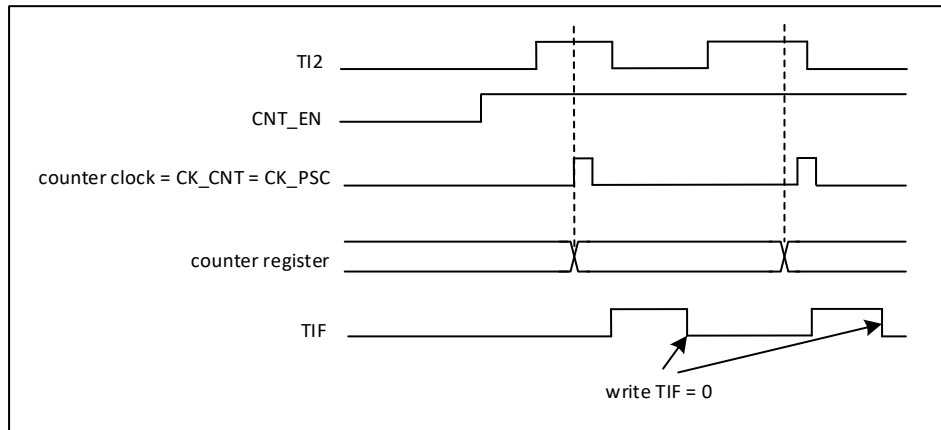


Figure 22-23 Control circuit in external clock mode 1

22.3.4. Capture/Compare Channel

Each capture/compare channel is organized around a capture/compare register (containing shadow registers), including the input portion of the capture (input filtering, multiplexing, and prescaler), and the output portion (comparator and output control).

The input section samples the corresponding TIx input signal and produces a filtered signal, TIxF. An edge monitor with polarity selection then generates a signal (TIxFPx) that can be used as an input trigger from the pattern controller or as a capture control. This signal enters the capture register (ICxPS) through pre-divided frequency.

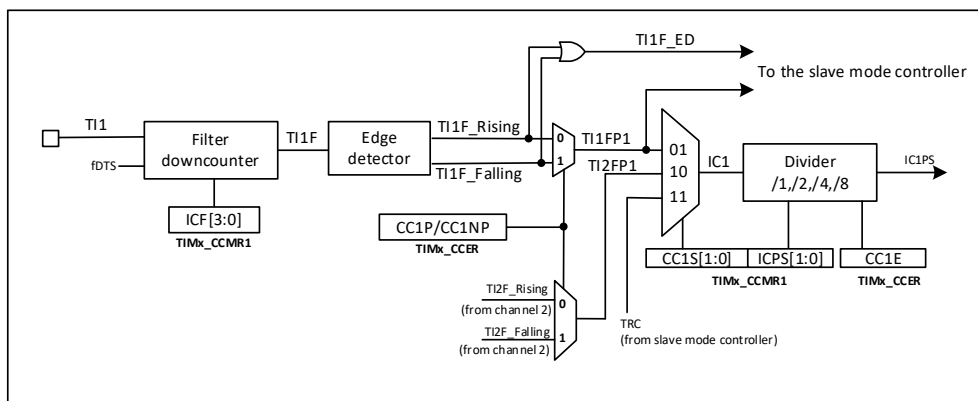


Figure 22-24 Capture/compare channels (e.g., channel 1 input section)

The output section generates an intermediate waveform OCxREF (highly active) as a reference, and the end of the chain determines the polarity of the final output signal.

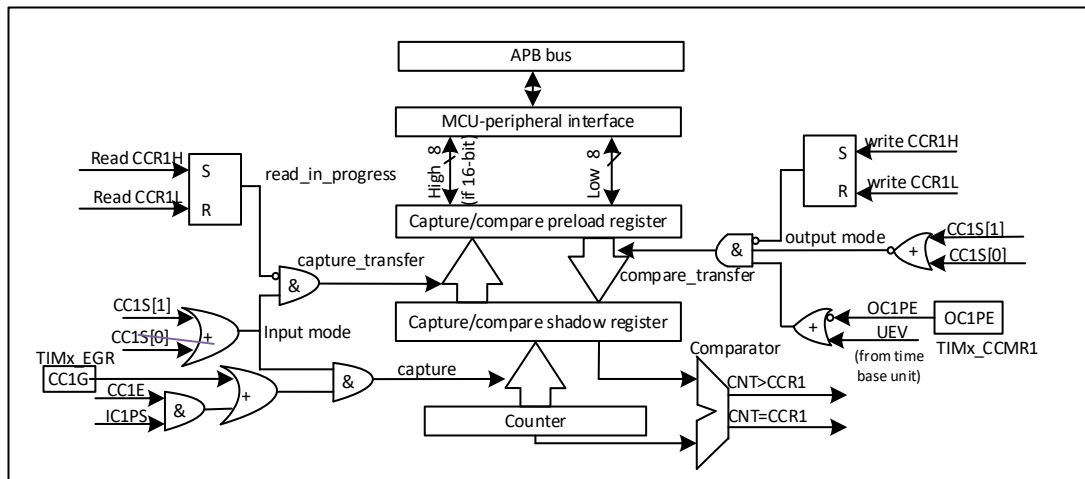


Figure 22-25 Capture/Compare Channel 1 Main Circuitry

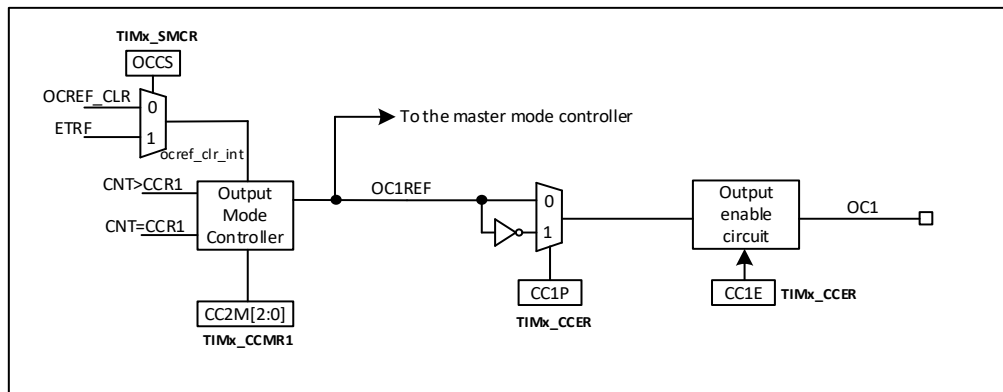


Figure 22-26 Output section of capture/compare channel (channel 1)

The capture/compare module consists of a preloaded register and a shadow register. The read/write process operates only on the preloaded registers.

In capture mode, the capture occurs on the shadow registers, which are then copied to the preloaded registers.

In comparison mode, the contents of the preloaded registers are copied to the shadow registers, and then the contents of the shadow registers are compared to the counter.

22.3.5. Input Capture Mode

In input capture mode, the current value of the counter is latched into the capture/compare register when the corresponding edge on the ICx signal is detected. When a capture event occurs, the corresponding CCxIF flag (TIMx_SR register) is set to 1, and an interrupt or DMA request will be generated if interrupt and DMA operations are turned on. If the CCxIF flag is already high when the capture event occurs, the repeat capture flag CCxOF (TIMx_SR register) is set to 1. Writing CCxIF=0 clears CCxIF, or reading the capture data stored in the TIMx_CCRx register also clears CCxIF. Writing CCxOF=0 clears CCxOF.

The following example shows how to capture the value of the counter into the TIMx_CCR1 register on the rising edge of the TI1 input as follows:

- Select valid inputs: TIMx_CCR1 must be connected to the TI1 input, so write CC1S=01 to the TIMx_CCR1 register, as long as CC1S is not '00', the channel is configured as an input and the TIMx_CCR1 register becomes read-only.
- Configure the input filter for the desired bandwidth based on the characteristics of the input signal (i.e., when the input is TIx, the input filter control bit is the ICxF bit in the TIMx_CMRx register). Assuming that the input signal is dithered for a maximum of 5 internal clock cycles, we have to configure the filter with a bandwidth longer than 5 clock cycles; we can therefore (at the fDTS frequency) sample 8 consecutive times in order to confirm a true edge shift on TI1, i.e., by writing IC1F=0011 in the TIMx_CMR1 register.
- Select the active conversion edge of the TI1 channel by writing CC1P=0 (rising edge) (and CC1NP=0) in the TIMx_CCER register
- Configure the input prescaler. In this example, we want the capture to occur at every valid level-transition moment, so the prescaler is disabled (write IC1PS=00 to the TIMx_CMR1 register).
- Setting CC1E=1 in the TIMx_CCER register allows the value of the capture counter to be captured into the capture register.
- If required, allow related interrupt requests by setting the CC1IE bit in the TIMx_DIER register and DMA requests by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:

- When a valid level transition is generated, the counter value is transferred to the TIMx_CCR1 register.
- The CC1IF flag is set (interrupt flag). When at least 2 consecutive captures occur without CC1IF having been cleared, CC1OF is also set to 1.
- If the CC1IE bit is set, an interrupt is generated.
- If the CC1DE bit is set, a DMA request is also generated.

In order to handle capture overflows, it is recommended that data be read before the capture overflow flag is read; this is to avoid losing capture overflow information that may be generated after the capture overflow flag is read and before the data is read.

Note: Setting the corresponding CCxG bit in the TIMx_EGR register allows you to generate input capture interrupts and/or DMA requests through software.

22.3.6. PWM Input Mode

This mode is a special case of the Input Capture mode and operates the same as the Input Capture mode except for the following differences:

- The two ICx signals are mapped to the same TIx input.
- These 2 ICx signals are edge valid, but of opposite polarity.
- One of the TIxFP signals is used as the trigger input signal, while the slave mode controller is configured in reset mode.

For example, when it is necessary to measure the length (TIMx_CCR1 register) and duty cycle (TIMx_CCR2 register) of a PWM signal input to TI1, proceed as follows (depending on the frequency of CK_INT and the value of the prescaler)

- To select the valid input for TIMx_CCR1: Set CC1S=01 of the TIMx_CMR1 register (TI1 selected).
- Select the active polarity of TI1FP1 (used to capture data into TIMx_CCR1 and clear the counter): set CC1P=0 (active on rising edge).
- To select the valid input for TIMx_CCR2: Set CC2S=10 in the TIMx_CMR1 register (TI1 selected).
- Select the active polarity of TI1FP2 (capture data to TIMx_CCR2): set CC2P=1 (active on falling edge).
- To select a valid trigger input signal: set TS=101 in the TIMx_SMCR register (select TI1FP1).
- Configure the slave mode controller for reset mode: set SMS=100 in TIMx_SMCR.
- Enable Capture: Set CC1E=1 and CC2E=1 in TIMx_CCER register.

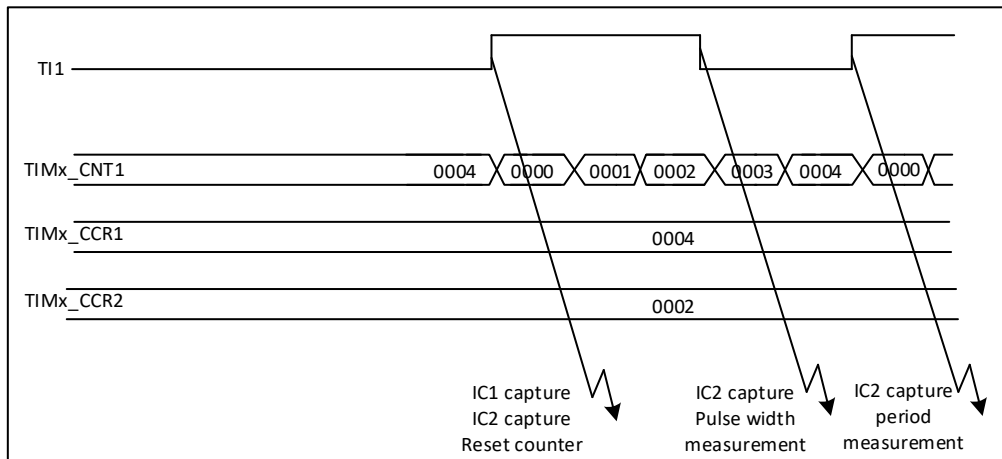


Figure 22-27 PWM Input Mode Timing

22.3.7. Forced Output Mode

In output mode (CCxS=00 in the TIMx_CMRx register), the output compare signals (OCxREF and the corresponding OCx) can be directly forced to a valid or invalid state by software, independent of the result of the comparison between the output compare register and the counter. Setting the corresponding OCxM=101 in the TIMx_CMRx register will force the output compare signal (OCxREF/OCx) to be active. In this way OCxREF is forced high (OCxREF is always active high), while OCx gets a signal with the opposite polarity of CCxP.

For example, if CCxP=0 (OCx active high), then OCx is forced high. Setting OCxM=100 in the TIMx_CMRx register forces the OCxREF signal low.

In this mode, the comparison between the TIMx_CCRx shadow register and the counter is still in progress and the corresponding flags are modified. Therefore corresponding interrupts and DMA requests are still generated. This will be covered in the Output Comparison Mode section below.

22.3.8. Output Comparison Mode

This function is used to control an output waveform or to indicate that a given period of time has elapsed. When the contents of the counter and the capture/compare register are the same, the output compare function does the following:

- Outputs the values defined by the output comparison mode (OCxM bit in the TIMx_CMRx register) and output polarity (CCxP bit in the TIMx_CCER register) to the corresponding pins. When comparing

matches, the output pin can hold its level (OCxM=000), be set to an active level (OCxM=001), be set to an inactive level (OCxM=010), or be flipped (OCxM=011).

- Set the flag bit in the interrupt status register (CxIF bit in the TIMx_SR register).
- An interrupt is generated if the corresponding interrupt mask (CCxIE bit in the TIMx_DIER register) is set.
- If the corresponding enable bits are set (CCxDE bit in the TIMx_DIER register and CCDS bit in the TIMx_CR2 register selects the DMA request function), a DMA request is generated.

The OCxPE bit in TIMx_CMRx selects whether or not the TIMx_CCRx register requires the use of a pre-loaded register.

In output comparison mode, the update event UEV has no effect on the OCxREF and OCx outputs. The synchronization can be accurate up to one counting cycle of the counter. Output compare mode (in single pulse mode) can also be used to output a single pulse.

Outputs the configuration steps for the compare mode:

1. Select the counter clock (internal, external, prescaler).
2. Write the corresponding data to the TIMx_ARR and TIMx_CCRx registers.
3. If an interrupt request is to be generated, set the CCxIE bit.
4. Select the output mode, for example:
 - Require the counter to flip the output pin of OCx when it matches CCRx, set OCxM=011.
 - Set OCxPE = 0 to disable preloaded registers.
 - Set CCxP = 0 to select polarity as active high.
 - Set CCxE = 1 to enable the output.

5. Set the CEN bit of the TIMx_CR1 register to start the counter

The TIMx_CCRx registers can be updated by software at any time to control the output waveform, provided that the preloaded registers are not used (OCxPE='0', otherwise the shadow registers of TIMx_CCRx can only be updated when the next update event occurs). An example is given in the figure below.

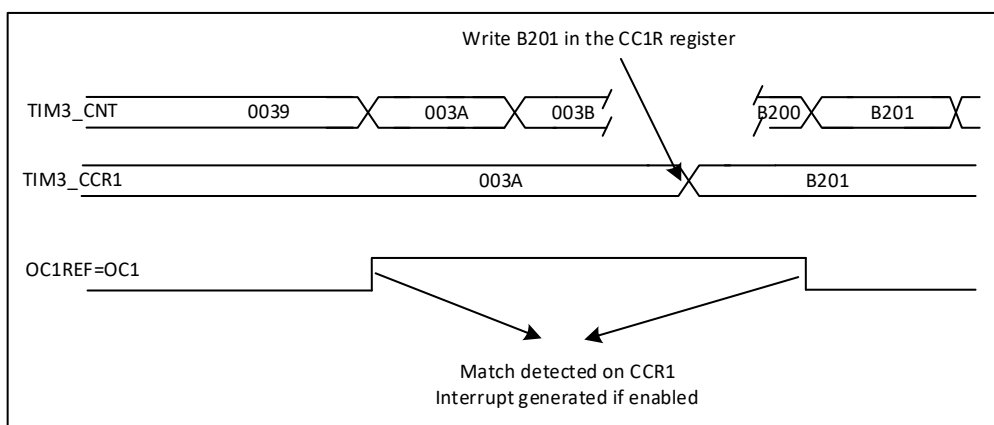


Figure 22-28 Output Compare Mode, Flip OC1

22.3.9. PWM mode

The pulse width modulation mode can allow the generation of a signal with a frequency determined by the TIMx_ARR register and a duty cycle determined by the TIMx_CCRx register.

Writing "110" (PWM mode 1) or "111" (PWM mode 2) to the OCxM bit in the TIMx_CMRx register can independently set each OCx output channel to generate one way PWM. The corresponding preload registers must be enabled by setting the OCxPE bit of the TIMx_CMRx register, and finally by setting the ARPE bit of the TIMx_CR1 register, which (in up-counting or center-symmetric modes) enables the auto-reloading preload registers.

Preloaded registers are transferred to the shadow registers only when an update event occurs, so all registers must be initialized by setting the UG bit in the TIMx_EGR register before the counter starts counting.

The polarity of the OCx can be set by software in the CCxP bit in the TIMx_CCER register, which can be set to active high or active low. The output enable of the OCx is controlled by a combination of the CcxE, CcxNE, MOE, OSSI, and OSSR bits (in the TIMx_CCER and TIMx_BDTR registers). See the description of the TIMx_CCER register for details.

In PWM mode (Mode 1 or Mode 2), TIMx_CNT and TIMx_CCRx are always being compared, (based on the counter count direction) to determine if $\text{TIMx_CCRx} \leq \text{TIMx_CNT}$ or $\text{TIMx_CNT} \leq \text{TIMx_CCRx}$ is met.

Depending on the state of the CMS bit in the TIMx_CR1 register, the timer is able to generate either an edge-aligned PWM signal or a center-aligned PWM signal.

PWM Edge Alignment Mode

■ Up Count Configuration

Performs an upward count when the DIR bit in the TIMx_CR1 register is low. See below for an example of PWM mode 1. The PWM reference signal OCxREF is high when $\text{TIMx_CNT} < \text{TIMx_CCRx}$ and low otherwise. If the comparison value in TIMx_CCRx is greater than the auto-reload value (TIMx_ARR), OCxREF is held to '1'. If the comparison value is 0, OCxREF remains '0'. The following figure shows an example of an edge-aligned PWM waveform with TIMx_ARR=8.

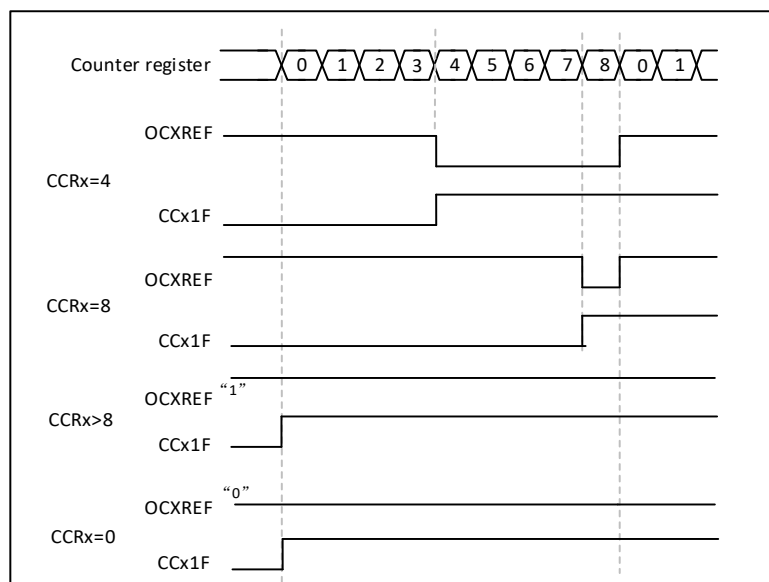


Figure 22-29 Edge-aligned PWM waveforms (ARR=8)

Down Count Configuration

Performs a down count when the DIR bit of the TIMx_CR1 register is high.

In PWM mode 1, the reference signal OCxREF is low when $TIMx_CNT > TIMx_CCRx$, otherwise it is high. If the comparison value in $TIMx_CCRx$ is greater than the auto-reload value in $TIMx_ARR$, OCxREF is held to '1'. A 0% PWM waveform cannot be generated in this mode.

PWM Central Alignment Mode

Central alignment mode when the CMS bit in the $TIMx_CR1$ register is not '00' (all other configurations have the same effect on the OCxREF/OCx signals). Depending on the CMS bit setting, the compare flag may be set to 1 when the counter is counting up, set to 1 when the counter is counting down, or set to 1 when the counter is counting up and down. The count direction bit (DIR) in the $TIMx_CR1$ register is updated by hardware; do not modify it with software.

The following figure gives some examples of centrally aligned PWM waveforms

- $TIMx_ARR = 8$
- PWM mode 1
- CMS=01 in the $TIMx_CR1$ register sets the compare flag when the counter counts down in central alignment mode

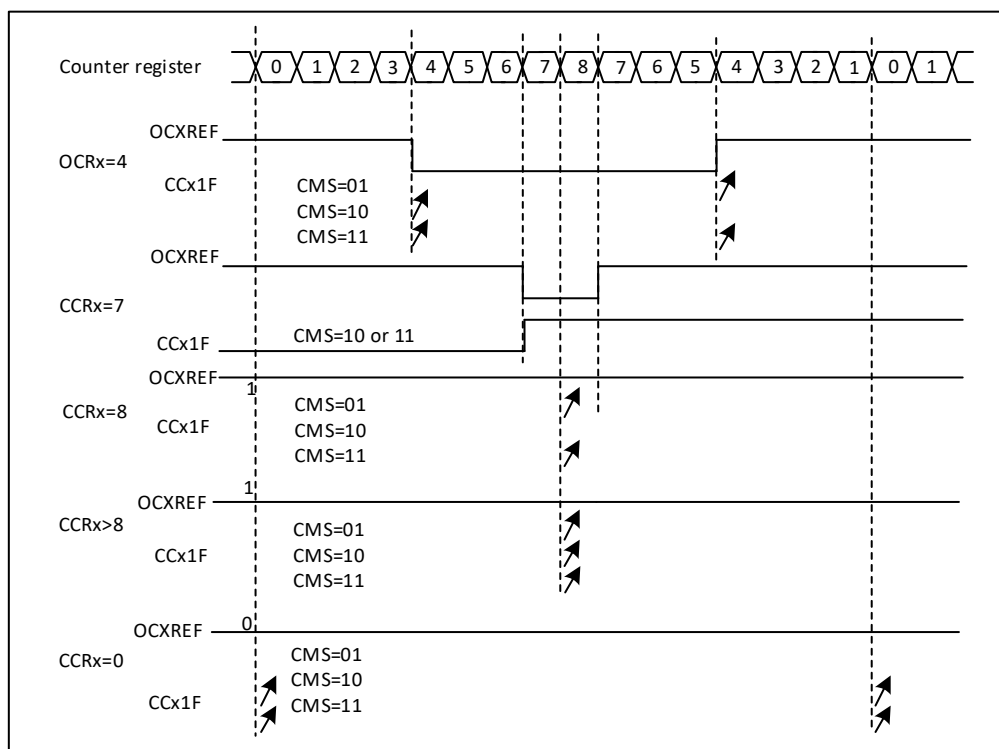


Figure 22-30 Centrally aligned PWM waveform (APR=8)

Tips for using the center alignment mode:

- The current count up/down configuration is used when entering central alignment mode; this means that whether the counter counts up or down depends on the current value of the DIR bit in the $TIMx_CR1$ register. In addition, the software cannot modify the DIR and CMS bits at the same time.
- It is not recommended to rewrite the counters when running in center-aligned mode, as this can produce unpredictable results. Especially:
 - If the write counter value is greater than the auto-reload value ($TIMx_CNT > TIMx_ARR$), the direction is not updated. For example, if the counter is counting up, it will continue to count up.

- If a value of 0 or TIMx_ARR is written to the counter, the direction is updated but no update event UEV is generated.
- The safest way to use the central alignment mode is to generate a software update (set the UG bit in the TIMx_EGR bit) before starting the counter and not to modify the counter value while the count is in progress.

22.3.10. Single Pulse Mode

The single pulse mode (OPM) is a special case of one of the many modes described previously. This mode allows the counter to respond to an excitation and, after a programmable delay, generate a pulse with a pulse width that can be controlled by the program.

The counter can be activated from the mode controller to generate waveforms in output comparison mode or PWM mode. Setting the OPM bit of the TIMx_CR1 register will select single-pulse mode, which allows the counter to automatically stop when the next update event, UEV, is generated.

A pulse is generated only when the comparison value is different from the initial value of the counter. Before starting (when the timer is waiting to be triggered), it must be configured as follows:

- Upward counting mode: counter $CNT < CCRx \leq ARR$ (in particular, $0 < CCRx$)
- Downward counting mode: Counter $CNT > CCRx$

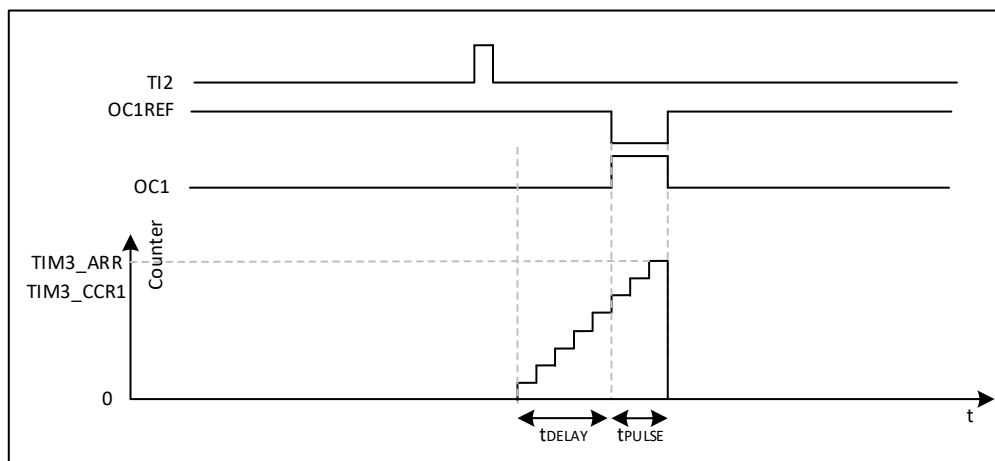


Figure 22-31 Example of single pulse mode

For example, when it is necessary to generate a positive pulse of length t_{PULSE} on OC1 after delaying t_{DELAY} at the beginning of a rising edge detected from the TI2 input pin.

Use TI2FP2 as trigger 1:

- Set CC2S=01 in the TIMx_CMR1 register to map TI2FP2 to TI2.
- Set CC2P=0 in the TIMx_CCER register to enable the TI2FP2 to detect the rising edge.
- Setting TS=110 in the TIMx_SMCR register triggers the TI2FP2 as a slave mode controller (TRGI).
- Setting SMS=110 (trigger mode) in the TIMx_SMCR register, the TI2FP2 is used to start the counter.

The OPM waveform is determined by the value written to the comparison register (taking into account the clock frequency and counter prescaler)

- t_{DELAY} is defined by the value in the TIMx_CCR1 register.
- t_{PULSE} is defined by the difference between the auto-reload value and the comparison value (TIMx_ARR - TIMx_CCR1).

- Assuming that a waveform from 0 to 1 is to be generated when a comparison match occurs, and a waveform from 1 to 0 is to be generated when the counter reaches the preloaded value; firstly, set OC1M=111 in TIMx_CMR1 register to enter PWM mode 2; selectively enable the preload registers as needed: set OC1PE=1 in TIMx_CMR1 and ARPE in TIMx_CR1 register; then fill the comparison value in TIMx_CCR1 register, and set the UG bit to generate an update event. then fill in the comparison value in the TIMx_CCR1 register and the auto-reload value in the TIMx_ARR register, set the UG bit to generate an update event, and wait for an external trigger event on TI2. In this example, CC1P = 0.

In this example, the DIR and CMS bits in the TIMx_CR1 register should be set low.

Since only one pulse is needed, OPM=1 in the TIMx_CR1 register must be set to stop counting at the next update event (when the counter flips from the auto-reload value to 0).

Special case: OCx fast enable:

In single pulse mode, the CEN bit is set by the edge detection logic at the TIx input pin to start the counter. The comparison operation between the counter and the comparison value then produces the conversion of the output. However, these operations require a certain number of clock cycles, so it limits the minimum delay tDELAY that can be obtained.

If you want to output the waveform with minimum delay, you can set the OCxFE bit in the TIMx_CMRx register; at this point, OCxREF (and OCx) responds directly to the excitation and no longer relies on the result of the comparison, and the output waveform is the same as the waveform when the comparison is matched. OCxFE only works when the channel is configured for PWM1 and PWM2 modes.

22.3.11. Encoder Interface Mode

The encoder interface mode is selected by setting SMS=001 in the TIMx_SMCR register if the counter counts only on the TI2 edge, SMS=010 if it counts only on the TI1 edge, or SMS=011 if the counter counts on both the TI1 and TI2 edges.

TI1 and TI2 polarity can be selected by setting the CC1P and CC2P bits in the TIMx_CCER register; the input filter can also be programmed if desired.

Two inputs TI1 and TI2 are used as interfaces to the incremental encoder. Referring to Table 22-1, assuming that the counter has been started (CEN=1 in the TIMx_CR1 register), the counter is driven by each valid hop on TI1FP1 or TI2FP2. TI1FP1 and TI2FP2 are the signals of TI1 and TI2 after passing them through the input filter and polarity control; without filtering and phasing, TI1FP1 = TI1 and TI2FP2 = TI2. Based on the hopping sequence of the two input signals, a counting pulse and a direction signal are generated. Depending on the hopping order of the two input signals, the counter counts up or down while the hardware sets the DIR bit of the TIMx_CR1 register accordingly. Whether the counter relies on TI1 for counting, on TI2 for counting, or on both TI1 and TI2 for counting, a trip on either input (TI1 or TI2) recalculates the DIR bit.

The encoder interface mode is basically equivalent to using an external clock with direction selection. This means that the counter only counts continuously between 0 and the auto-reload value of the TIMx_ARR register (either 0 to ARR count or ARR to 0 count, depending on the direction). So TIMx_ARR must be configured before counting starts; again, the captures, comparators, prescalers, trigger output characteristics, etc. still work as usual. Encoder mode and external clock mode 2 are not compatible and therefore cannot be operated simultaneously. In this mode, the counter is automatically modified according to the speed and direction

of the incremental encoder, so that the contents of the counter always indicate the position of the encoder. The counting direction corresponds to the direction of rotation of the connected sensor. The following table lists all possible combinations, assuming that TI1 and TI2 do not transform simultaneously.

Table 22-1 Relationship between count direction and encoder signal

active edge	Relative signal level (TI2 for TI1FP1 and TI1 for TI2FP2)	TI1FP1 signal		TI2FP2 signal	
		go up	go down	go up	go down
TI1 count only	your (honorific)	down-count	Upward Counting	disregard	disregard
	lower (one's head)	Upward Counting	down-count	disregard	disregard
TI2 count only	your (honorific)	disregard	disregard	Upward Counting	down-count
	lower (one's head)	disregard	disregard	down-count	Upward Counting
Counting on TI1 and TI2	your (honorific)	down-count	Upward Counting	Upward Counting	down-count
	lower (one's head)	Upward Counting	down-count	down-count	Upward Counting

An external incremental encoder can be connected directly to the MCU without external interface logic. However, a comparator is typically used to convert the differential output of the encoder to a digital signal, which greatly increases the resistance to noise interference. The third signal from the encoder output indicates the mechanical zero point and can be connected to an external interrupt input and trigger a counter reset.

The following figure is an example of counter operation showing the generation of counting signals and direction control. It also shows how input jitter is suppressed when a double edge is selected; jitter may be generated when the sensor's position is close to a transition point. In this example, we assume the following configuration:

- CC1S='01' (TIMx_CMR1 register, TI1FP1 mapped to IC1)
- CC2S='01' (TIMx_CMR2 register, TI2FP2 mapped to IC2)
- CC1P='0' (TIMx_CCER register, TI1FP1 not inverted, TI1FP1=TI1)
- CC2P='0' (TIMx_CCER register, TI2FP2 not inverted, TI2FP2=TI2)
- SMS='011' (TIMx_SMCR register, all inputs valid on rising and falling edges)
- CEN='1' (TIMx_CR1 register, counter enable)

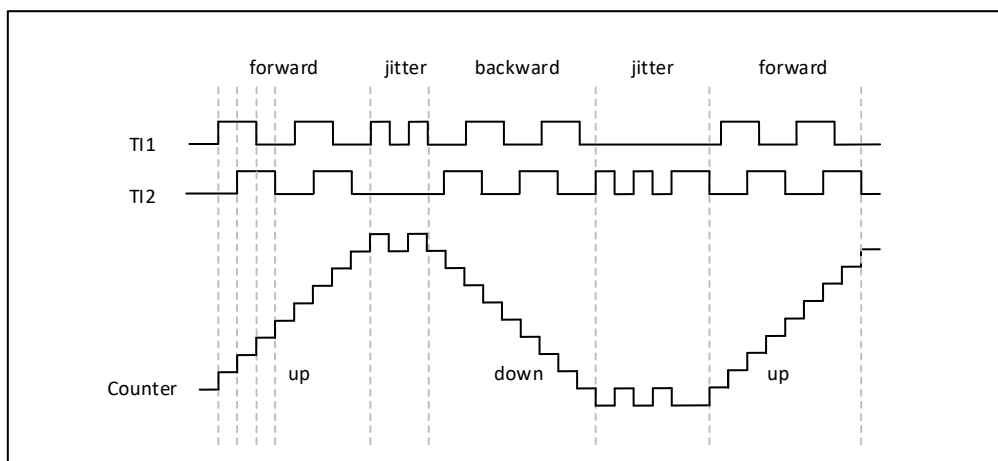


Figure 22-32 Example of counter operation in encoder mode

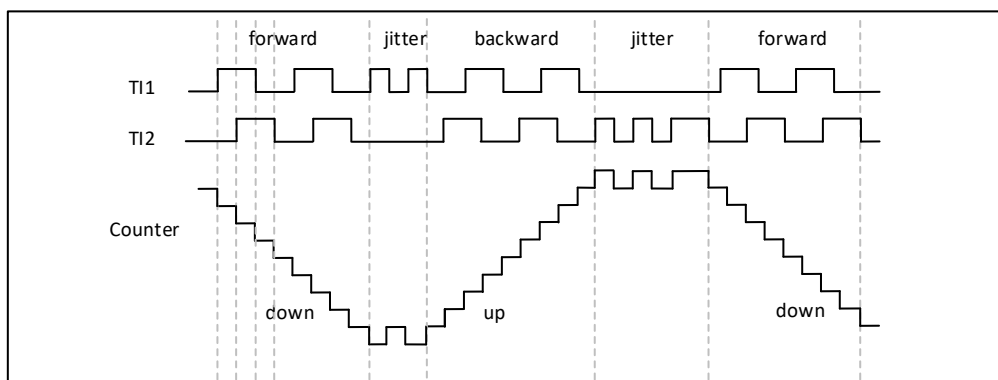


Figure 22-33 Example of encoder interface mode for IC1FP1 inversion

Provides information on the current position of the sensor when the timer is configured in encoder interface mode. Using a second timer configured in capture mode, it is possible to measure the interval between the two encoder events and obtain information about the dynamics (velocity, acceleration, deceleration). The encoder output indicating the mechanical zero point can be used for this purpose. Depending on the interval between two events, the counter can be read out at a fixed time. If possible, the counter's value can be latched to a third input capture register (the capture signal must be periodic and can be generated by another timer); its value can also be read by a DMA request generated by the real-time clock.

22.3.12. Timer input heterodyne function

The TI1S bit of the TIMx_CR2 register allows the input filter of channel 1 to be connected to the output of an iso-gate whose three inputs are TIMx_CH1, TIMx_CH2, and TIMx_CH3.

The heterosync outputs can be used for all timer input functions such as triggering or input capture.

An application example is the Hall sensor interface.

22.3.13. Synchronization of timers and external triggers

The TIMx timer can be synchronized with an external trigger in a variety of modes: reset mode, gated mode, and triggered mode.

Slave mode: Reset mode

On the occurrence of a trigger input event, the counter and its prescaler are able to be reinitialized; at the same time, an update event UEV is also generated if the URS bit of the TIMx_CR1 register is low; all pre-loaded registers (TIMx_ARR, TIMx_CCRx) are then updated.

In the following example, the rising edge of the TI1 input causes the up counter to be cleared:

- Configure channel 1 to detect the rising edge of TI1. Configure the bandwidth of the input filter (in this example, no filter is required, so keep IC1F=0000). The capture prescaler is not used in the trigger operation, so no configuration is required. The CC1S bit selects only the input capture source, i.e., CC1S=01 in the TIMx_CMR1 register. Set CC1P=0 (and CC1NP=0) in the TIMx_CCER register to determine the polarity (only the rising edge is detected).
- Set SMS=100 in TIMx_SMCR register to configure the timer in reset mode; set TS=101 in TIMx_SMCR register to select TI1 as the input source.
- Set CEN=1 in the TIMx_CR1 register to start the counter.

The counter starts counting according to the internal clock and then operates normally until a rising edge occurs in TI1; at this point, the counter is cleared to zero and then restarts counting from zero. At the same time, the trigger flag (TIF bit in the TIMx_SR register) is set, generating either an interrupt request or a DMA request depending on the setting of the TIE (interrupt enable) bit and the TDE (DMA enable) bit in the TIMx_DIER register.

The following figure shows the action when the auto-reload register TIMx_ARR = 0x36. The delay between the rising edge of TI1 and the actual reset of the counter depends on the resynchronization circuit at the TI1 input.

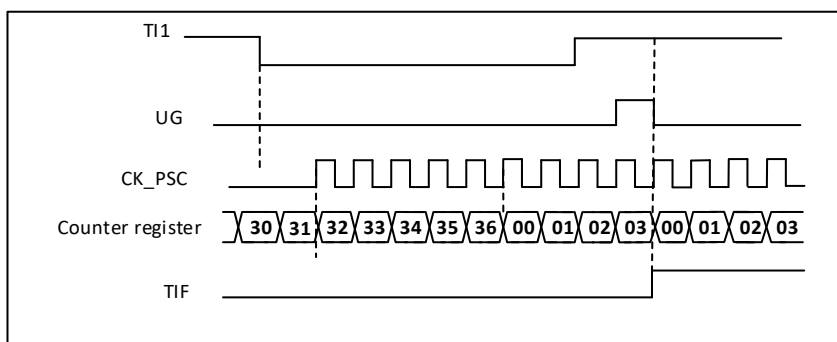


Figure 22-34 Control circuit in reset mode

Slave mode: Gated mode

Enables the counter according to the level of the selected input.

In the following example, the counter only counts up when TI1 is low:

- Configure channel 1 to detect a low level on TI1. Configure the input filter bandwidth (in this case, no filtering is required, so keep IC1F=0000). The capture prescaler is not used in the trigger operation, so no configuration is required. The CC1S bit is used to select the input capture source by setting CC1S=01 in the TIMx_CMR1 register. Set CC1P=1 (and CC1NP=0) in the TIMx_CCER register to determine the polarity (detect low only).
- Set SMS=101 in the TIMx_SMCR register to configure the timer for gating mode; set TS=101 in the TIMx_SMCR register to select TI1 as the input source.

- Set CEN=1 in the TIMx_CR1 register to start the counter. In gated mode, if CEN=0, the counter cannot start, regardless of the trigger input level.

As long as TI1 is low, the counter starts counting based on the internal clock and stops counting once TI1 goes high. The TIF scaler in TIMx_SR is set when either the counter starts or stops.

The delay between the rising edge of TI1 and the actual stopping of the counter depends on the resynchronization circuit at the TI1 input.

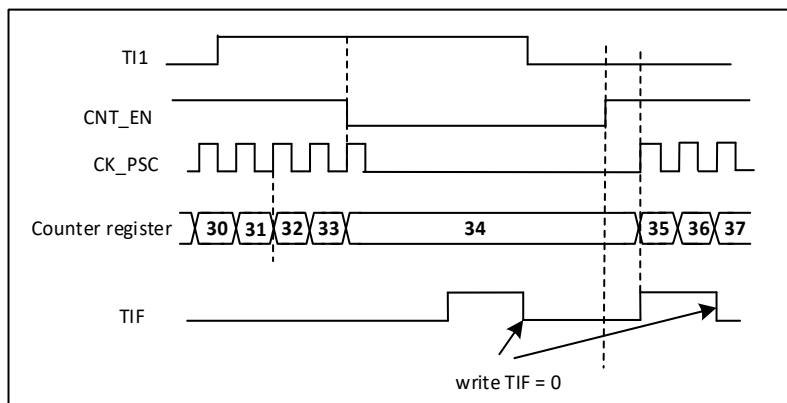


Figure 22-35 Control circuit in gated mode

Slave mode: Trigger mode

The selected event on the input enables the counter.

In the following example, the counter starts counting up on the rising edge of the TI2 input:

- Configure channel 2 to detect the rising edge of TI2. Configure the input filter bandwidth (in this example, no filter is required, keep IC2F=0000). The capture prescaler is not used in the trigger operation and does not require configuration. The CC2S bit is only used to select the input capture source by setting CC2S=01 in the TIMx_CMR1 register. Set CC2P=1 (and CC2NP=0) in the TIMx_CCER register to determine polarity (detects only low levels).
- Set SMS=110 in TIMx_SMCR register to configure the timer in trigger mode; set TS=110 in TIMx_SMCR register to select TI2 as the input source.

When a rising edge of TI2 occurs, the counter starts counting driven by the internal clock and the TIF flag is set. The delay between the rising edge of TI2 and the counter starting to count depends on the resynchronization circuit at the TI2 input.

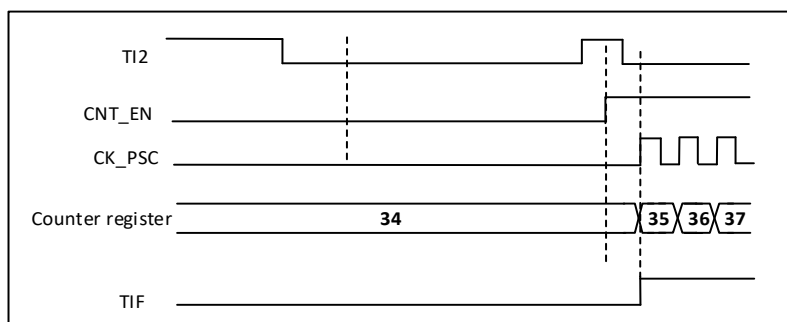


Figure 22-36 Control circuit in trigger mode

Slave Mode: External Clock Mode 2 + Trigger Mode

External clock mode 2 can be used with another slave mode (except external clock mode 1 and encoder mode). In this case, the ETR signal is used as the input for the external clock, and another input can be selected as the trigger input in reset mode, gated mode, or trigger mode. It is not recommended to use the TS bit of the TIMx_SMCR register to select ETR as the TRGI.

In the following example, once a rising edge occurs on TI1, the counter counts up on each rising edge of ETR:

- Configure the external trigger input circuit through the TIMx_SMCR register:
 - ETF=0000: no filtering
 - ETPS=00: no prescaler used
 - ETP=0: detect the rising edge of ETR, set ECE=1 to enable external clock mode 2.
- Configure channel 1 as follows to detect the rising edge of TI1:
 - IC1F=0000: No filtering.
 - The capture prescaler is not used in the trigger operation and does not need to be configured.
 - Set CC1S=01 in the TIMx_CMCR1 register to select the input capture source.
 - Set CC1P=0 in the TIMx_CCER register to determine polarity (detects only rising edges)
- Set SMS=110 in the TIMx_SMCR register to configure the timer for trigger mode. Set TS=101 in the TIMx_SMCR register to select TI1 as the input source.

When a rising edge occurs on TI1, the TIF flag is set and the counter starts counting on the rising edge of ETR. The delay between the rising edge of the ETR signal and the actual reset of the counter depends on the resynchronization circuit at the ETRP input.

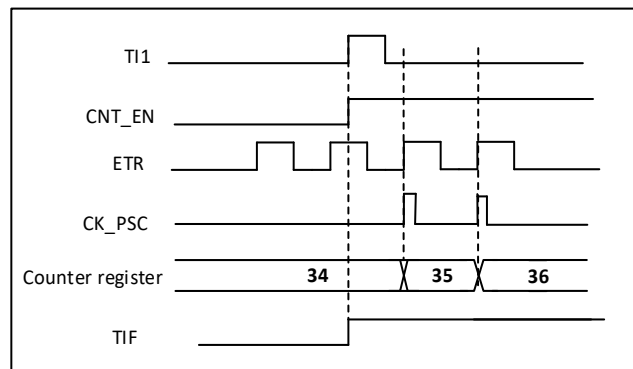


Figure 22-37 Control circuit in external clock mode 2 + trigger mode

22.3.14. Timer synchronization

All timers are connected internally for timer synchronization or linking. When a timer is in master mode, it can reset, start, stop, or clock the counter of another timer that is in slave mode.

The following figure shows an overview of the Trigger Select and Master Mode Select modules.

Using one timer as a prescaler for the other

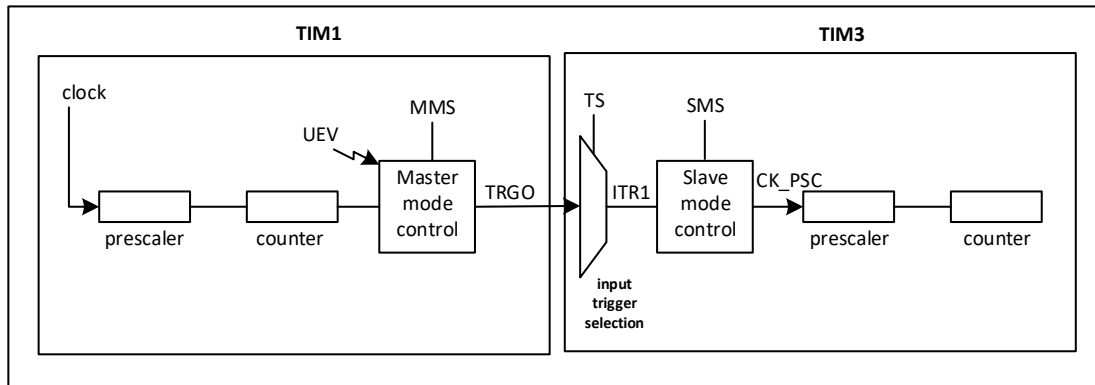


Figure 22-38 Example of a master/slave timer

For example, Timer 1 can be configured as a prescaler for Timer 3. Perform the following operations:

- Configure Timer 1 as the master mode, which can output a periodic trigger signal at each update event UEV. With MMS='010' in the TIM1_CR2 register, a rising edge signal is output on TRGO whenever an update event is generated.
- Connect the TRGO output of Timer1 to Timer3, set TS='000' in the TIM3_SMCR register, and configure Timer3 for slave mode using ITR1 as the internal trigger.
- The slave mode controller is then placed in external clock mode 1 (SMS=111 in the TIM3_SMCR register); this allows Timer 3 to be driven by the periodic rising edge (i.e., Timer 1's counter overflow) signal from Timer 1.
- Finally, the CEN bit of the corresponding (TIM3_CR1 register) must be set to start each of the two timers, ensuring that Timer3 is started first, followed by Timer1.

Note: If OCx has been selected as the trigger output of Timer 1 (MMS=1xx), its rising edge is used to drive the counter of Timer 3.

Using a timer to enable another timer

In this example, the enable of Timer 3 is controlled by the output comparison of Timer 1. Refer to the diagram above for connections. Timer 3 counts the divided internal clock only when OC1REF of Timer 1 is high. The clock frequency of both timers is obtained by dividing CK_INT by 3 ($f_{CK_CNT} = f_{CK_INT}/3$) by the prescaler.

- Configure Timer 1 to be the master mode, sending its output comparison reference signal (OC1REF) as the trigger output (MMS=100 in the TIM1_CR2 register)
- Configure the OC1REF waveform for Timer 1 (TIM1_CMR1 register)
- Configure Timer 3 to get an input trigger from Timer 1 (TS=000 in the TIM3_SMCR register)
- Configure Timer 3 for gated mode (SMS=101 in the TIM3_SMCR register)
- Set CEN=1 in TIM3_CR1 register to enable Timer 3
- Set CEN=1 in the TIM1_CR1 register to start Timer 1

Note: The clock of Timer 3 is not synchronized with the clock of Timer 1. This mode only affects the enable signal of the Timer 3 counter.

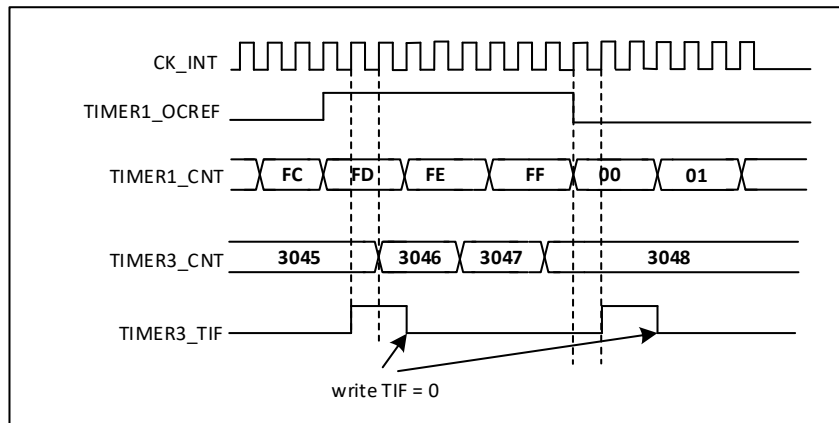


Figure 22-39 OC1REF of Timer 1 Controls Timer 3

In the example above, their counters and prescalers are not initialized before Timer 3 is started, so they start counting from their current values. It is possible to reset the 2 timers before starting timer 1 so that they start from a given value, i.e., write any value needed in the timer counter. The timer is reset by writing the UG bit of the TIMx_EGR register.

In the next example, Timer 1 and Timer 2 need to be synchronized. Timer 1 is in master mode and starts at 0, Timer 2 is in slave mode and starts at 0xE7; the prescaler coefficients are the same for both timers. Writing '0' to the CEN bit of TIM1_CR1 will disable Timer 1 and Timer 2 will then stop.

- Configure Timer 1 as the master mode and send out the timer enable signal (CNT_EN) as the trigger output (for TIM1_CR2 register MMS=001).
- Configure the OC1REF waveform for Timer 1 (TIM1_CMR1 register).
- Configure Timer 2 to get an input trigger from Timer 1 (TS=000 in the TIM2_SMCR register)
- Configure Timer 2 for gated mode (SMS=101 in the TIM2_SMCR register)
- Set UG='1' in the TIM1_EGR register to reset Timer 1.
- Set UG='1' in the TIM2_EGR register to reset Timer 2.
- Write '0xE7' to Timer 2's counter (TIM2_CNT) to initialize it to 0xE7.
- Set CEN='1' of the TIM2_CR1 register to enable Timer 2.
- Set CEN='1' in the TIM1_CR1 register to start Timer 1.
- Set CEN='0' in the TIM1_CR1 register to stop Timer 1.

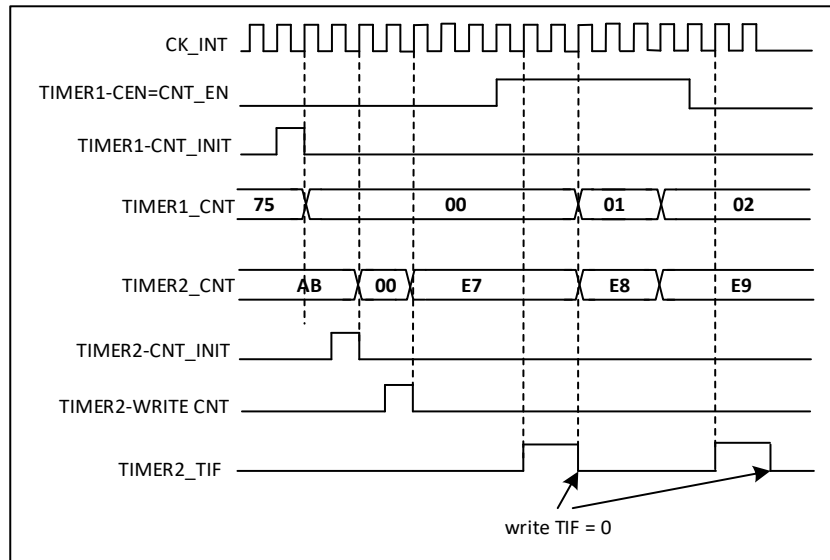


Figure 22-40 Timer 2 can be controlled by enabling Timer 1

Using a timer to start another timer

In this example, Timer 3 is enabled using an update event from Timer 1. Refer to Figure 22-38 for connections. As soon as Timer 1 generates an update event, Timer 3 starts counting from its current value (which can be non-zero) according to the divided internal clock. Upon receipt of the trigger signal, the CEN bit of Timer 3 is automatically set to '1' while the counter starts counting until a '0' is written to the CEN bit of the TIM3_CR1 register. Both timers are clocked by dividing the prescaler pair CK_INT by 3 ($f_{CK_CNT} = f_{CK_INT}/3$).

- Configure Timer 1 to be the master mode and send its update event (UEV) as a trigger output (MMS=010 in the TIM1_CR2 register)
- Configure Timer 1 period (TIM1_ARR register)
- Configure Timer 3 to get an input trigger from Timer 1 (TS=000 in the TIM3_SMCR register)
- Configure Timer 3 for trigger mode (SMS=110 for TIM3_SMCR register)
- Set CEN=1 in the TIM1_CR1 register to start Timer 1

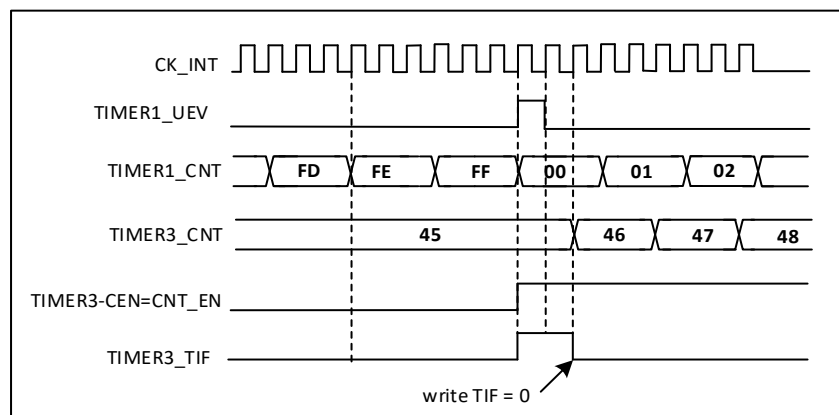


Figure 22-41 Trigger Timer 3 using Timer 1 update

In the previous example, two counters can be initialized before starting the count. The following figure shows the action using trigger mode instead of gated mode (SMS=110 in the TIM3_SMCR register) in the same configuration as above.

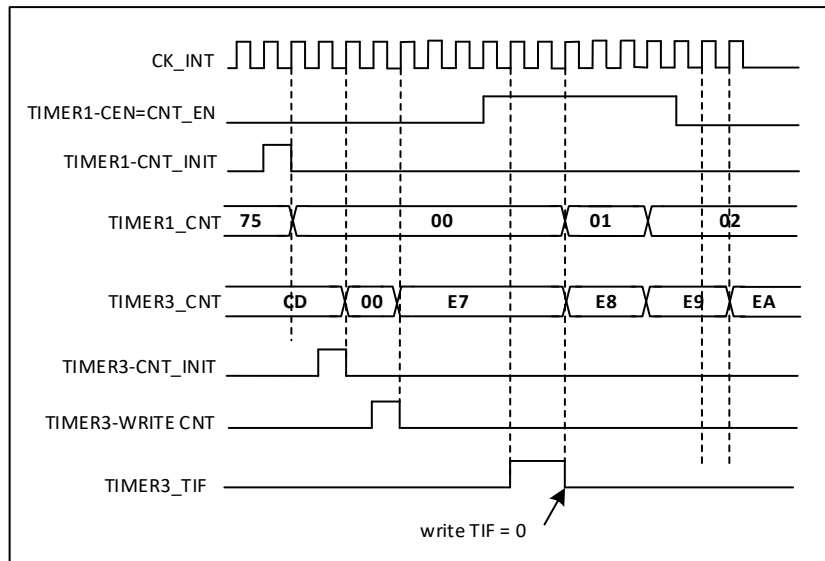


Figure 22-42 Trigger Timer 3 with Timer 1 Enable

Synchronized start of 2 timers using an external trigger

This example enables Timer 1 when the TI1 input to Timer 1 rises, and enables Timer 1 while enabling Timer 3, see Figure 22-38 for the connection. To ensure counter alignment, Timer 1 must be configured in master/slave mode (corresponding to TI1 as slave and Timer 3 as master):

- Configure Timer 1 as the master mode, send its enable as a trigger output (MMS='001' in the TIM1_CR2 register).
- Configure Timer 1 to be in slave mode and get input trigger from TI1 (TS='100' in TIM1_SMCR register).
- Configure Timer 1 for trigger mode (SMS='110' for TIM1_SMCR register).
- Configure Timer 1 for master/slave mode with MSM='1' in the TIM1_SMCR register.
- Configure Timer 3 to get an input trigger from Timer 1 (TS=000 in the TIM3_SMCR register)
- Configure Timer 3 for trigger mode (SMS='110' for TIM3_SMCR register).

When a rising edge occurs on TI1 of Timer 1, both timers synchronously start counting according to the internal clock and both TIF flags are set simultaneously.

Note: In this example, both timers are initialized (set the appropriate UG bit) before startup, and both counters start at 0. However, it is possible to insert an offset between the timers by writing to any of the counter registers (TIMx_CNT). *In the figure below, you can see that there is a delay between CNT_EN and CK_PSC of timer 1 in master/slave mode.*

22.3.15. Debug mode

When the chip enters the debug mode, the TIMx counter can continue to work normally or stop working according to the setting of DBG_TIMx_STOP in the DBG module.

22.4. Register description

TIM2 register base address: 0x4000 0000

TIM3 register base address: 0x4000 0400

22.4.1. TIM2/3 Control Register 1 (TIMx_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
-	-	-	-	-	-	RW		RW	RW		RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 10	Res	-	-	Res
9: 8	CKD [1:0]	RW	00	Clock Dividing Factor These 2 bits define the ratio between the frequency of the timer clock (CK_INT), the dead time and the dividing ratio between the dead time generator and the sampling clock used by the digital filters (ETR, Tlx) 00: tDTS = tCK_INT 01: tDTS = 2 x tCK_INT 10: tDTS = 4 x tCK_INT 11: Reserved, do not use this! Configuration
7	ARPE	RW	0	Auto Reload Preload Allowed Bit 0: TIMx_ARR register is not buffered 1: TIMx_ARR register is buffered
6: 5	CMS [1:0]	RW	00	Selecting Central Alignment Mode 00: Edge alignment mode. The counter counts up or down based on the direction bit (DIR). 01: Central Alignment Mode 1. The counter alternately counts up and down. Configured as an output channel (The output compare interrupt flag bit for (CCxS=00 in the TIMx_CMRx register) is set only when the counter counts down. 10: Central Alignment Mode 2. The counter alternately counts up and down. Configured as an output compare interrupt flag bit for the output channel (CCxS=00 in the TIMx_CMRx register), which is only set when the counter counts up. 11: Central Alignment Mode 3. The counter alternately counts up and down. Configured as an output compare interrupt flag bit for the output channel (CCxS=00 in the TIMx_CMRx register) that is set when the counter counts up and down.

Bit	Name	R/W	Reset Value	Function
				Note: While the counter is on (CEN=1), switching from edge-aligned mode to center-aligned mode is not allowed.
4	DIR	RW	0	<p>orientations</p> <p>0: Counter counts up</p> <p>1: Counter counts down</p> <p>Note: This bit is read-only when the counter is configured for center-aligned mode or encoder mode</p>
3	OPM	RW	0	<p>Single Pulse Mode</p> <p>0: counter does not stop when an update event occurs</p> <p>1: The counter stops when the next update event (clearing the CEN bit) occurs.</p>
2	URS	RW	0	<p>Update request source</p> <p>The software selects the source of UEV events with this bit</p> <p>0: If generating an update interrupt or DMA request is allowed, either of the following events generates an update interrupt or DMA request:</p> <ul style="list-style-type: none"> - Counter overflow/underflow - Setting the UG position - Updates generated from the schema controller <p>1: If generating update interrupts or DMA requests is allowed, only counter overflow/underflow generates an update interrupt or DMA request</p>
1	UDIS	RW	0	<p>prohibit updating</p> <p>Software allows/disallows the generation of UEV events with this bit</p> <p>0: UEV is allowed. Update (UEV) events are generated by any of the following events:</p> <ul style="list-style-type: none"> - Counter overflow/underflow - Setting the UG position - Updates generated from the schema controller <p>The cached registers are loaded with their preloaded values.</p> <p>1: UEV is prohibited. No update events are generated and the shadow registers (ARR,PSC,CCR_x) hold their values. If the UG bit is set or a hardware reset is issued from the mode controller, the counter and prescaler are reinitialized.</p>
0	CEN	RW	0	Allowed counters

Bit	Name	R/W	Reset Value	Function
				0: Disable counter 1: Turn on the counter Note: The external clock, gated mode and encoder mode will not work until the CEN bit is set in software. Trigger mode can be automatically set in hardware with the CEN bit.

22.4.2. TIM2/3 Control Register 2 (TIMx_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	TI1S	MMS [2:0]			CCDS	Res	Res	Res
-	-	-	-	-	-	-	-	RW	RW	RW	RW	RW	-	-	-

Bit	Name	R/W	Reset Value	Function
31: 8	Res	-	-	Res
7	TI1S	RW	0	TI1 Selection 0: The TIMx_ pin is connected to the TI1 input. 1: The TIMx_CH1, TIMx_CH2 and TIMx_CH3 pins are connected to the TI1 inputs via heterodyne.
6: 4	MMS [2:0]	RW	000	Master Mode Selection These two bits are used to select the synchronization message (TRGO) sent to the slave timer in master mode. The possible combinations are as follows: 000: Reset - The UG bit of the TIMx_EGR register is used as a trigger output (TRGO). If the trigger input (slave mode controller in reset mode) generates a reset, the signal on the TRGO relative to the actual reset There will be a delay. 001: Allowed - The counter enable signal CNT_EN is used as a trigger output (TRGO). Sometimes it is necessary to start multiple timers at the same time or to control a window from a timer. The counter enable signal is generated by a logical or of the CEN control bits and the trigger input signal in gated mode. When the counter enable signal is controlled on the trigger input, there is a delay on TRGO

Bit	Name	R/W	Reset Value	Function
				<p>unless master/slave mode is selected (see description of the MSM bit in the TIMx_SMCR register).</p> <p>010: Update - The update event is selected as a trigger input (TRGO). For example, a master timer clock may be used as a prescaler for a slave timer.</p> <p>011: Compare Pulse - Once a capture has occurred or a successful comparison has occurred, the trigger output sends a positive pulse (TRGO) when the CC1IF flag is to be set (i.e. is it already high).</p> <p>100: Compare - The OC1REF signal is used as a trigger output (TRGO).</p> <p>101: Compare - The OC2REF signal is used as a trigger output (TRGO).</p> <p>110: Compare - The OC3REF signal is used as a trigger output (TRGO).</p> <p>111: Compare - The OC4REF signal is used as a trigger output (TRGO).</p>
3	CCDS	RW	0	<p>DMA selection for capture/compare</p> <p>0: DMA request for CCx is sent when a CCx event occurs.</p> <p>1: Send a DMA request for CCx when an update event occurs.</p>
2: 0	Res	-	-	Res

22.4.3. TIM2/3 Slave Mode Control Register (TIMx_SMCR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS [1:0]		ETF [3:0]				MSM	TS [2:0]			OCCS	SMS [2:0]		
RW	RW	RW		RW				RW	RW			RW	RW		

Bit	Name	R/W	Reset Value	Function
31: 16	Res	-	-	Res
15	ETP	RW	0	<p>External trigger polarity</p> <p>This bit selects whether to use ETR or the inverse of ETR as the trigger operation</p> <p>0: ETR is not inverted and is active high or on the rising edge;</p>

Bit	Name	R/W	Reset Value	Function
				1: ETR is inverted, active low or falling edge.
14	ECE	RW	0	<p>External clock enable bit</p> <p>This bit enables external clock mode 2</p> <p>0: Disable external clock mode 2;</p> <p>1: Enable external clock mode 2. The counter is driven by any valid edge on the ETRF signal.</p> <p>Note 1: Setting the ECE bit has the same effect as selecting External Clock Mode 1 and connecting TRGI to ETRF (SMS=111 and TS=111).</p> <p>Note 2: The following slave modes can be used in conjunction with External Clock Mode 2: Reset Mode, Gated Mode, and Trigger Mode; however, TRGI cannot be connected to ETRF at this time (TS bit cannot be '111').</p> <p>Note 3: When External Clock Mode 1 and External Clock Mode 2 are both enabled, the external clock input is ETRF.</p>
13: 12	ETPS [1:0]	RW	0	<p>Externally triggered prescaler. The external trigger input signal ETR frequency must be at most 1/4 of the TIMxCLK frequency. The prescaler can be enabled to reduce the frequency of the ETRP.</p> <p>00: Prescaler off</p> <p>01: 2 crossover of ETRP frequency</p> <p>10: 4 divisions of the ETRP frequency</p> <p>11: ETRP frequency in 8 divisions</p>
11: 8	ETF [3:0]	RW	0	<p>External trigger filter</p> <p>These bits define the frequency at which the ETRP signal is sampled and the bandwidth at which the ETRP is digitally filtered. In effect, the digital filter is an event counter that records N events and then produces a jump in the output.</p> <p>0000: no filter, sampled at fDTS</p> <p>0001: Sampling frequency fSAMPLING=fCK_INT, N=2</p> <p>0010: Sampling frequency fSAMPLING=fCK_INT, N=4</p> <p>0011: Sampling frequency fSAMPLING=fCK_INT, N=8</p> <p>0100: Sampling frequency fSAMPLING=fDTS/2, N=6</p> <p>0101: Sampling frequency fSAMPLING = fDTS/2, N=8</p> <p>0110: Sampling frequency fSAMPLING = fDTS/4, N=6</p> <p>0111: Sampling frequency fSAMPLING=fDTS/4, N=8</p> <p>1000: Sampling frequency fSAMPLING = fDTS/8, N=6</p> <p>1001: Sampling frequency fSAMPLING = fDTS/8, N=8</p> <p>1010: Sampling frequency fSAMPLING = fDTS/16, N=5</p>

Bit	Name	R/W	Reset Value	Function
				<p>1011: Sampling frequency $f_{\text{SAMPLING}} = f_{\text{DTS}}/16$, $N=6$</p> <p>1100: Sampling frequency $f_{\text{SAMPLING}} = f_{\text{DTS}}/16$, $N=8$</p> <p>1101: Sampling frequency $f_{\text{SAMPLING}} = f_{\text{DTS}}/32$, $N=5$</p> <p>1110: Sampling frequency $f_{\text{SAMPLING}} = f_{\text{DTS}}/32$, $N=6$</p> <p>1111: Sampling frequency $f_{\text{SAMPLING}} = f_{\text{DTS}}/32$, $N=8$</p>
7	MSM	RW	0	<p>Master/slave mode (Master/slave mode)</p> <p>0: No effect;</p> <p>1: Events on the trigger input (TRGI) are delayed to allow perfect synchronization between the current timer (via TRGO) and its slave timer. This is useful when it is required to synchronize several timers to a single external event.</p>
6: 4	TS	RW	0	<p>Trigger selection</p> <p>These 3 bits select the trigger input used to synchronize the counter.</p> <p>000: Internal trigger 0 (ITR0) 100: Edge detector for TI1 (TI1F_ED)</p> <p>001: Internal Trigger 1 (ITR1) 101: Filtered Timer Input 1 (TI1FP1)</p> <p>010: Internal Trigger 2 (ITR2) 110: Filtered Timer Input 2 (TI2FP2)</p> <p>011: Internal trigger 3 (ITR3) 111: External trigger input (ETRF)</p> <p>For more details on ITRx, see Table 22-1.</p> <p>Note: These bits can only be changed if they are not used (e.g. SMS=000) to avoid false edge detection when changed.</p>
3	OCCS	RW	0	<p>OCREF clear selection bit (OCREF clear selection)</p> <p>This one chose the OCREF removal source.</p> <p>0: OCREF_CLR_INPUT connects OCREF_CLR inputs</p> <p>1: OCREF_CLR_INPUT connects to ETRF</p>
2: 0	SMS	RW	0	<p>Slave mode selection</p> <p>When an external signal is selected, the active edge of the trigger signal (TRGI) is related to the selected external input polarity (see description of the Input Control Register and Control Register)</p> <p>000: Off Slave Mode - If CEN=1, the prescaler is driven directly from the internal clock.</p> <p>001: Encoder Mode 1 - Depending on the level of TI1FP1, the counter counts up/down on the edge of TI2FP2.</p> <p>010: Encoder Mode 2 - Depending on the level of TI2FP2, the counter counts up/down on the edge of TI1FP1.</p> <p>011: Encoder Mode 3 - Depending on the input level of another signal, the counter counts up/down on the edges of TI1FP1 and TI2FP2.</p>

Bit	Name	R/W	Reset Value	Function
				<p>100: Reset Mode - The rising edge of the selected trigger input (TRGI) reinitializes the counter and generates a signal to update the register.</p> <p>101: Gated Mode - When the trigger input (TRGI) is high, the counter is clocked on. Once the trigger input goes low, the counter stops (but does not reset). The start and stop of the counter is controlled.</p> <p>110: Trigger Mode - The counter is started (but not reset) on the rising edge of the trigger input TRGI, only the start of the counter is controlled.</p> <p>111: External Clock Mode 1 - The rising edge of the selected trigger input (TRGI) drives the counter.</p> <p>Note: Do not use gating mode if TI1F_EN is selected as the trigger input (TS=100). This is because TI1F_ED outputs a pulse each time TI1F changes, however the gated mode is to check the level of the trigger input.</p>

Table 22-2 TIMx Internal Trigger Connections

From Timer	ITR0 (TS=000)	ITR1 (TS=001)	ITR2 (TS=010)	ITR3 (TS=011)
TIM2	TIM1	TIM15	TIM3	TIM14_OC
TIM3	TIM1	TIM2	TIM15	TIM14_OC

22.4.4. TIM2/3 DMA/Interrupt Enable Register (TIMx_DIER)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Re	TD	Re	CC4D	CC3D	CC2D	CC1	UD	R	TIE	Re	CC4I	CC3I	CC2IE	CC	UIE
s	E	s	E	E	E	DE	E	es		s	E	E		1IE	
-	RW	-	RW					-	RW	-	RW				

Bit	Name	R/W	Reset Value	Function
31: 15	Res	-	-	Res
14	TDE	RW	0	<p>TDE: Allow triggering of DMA requests</p> <p>0: Disable triggering of DMA request</p>

Bit	Name	R/W	Reset Value	Function
				1: Allow triggering of DMA requests
13	Res	-	-	Res
12	CC4DE	RW	0	CC4DE: Allows capture/comparison of DMA requests for 4 0: Disable capture/compare 4 DMA requests 1: Allow capture/comparison of DMA requests for 4
11	CC3DE	RW	0	CC3DE: Allows capture/comparison of DMA requests for 3 0: Disable capture/compare 3 DMA requests 1: Allow capture/comparison of 3 DMA requests
10	CC2DE	RW	0	CC2DE: Allows capture/comparison of 2 DMA requests 0: Disable capture/compare 2 DMA requests 1: Allow capture/compare 2 DMA requests
9	CC1DE	RW	0	CC1DE: Allow capture/compare 1 DMA requests 0: Capture/compare 1 DMA request disabled 1: Allow capture/compare 1 for DMA requests
8	UDE	RW	0	UDE: DMA request to allow updates 0: DMA requests for updates are disabled 1: Allow updated DMA requests
7	Res	-	-	Res
6	TIE	RW	0	TIE: Allowed to trigger an interrupt 0: Disable interrupt triggering 1: Allow triggering of interrupts
5	Res	-	-	Res
4	CC4IE	RW	0	CC4IE: Allow capture/compare 4 interrupts 0: Capture/compare 4 interrupt disabled 1: Capture/compare 4 interrupts allowed
3	CC3IE	RW	0	CC3IE: Allow capture/compare 3 interrupts 0: Capture/compare 3 interrupt disabled 1: Capture/compare 3 interrupts allowed
2	CC2IE	RW	0	CC2IE: Capture/Compare 2 interrupts allowed 0: Capture/compare 2 interrupt disabled 1: Capture/compare 2 interrupts allowed
1	CC1IE	RW	0	CC1IE: Capture/Compare 1 interrupt allowed 0: Capture/compare 1 interrupt disabled 1: Allow capture/compare 1 interrupt
0	UIE	RW	0	UIE: Allow update interruptions 0: Disable update interrupt 1: Allow updates to be interrupted

22.4.5. TIM2/3 status register (TIMx_SR)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res			Res	Res	Res	Res	Res	IC4IF	IC3IF	IC2IF	IC1IF	IC4IR	IC3IR	IC2IR	IC1IR
-			-	-	-	-	-	RC_W0							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res			CC4OF	CC3OF	CC2OF	CC1OF	Res		TIF	Res	CC4IF	CC3IF	CC2IF	CC1IF	UIF
-	-	-	RC_W0				-	RC_W0	-	RC_W0					

Bit	Name	R/W	Reset Value	Function
31: 24	Res	-	-	Res
23	IC4IF	RC_W0	0	Capture 4 flags on falling edge See IC1IF description.
22	IC3IF	RC_W0	0	Capture 3 flag on falling edge See IC1IF description.
21	IC2IF	RC_W0	0	Capture 2 flag on falling edge See IC1IF description.
20	IC1IF	RC_W0	0	Capture 1 flag on falling edge This flag can be set to 1 by hardware only if the corresponding channel is configured for input capture and the capture event is triggered by a falling edge. It is cleared '0' by software or by reading TIMx_CCR1. 0: No falling edge capture is generated; 1: A falling edge capture event occurs.
19	IC4IR	RC_W0	0	Rising edge capture 4 flags See IC1IR for description.
18	IC3IR	RC_W0	0	Rising edge capture 3 flag See IC1IR for description.
17	IC2IR	RC_W0	0	Rising edge capture 2 flag See IC1IR for description.
16	IC1IR	RC_W0	0	Rising edge capture 1 flag This flag can be set to 1 by hardware only if the corresponding channel is configured for input capture and the capture event is triggered by a rising edge. It is cleared '0' by software or by reading TIMx_CCR1.

Bit	Name	R/W	Reset Value	Function
				0: No rising edge capture is generated; 1: A rising edge capture event occurs.
15: 13	Res	-	-	Res
12	CC4OF	RC_W0	0	Capture/Compare 4 overcapture flag See CC1OF for description.
11	CC3OF	RC_W0	0	Capture/Compare 3 overcapture flag See CC1OF for description.
10	CC2OF	RC_W0	0	Capture/Compare 2 overcapture flag See CC1OF for description.
9	CC1OF	RC_W0	0	Capture/Compare 1 overcapture flag This flag can be set to 1 by hardware only if the corresponding channel is configured for input capture. Writing a 0 clears the bit. 0: No duplicate captures are generated; 1: The state of CC1IF is already '1' when the counter value is captured into the TIMx_CCR1 register.
8: 7	Res	-	-	Res
6	TIF	RC_W0	0	Trigger interrupt flag (Trigger interrupt flag) '1' to this position by hardware when a trigger event occurs (a valid edge is detected on the TRGI input when the slave mode controller is in a mode other than gated mode, or either edge in gated mode). It is cleared '0' by the software. 0: No trigger event is generated; 1: Trigger an interrupt to wait for a response.
5	Res	-	-	Res
4	CC4IF	RC_W0	0	Capture/Compare 4 interrupt flag Refer to the CC1IF description.
3	CC3IF	RC_W0	0	Capture/Compare 3 interrupt flag Refer to the CC1IF description.
2	CC2IF	RC_W0	0	Capture/Compare 2 interrupt flag Refer to the CC1IF description.
1	CC1IF	RC_W0	0	Capture/Compare 1 interrupt flag (Capture/Compare 1 interrupt flag) If channel CC1 is configured for output mode: This bit is set by hardware to 1 one clock cycle after the counter value matches the comparison value, except in center-symmetric mode (refer to the CMS bit in the TIMx_CR1 register). It is cleared '0' by the software. 0: No match occurred;

Bit	Name	R/W	Reset Value	Function
				<p>1: The value of TIMx_CNT matches the value of TIMx_CCR1. When the content of TIMx_CCR1 is greater than the content of TIMx_APR, the CC1IF bit goes high for a counter overflow condition in up or up/down counting modes, or a counter underflow condition in down counting mode</p> <p>If channel CC1 is configured for input mode:</p> <p>This bit is set '1' by hardware when a capture event occurs and it is cleared '0' by software or by reading TIMx_CCR1.</p> <p>0: No input capture is generated;</p> <p>1: The counter value has been captured (copied) to TIMx_CCR1 (an edge of the same polarity as selected is detected on IC1).</p>
0	UIF	RC_W0	0	<p>Update interrupt flag (Update interrupt flag)</p> <p>This bit is set '1' by hardware when an update event is generated. It is cleared '0' by the software.</p> <p>0: No update events are generated;</p> <p>1: Update interrupt waiting for response. This bit is set '1' by hardware when the register is updated:</p> <ul style="list-style-type: none"> - If UDIS=0 in the TIMx_CR1 register, an update event is generated when the repeat counter value overflows or underflows (repeat counter=0). - If URS=0 and UDIS=0 of the TIMx_CR1 register, an update event is generated when UG=1 of the TIMx_EGR register is set, and when the counter CNT is re-initialized by software. - If URS=0 and UDIS=0 in the TIMx_CR1 register, when the counter CNT is reinitialized by the trigger event. (Refer to TIM2/3 Slave Mode Control Register (TIMx_SMCR)).

22.4.6. TIM2/3 Event Generation Register (TIMx_EGR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	Res	TG	Res	CC4G	CC3G	CC2G	CC1G	UG
-	-	-	--	-	-	-	-	-	W	-	W	W	W	W	W

Bit	Name	R/W	Reset Value	Function
31: 7	Res	-	-	Res
6	TG	W	0	<p>generate a trigger event</p> <p>This bit is set to 1 by software to generate a trigger event that is automatically cleared to 0 by hardware.</p> <p>0: No action;</p> <p>1: TIF=1 in the TIMx_SR register generates the corresponding interrupt and DMA if they are turned on.</p>
5	Res	-	-	Res
4	CC4G	W	0	<p>Generate capture/compare 4 events</p> <p>Refer to CC1G description</p>
3	CC3G	W	0	<p>Generate Capture/Compare 3 events</p> <p>Refer to CC1G description</p>
2	CC2G	W	0	<p>Generate Capture/Compare 2 events</p> <p>Refer to CC1G description</p>
1	CC1G	W	0	<p>Generate Capture/Compare 1 events</p> <p>This bit is set to 1 by software to generate a capture/compare event that is automatically cleared to 0 by hardware.</p> <p>0: No action;</p> <p>1: Generate a capture/compare event on channel CC1:</p> <p>If channel CC1 is configured as an output:</p> <p>Set CC1IF=1 to generate the corresponding interrupt and DMA if it is turned on.</p> <p>If channel CC1 is configured as an input:</p> <p>The current counter value is captured to the TIMx_CCR1 register, CC1IF=1 is set, and the corresponding interrupt and DMA are generated if they are turned on. If CC1IF is already 1, set CC1OF=1.</p>
0	UG	W	0	<p>Generating update events</p> <p>This bit is set to 1 by software and cleared to 0 automatically by hardware.</p> <p>0: No action;</p> <p>1: Reinitialize the counter and generate an update event.</p> <p>Note that the prescaler counter is also cleared to 0 (but the prescaler</p> <p>(The coefficients are unchanged). The counter is cleared to 0 if in centrosymmetric mode or DIR=0 (counting up), if DIR=1 (counting down) the counter takes the value of TIMx_ARR.</p>

22.4.7. TIM2/3 Capture/Compare Mode Register 1 (TIMx_CM1)

Address offset: 0x18

Reset value: 0x0000 0000

Output compare mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2: 0]			OC2PE	CO2FE	CC2S[1: 0]		OC1CE	OC1M[2: 0]			OC1PE	OC1FE	CC1S[1: 0]	
IC2F[3: 0]				IC2PSC[1: 0]				IC1F[3: 0]			IC1PSC[1: 0]				
RW															

Output compare mode

Bit	Name	R/W	Reset Value	Function
31: 16	Res	-	-	Res
15	OC2CE	RW	0	Output compare 2 clear 0 enable
14: 12	OC2M[2: 0]	RW	000	Output Compare 2 Mode Selection
11	OC2PE	RW	0	Output compare 2 preload enable
10	OC2FE	RW	0	Output compare 2 fast enable
9: 8	CC2S[1: 0]	RW	00	<p>Capture/Compare 2 Selection.</p> <p>This bit defines the direction of the channel (input/output), and the selection of the input pin:</p> <p>00: The CC2 channel is configured as an output;</p> <p>01: CC2 channel is configured as an input and IC2 is mapped on TI2;</p> <p>10: The CC2 channel is configured as an input and IC2 is mapped on TI1;</p> <p>11: The CC2 channel is configured as an input and IC2 is mapped on TRC. This mode works only when the internal trigger input is selected.</p> <p>(selected by the TS bit of the TIMx_SMCR register).</p> <p>Note: CC2S is writable only when the channel is closed (CC2E=0 in the TIMx_CCER register).</p>
7	OC1CE	RW	0	<p>Output compare 1 clear 0 enable</p> <p>0: OC1REF is not affected by the ETRF input;</p> <p>1: Clear OC1REF=0 as soon as the ETRF input is detected high.</p>
6: 4	OC1M[2: 0]	RW	00	<p>Output Comparison 1 Mode</p> <p>This bit defines the action of the output reference signal OC1REF, which determines the value of OC1, OC1N.</p>

Bit	Name	R/W	Reset Value	Function
				<p>OC1REF is active high, while the active levels of OC1 and OC1N depend on the CC1P and CC1NP bits.</p> <p>000: Frozen. Comparison between output comparison register TIMx_CCR1 and counter TIMx_CNT does not work for OC1REF.</p> <p>Use;</p> <p>001: Set channel 1 as the effective level for matching. When the value of counter TIMx_CNT is the same as the capture/comparison register, the value of TIMx_CNT is the same as the value of TIMx_CNT.</p> <p>1 (TIMx_CCR1) is the same, force OC1REF high.</p> <p>010: Set channel 1 as the invalid level for matching. When the value of counter TIMx_CNT is the same as the capture/comparison register, the value of TIMx_CNT is the same as the value of TIMx_CNT.</p> <p>1 (TIMx_CCR1) is the same, force OC1REF low.</p> <p>011: Flip. Flips the level of OC1REF when TIMx_CCR1 = TIMx_CNT.</p> <p>100: Forced to an invalid level. Force OC1REF to be low.</p> <p>101: Forced to active level. Force OC1REF to be high.</p> <p>110: PWM Mode 1- In up count, channel 1 is active once TIMx_CNT < TIMx_CCR1, otherwise it is invalid; in down count, channel 1 is invalid once TIMx_CNT > TIMx_CCR1 (OC1REF=0), otherwise it is valid (OC1REF= 1).</p> <p>111: PWM Mode 2- In up count, channel 1 is invalid once TIMx_CNT < TIMx_CCR1, otherwise it is valid; in down count, channel 1 is valid once TIMx_CNT > TIMx_CCR1, otherwise it is invalid.</p> <p>Note 1: This bit cannot be modified once the LOCK level is set to 3 (LOCK bit in the TIMx_BDTR register) and CC1S=00 (the channel is configured as an output).</p> <p>Note 2: In PWM mode 1 or PWM mode 2, the OC1REF level changes only when the comparison result is changed or when switching from freeze mode to PWM mode in the output comparison mode.</p>
3	OC1PE	RW	0	<p>Output compare 1 preload enable</p> <p>0: Disables the preload function of the TIMx_CCR1 register, which can be written to the TIMx_CCR1 register at any time and the new value takes effect immediately.</p>

Bit	Name	R/W	Reset Value	Function
				<p>1: Enable the preload function of TIMx_CCR1 register, the read and write operation only operates on the preloaded register, the preloaded value of TIMx_CCR1 is loaded into the current register when the update event arrives.</p> <p>Note 1: Once the LOCK level is set to 3 (LOCK bit in the TIMx_BDTR register) and CC1S=00 (the channel is configured as a output) then the bit cannot be modified.</p> <p>Note 2: In single pulse mode only, PWM mode can be used without checking the preload register, otherwise its action is uncertain.</p>
2	OC1FE	RW	0	<p>Output Compare 1 Fast Enable</p> <p>This bit is used to speed up the response of the CC output to trigger input events.</p> <p>0: Depending on the value of the counter and CCR1, CC1 operates normally, even if the trigger is open. The minimum delay to activate the CC1 output is 5 clock cycles when the input of the flip-flop has an active edge.</p> <p>1: The active edge of the input to the trigger acts as if a comparison match has occurred. Therefore, OC is set to the comparison level and the</p> <p>Irrelevant to the comparison of results. The delay between the active edge of the sampling flip-flop and the CC1 output is reduced to 3 clock cycles.</p> <p>OCFE's only work when the channel is configured for PWM1 or PWM2 mode.</p>
1: 0	CC1S[1: 0]	RW	00	<p>Capture/Compare1 Select.</p> <p>These 2 bits define the direction of the channel (input/output), and the selection of the input pin:</p> <p>00: CC1 channel is configured as an output;</p> <p>01: CC1 channel is configured as an input and IC1 is mapped on TI1;</p> <p>10: CC1 channel is configured as an input and IC1 is mapped on TI2;</p> <p>11: CC1 channel is configured as an input and IC1 is mapped on TRC. This mode works only when the internal trigger input is selected.</p> <p>(selected by the TS bit of the TIMx_SMCR register).</p>

Bit	Name	R/W	Reset Value	Function
				Note: CC1S is writable only when the channel is closed (CC1E=0 in the TIMx_CCER register).

Input Capture Mode

Bit	Name	R/W	Reset Value	Function
31: 16	Res	-	-	Res
15: 12	IC2F[3: 0]	RW	0000	Input Capture 2 Filter
11: 10	IC2PSC[1: 0]	RW	00	Capture/Compare 2 Prescaler
9: 8	CC2S[1: 0]	RW	0	<p>Capture/Compare 2 Selection.</p> <p>These 2 bits define the direction of the channel (input/output), and the selection of the input pin:</p> <p>00: The CC2 channel is configured as an output;</p> <p>01: CC2 channel is configured as an input and IC2 is mapped on TI2;</p> <p>10: The CC2 channel is configured as an input and IC2 is mapped on TI1;</p> <p>11: The CC2 channel is configured as an input and IC2 is mapped on TRC. This mode works only when the internal trigger input is selected.</p> <p>(selected by the TS bit of the TIMx_SMCR register).</p> <p>Note: CC2S is writable only when the channel is closed (CC2E=0 in the TIMx_CCER register).</p>
7: 4	IC1F[3: 0]	RW	0000	<p>Input Capture 1 Filter</p> <p>These bits define the sampling frequency of the TI1 input and the digital filter length. The digital filter consists of an event counter that</p> <p>It records N events and then produces a jump in output:</p> <p>0000: no filter, sampled at fDTS</p> <p>0001: Sampling frequency fSAMPLING=fCK_INT, N=2</p> <p>0010: Sampling frequency fSAMPLING=fCK_INT, N=4</p> <p>0011: Sampling frequency fSAMPLING=fCK_INT, N=8</p> <p>0100: Sampling frequency fSAMPLING=fDTS/2, N=6</p> <p>0101: Sampling frequency fSAMPLING = fDTS/2, N=8</p> <p>0110: Sampling frequency fSAMPLING = fDTS/4, N=6</p> <p>0111: Sampling frequency fSAMPLING=fDTS/4, N=8</p> <p>1000: Sampling frequency fSAMPLING = fDTS/8, N=6</p> <p>1001: Sampling frequency fSAMPLING = fDTS/8, N=8</p> <p>1010: Sampling frequency fSAMPLING = fDTS/16, N=5</p> <p>1011: Sampling frequency fSAMPLING=fDTS/16, N=6</p>

Bit	Name	R/W	Reset Value	Function
				1100: Sampling frequency $f_{\text{SAMPLING}} = f_{\text{DTS}}/16$, $N=8$ 1101: Sampling frequency $f_{\text{SAMPLING}} = f_{\text{DTS}}/32$, $N=5$ 1110: Sampling frequency $f_{\text{SAMPLING}} = f_{\text{DTS}}/32$, $N=6$ 1111: Sampling frequency $f_{\text{SAMPLING}} = f_{\text{DTS}}/32$, $N=8$
3: 2	IC1PSC[1: 0]	RW	00	Capture/Compare 1 prescaler 00: No prescaler, each edge detected on the capture input triggers a capture; 01: Capture is triggered every 2 events; 10: Capture is triggered every 4 events; 11: Capture is triggered every 8 events.
1: 0	CC1S[1: 0]	RW	00	CC1S[1:0]: capture/compare 1 selection. These 2 bits define the direction of the channel (input/output), and the selection of the input pin: 00: CC1 channel is configured as an output; 01: CC1 channel is configured as an input and IC1 is mapped on TI1; 10: CC1 channel is configured as an input and IC1 is mapped on TI2; 11: CC1 channel is configured as an input and IC1 is mapped on TRC. This mode works only when the internal trigger input is selected. (selected by the TS bit of the TIMx_SMCR register). Note: CC1S is writable only when the channel is closed (CC1E=0 in the TIMx_CCER register).

22.4.8. TIM2/3 Capture/Compare Mode Register 2 (TIMx_CMCR2)

Address offset: 0x1C

Reset value: 0x0000 0000

Output compare mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4CE	OC4M[2: 0]			OC4PE	CO4FE	CC4S[1: 0]		OC3CE	OC3M[2: 0]			OC3PE	OC3FE	CC3S[1: 0]	
IC4F[3: 0]			IC4PSC[1: 0]					IC3F[3: 0]			IC3PSC[1: 0]				
RW															

Output Comparison Mode

Bit	Name	R/W	Reset Value	Function
31: 16	Res	-	-	Res

15	OC4CE	RW	0	Output compare 4 clear 0 enable
14: 12	OC4M[2: 0]	RW	000	Output Comparison 4 Mode
11	OC4PE	RW	0	Output Compare 4 Preload Enable
10	OC4FE	RW	0	Output Compare 4 Fast Enable
9: 8	CC4S[1: 0]	RW	00	<p>Capture/Compare 4 Selection.</p> <p>This bit defines the direction of the channel (input/output), and the selection of the input pin:</p> <p>00: The CC4 channel is configured as an output;</p> <p>01: CC4 channel is configured as an input and IC4 is mapped on TI4;</p> <p>10: The CC4 channel is configured as an input and IC4 is mapped on TI3;</p> <p>11: The CC4 channel is configured as an input and IC4 is mapped on TRC. This mode works only when the internal trigger input is selected.</p> <p>(selected by the TS bit of the TIMx_SMCR register).</p> <p>Note: CC4S is writable only when the channel is closed (CC4E=0 in the TIMx_CCER register).</p>
7	OC3CE	RW	0	Output compare 3 clear 0 enable
6: 4	OC3M[2: 0]	RW	00	Output Comparison 3 Mode
3	OC3PE	RW	0	Output Compare 3 Preload Enable
2	OC3FE	RW	0	Output Compare 3 Fast Enable
1: 0	CC3S[1: 0]	RW	00	<p>Capture/Compare 3 options.</p> <p>These 2 bits define the direction of the channel (input/output), and the selection of the input pin:</p> <p>00: The CC3 channel is configured as an output;</p> <p>01: CC3 channel is configured as an input and IC3 is mapped on TI3;</p> <p>10: CC3 channel is configured as an input and IC3 is mapped on TI4;</p> <p>11: CC3 channel is configured as an input and IC3 is mapped on TRC. This mode works only when the internal trigger input is selected.</p> <p>(selected by the TS bit of the TIMx_SMCR register).</p> <p>Note: CC3S is writable only when the channel is closed (CC3E=0 in the TIMx_CCER register).</p>

Input Capture Mode

Bit	Name	R/W	Reset Value	Function
31: 16	Res	-	-	Res

Bit	Name	R/W	Reset Value	Function
15: 12	IC4F[3: 0]	RW	0000	Input Capture 4 Filter
11: 10	IC4PSC[1: 0]	RW	00	Capture/compare 4 prescalers
9: 8	CC4S[1: 0]	RW	0	<p>Capture/Compare 4 Selection.</p> <p>These 2 bits define the direction of the channel (input/output), and the selection of the input pin:</p> <p>00: The CC4 channel is configured as an output;</p> <p>01: CC4 channel is configured as an input and IC4 is mapped on TI4;</p> <p>10: The CC4 channel is configured as an input and IC4 is mapped on TI3;</p> <p>11: The CC4 channel is configured as an input and IC4 is mapped on TRC. This mode works only when the internal trigger input is selected.</p> <p>(selected by the TS bit of the TIMx_SMCR register).</p> <p>Note: CC4S is writable only when the channel is closed (CC4E=0 in the TIMx_CCER register).</p>
7: 4	IC3F[3: 0]	RW	0000	Input Capture 3 Filter
3: 2	IC3PSC[1: 0]	RW	00	Capture/Compare 3 Prescaler
1: 0	CC3S[1: 0]	RW	00	<p>Capture/Compare 3 options.</p> <p>These 2 bits define the direction of the channel (input/output), and the selection of the input pin:</p> <p>00: The CC3 channel is configured as an output;</p> <p>01: CC3 channel is configured as an input and IC3 is mapped on TI3;</p> <p>10: CC3 channel is configured as an input and IC3 is mapped on TI3;</p> <p>11: CC3 channel is configured as an input and IC3 is mapped on TRC. This mode works only when the internal trigger input is selected.</p> <p>(selected by the TS bit of the TIMx_SMCR register).</p> <p>Note: CC3S is writable only when the channel is closed (CC3E=0 in the TIMx_CCER register).</p>

22.4.9. TIM2/3 Capture/Compare Enable Register (TIMx_CCER)

Address offset: 0x20

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4N P	Re s	CC4 P	CC4 E	CC3N P	Re s	CC3 P	CC3 E	CC2N P	Re s	CC2 P	CC2 E	CC1N P	Re s	CC1 P	CC1 E
RW	-	RW	RW	RW	-	RW	RW	RW	-	RW	RW	RW	-	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 16	Res	-	-	Res
15	CC4NP	RW	0	Capture/compare 4 complementary output polarity. Refer to the description of CC1NP.
14	Res	-	-	Res
13	CC4P	RW	0	Capture/compare 4 output polarity. Refer to the description of CC1P.
12	CC4E	-	0	Capture/Compare 4 output enable. Refer to the description of CC1E.
11	CC3NP	RW	0	Capture/compare 3 complementary output polarities. Refer to the description of CC1NP.
10	Res	-	-	Res
9	CC3P	RW	0	Capture/compare 3 output polarity. Refer to the description of CC1P.
8	CC3E	RW	0	Capture/Compare 3 output enable. Refer to the description of CC1E.
7	CC2NP	RW	0	Capture/compare 2 complementary output polarities. Refer to the description of CC1NP.
6	Res	-	-	Res
5	CC2P	RW	0	Capture/Compare 2 Output Polarity. Refer to the description of CC1P.
4	CC2E	RW	0	Capture/Compare 2 output enable. Refer to the description of CC1E.
3	CC1NP	RW	0	Capture/compare 1 complementary output polarity 0: OC1N high level active 1: OC1N active-low This is used in conjunction with CC1P to define the TI1FP1/TI2FP1 polarity, refer to the CC1P description.
2	Res	-	-	Res
1	CC1P	RW	0	Capture/Compare 1 Output Polarity The CC1 channel is configured as an output: 0: OC1 active high 1: OC1 active low The CC1 channel is configured as an input:

Bit	Name	R/W	Reset Value	Function
				<p>The CC1NP/CC1P bits select the polarity of TI1FP1 and TI2FP1 used as trigger or capture signals.</p> <p>00: No inversion/rising edge:</p> <p>TIxFP1 rising edge active (capture, trigger in reset mode, external clock, or trigger mode);</p> <p>TIxFP1 is not inverted (gated mode, encoder mode).</p> <p>01: Inverted/falling edge:</p> <p>TIxFP1 falling edge active (capture, trigger in reset mode, external clock, or trigger mode);</p> <p>TIxFP1 inversion (gated mode, encoder mode).</p> <p>10: Reserved, do not use this configuration.</p> <p>11: Non-inverting/double-edge</p> <p>TIxFP1 is valid on both rising and falling edges (capture, triggered in reset mode, externally clocked, or in trigger mode);</p> <p>TIxFP1 is not inverted (gated mode). This configuration cannot be applied in encoder mode.</p>
0	CC1E	RW	0	<p>Capture/compare 1 output enable</p> <p>The CC1 channel is configured as an output:</p> <p>0: off - OC1 disable output.</p> <p>1: ON - OC1 signal output to the corresponding output pin.</p> <p>The CC1 channel is configured as an input:</p> <p>This bit determines whether the counter value can be captured into the TIMx_CCR1 register.</p> <p>0: Capture disabled;</p> <p>0: Capture enable</p>

Output control for standard OCx channels

CcxE bit	OCx output State
0	Output inhibit (OCx=0,OCx_EN=0)
1	OCx=OCxREF+Polarity,OCx_EN=1

22.4.10. TIM2/3 counter (TIMx_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNT [31:16] (TIM2 only)															
RW															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT [15:0]															

RW

Bit	Name	R/W	Reset Value	Function
31: 16	CNT [31:16]	RW	0	Counter value (TIM2 only)
15: 0	CNT [15:0]	RW	0	Counter value

22.4.11. TIM2/3 prescaler (TIMx_PSC)

Address offset: 0x28

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 16	Res	-	-	Res
15: 0	PSC [15:0]	RW	0	Prescaler value The counter clock frequency (CK_CNT) is equal to $f_{CK_PSC}/(PSC[15:0]+1)$. The PSC contains the value loaded into the current pre-scaler register each time an update event is generated; an update event consists of the counter being The UG bit of TIM_EGR is cleared '0' or is cleared '0' by a slave controller operating in reset mode.

22.4.12. TIM2/3 Automatic Reload Register (TIMx_ARR)

Address offset: 0x2C

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARR [31:16] (TIM2 only)															
RW															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR [15:0]															
RW															

Bit	Name	R/W	Reset Value	Function
31: 16	ARR [31:16].	RW	FFFF	Value for automatic reloading (TIM2 only)

15: 0	ARR [15:0]	RW	FFFF	<p>Auto-reload values</p> <p>The ARR contains the value that will be loaded into the actual auto-reload register.</p> <p>Refer to 22.3.1: Updates and actions of the time base unit regarding ARR for details.</p> <p>The counter does not work when the value of Auto-Reload is empty.</p>
-------	------------	----	------	---

22.4.13. TIM2/3 Capture/Compare Register 1 (TIMx_CCR1)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR1 [31:16] (TIM2 only)															
RW/RO															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15: 0]															
RW/RO															

Bit	Name	R/W	Reset Value	Function
31: 16	CCR1[31: 16]	RW	0	Capture/compare the value of 1 (TIM2 only)
15: 0	CCR1[15: 0]	RW	0	<p>Capture/compare the value of 1</p> <p>If the CC1 channel is configured as an output: CCR1 contains the value loaded into the current Capture/Compare 1 register (preloaded value). If the preload feature is not selected in the TIMx_CMR1 register (OC1PE bit), it is always loaded into the current register.</p> <p>Otherwise, this preloaded value is loaded into the current Capture/Compare 1 register only when an update event occurs.</p> <p>The current capture/compare register contains the value to be compared with counter TIMx_CNT and is signaled on the OC1 port.</p> <p>If the CC1 channel is configured as an input: CCR1 contains the counter value transmitted by the last input capture 1 event (IC1).</p>

22.4.14. TIM2/3 Capture/Compare Register 2 (TIMx_CCR2)

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR2 [31:16] (TIM2 only)															
RW/RO															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15: 0]															
RW/RO															

Bit	Name	R/W	Reset Value	Function
31: 16	CCR2[31: 16]	RW	0	Capture/compare the value of 2 (TIM2 only)
15: 0	CCR2[15: 0]	RW	0	<p>Capture/compare the value of 2</p> <p>If the CC2 channel is configured as an output: CCR2 contains the value loaded into the current Capture/Compare 2 register (preloaded value).</p> <p>If the preload feature is not selected in the TIMx_CMR1 register (OC2PE bit), it is always loaded into the current register.</p> <p>Otherwise, this preloaded value is loaded into the current Capture/Compare 2 register only when an update event occurs.</p> <p>The current capture/compare register contains the value to be compared with counter TIMx_CNT and signaled on the OC port.</p> <p>If the CC2 channel is configured as an input: CCR2 contains the counter value transmitted by the last input capture 2 event (IC2).</p>

22.4.15. TIM2/3 Capture/Compare Register 3 (TIMx_CCR3)

Address offset: 0x3C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR3 [31:16] (TIM2 only)															
RW/RO															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15: 0]															
RW/RO															

Bit	Name	R/W	Reset Value	Function
31: 16	CCR3[31: 16]	RW	0	Capture/compare the value of 3 (TIM2 only)
15: 0	CCR3[15: 0]	RW	0	<p>Capture/compare the value of 3</p> <p>If the CC3 channel is configured as an output:</p> <p>CCR3 contains the value loaded into the current Capture/Compare 3 register (preloaded value).</p> <p>If the preload feature is not selected in the TIMx_CMR2 register (OC3PE bit), it is always loaded into the current register.</p> <p>Otherwise, this preloaded value is loaded into the current Capture/Compare 3 register only when an update event occurs.</p> <p>The current capture/compare register contains the value to be compared with counter TIMx_CNT and signaled on the OC port.</p> <p>If the CC3 channel is configured as an input:</p> <p>CCR3 contains the counter value transmitted by the last input capture 3 event (IC3).</p>

22.4.16. TIM2/3 Capture/Compare Register 4 (TIMx_CCR4)

Address offset: 0x40

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR4 [31:16] (TIM2 only)															
RW/RO															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15: 0]															
RW/RO															

Bit	Name	R/W	Reset Value	Function
31: 16	CCR4[31: 16]	RW	0	Capture/compare the value of 4 (TIM2 only)
15: 0	CCR4[15: 0]	RW	0	<p>Capture/compare the value of 4</p> <p>If the CC4 channel is configured as an output:</p> <p>CCR4 contains the value loaded into the current Capture/Compare 4 register (preloaded value).</p> <p>If the preload feature is not selected in the TIMx_CMR2 register (OC4PE bit), it is always loaded into the current register.</p>

				<p>Otherwise, this preloaded value is loaded into the current Capture/Compare 4 register only when an update event occurs.</p> <p>The current capture/compare register contains the value to be compared with counter TIMx_CNT and signaled on the OC port.</p> <p>If the CC4 channel is configured as an input:</p> <p>CCR4 contains the counter value transmitted by the last input capture 4 event (IC4).</p>
--	--	--	--	---

22.4.17. TIM2/3 DMA Control Register (TIMx_DCR)

Address offset: 0x48

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 13	Res	-	-	Res
12: 8	DBL [4:0]	RW	0 0000	<p>DMA sequential transfer length</p> <p>These bits define the transfer length of the DMA in continuous mode (when reading or writing to the address of the TIMx_DMAR register)</p> <p>(when the timer then performs one consecutive transmission), i.e.: defines the number of times it is transmitted:</p> <p>00000: 1 transmission</p> <p>00001: 2 transmissions</p> <p>00010: 3 transmissions</p> <p>.....</p> <p>.....</p> <p>10001: 18 transmissions</p> <p>Example: let's consider this transmission: DBL=7,</p> <p>DBA=TIMx_CR1</p> <p>- If DBL=7 and DBA=TIMx_CR1 indicates the address of the data to be transmitted, then the address to be transmitted is given by the following equation:</p>

Bit	Name	R/W	Reset Value	Function
				<p>(address of TIMx_CR1) + DBA + (DMA index), where DMA index = DBL</p> <p>Where (address of TIMx_CR1) + DBA plus 7 gives the address where the data will be written or read, so that the transfer of data will take place in the 7 registers starting from the address (address of TIMx_CR1) + DBA.</p> <p>Depending on the DMA data length setting, the following may occur:</p> <ul style="list-style-type: none"> - If the data is set to a half word (16 bits), then the data is transferred to all 7 registers. - If the data is set to byte, the data is still transferred to all seven registers: the first register contains the first MSB byte, the second register contains the first LSB byte, and so on. Therefore for timers, the user must specify the width of the data to be transferred by the DMA.
7: 5	Res	-	-	Res
4: 0	DBA [4:0]	RW	0 0000	<p>DBA[4:0]: DMA base address</p> <p>These bits define the base address of the DMA in continuous mode (when the address of the TIMx_DMAR register is read or written)</p> <p>time), DBA is defined as the offset from the address where the TIMx_CR1 register is located:</p> <p>00000: TIMx_CR1, 00001: TIMx_CR2, 00010: TIMx_SMCR. </p>

22.4.18. DMA address for TIM2/3 continuous mode (TIMx_DMAR)

Address offset: 0x4C

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 0	DMAB [31:0]	RW	0	<p>DMA register for burst accesses</p> <p>A read or write to the TIMx_DMAR register results in an access operation to the register where the following address is located:</p> <p>$\text{TIMx_CR1 address} + (\text{DBA} + \text{DMA index}) \times 4$, where:</p> <p>"TIMx_CR1 Address" is the address where control register 1 (TIMx_CR1) is located;</p> <p>"DBA" is the base address defined in the TIMx_DCR register;</p> <p>The "DMA Index" is an offset that is automatically controlled by the DMA and depends on the DBL defined in the TIMx_DCR register.</p>

23. Basic Timer (TIM6/7)

23.1. Introduction to TIM6 and TIM7

The basic timers TIM6 and TIM7 each contain a 16-bit auto-reload counter driven by their respective programmable prescalers.

They can be used as general-purpose timers to provide time references and, in particular, to provide triggering for digital-to-analog converters (DACs). In fact, they are connected directly to the DAC inside the chip and drive the DAC directly via the trigger output.

Both TIM6 and TIM7 are completely independent and do not share any resources with each other.

23.2. Key features of TIM6 and TIM7

- 16-bit auto-reload up counter
- 16-bit programmable (can be modified in real time) prescaler, counter clock frequency crossover coefficient of any value between 1 and 65536
- Synchronization circuit for triggering DAC
- Generate an interrupt/DMA when an update event occurs

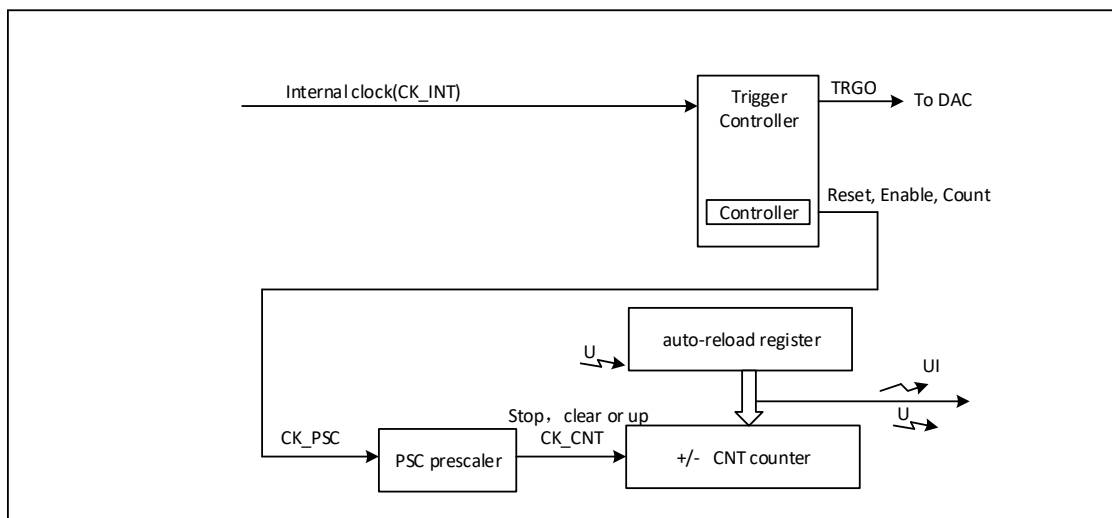


Figure 23-1 Block diagram of basic timer architecture

23.3. TIM6 and TIM7 Functional Description

23.3.1. Time base unit (in computing)

The main part of this programmable timer is a 16-bit up counter with auto-reload, the clock of the counter is obtained through a prescaler.

The software can read and write counters, auto-reload registers, and prescaler registers, and operate them even when the counters are running.

The time base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Automatic Reload Register (TIMx_ARR)

The auto-reload registers are preloaded, and writing or reading the auto-reload registers will access the preloaded registers. Depending on the setting of the auto-reload preload enable bit (ARPE) in the TIMx_CR1 register, the contents of the preload register are always or at each update event UEV transferred to the shadow register. The update event is generated when the counter reaches upper overflow and when the UDIS bit in the TIMx_CR1 register is equal to zero. Update events can also be generated by software.

The counter is driven by the clock output of the prescaler, CK_CNT, which is valid only when the counter enable bit (CEN) in the counter TIM14_CR1 register is set.

Note that the counter starts counting one clock cycle after the CEN bit of the TIMx_CR register is set.

Prescaler Description:

The prescaler divides the counter clock by any value between 1 and 65536. It is a 16-bit counter based on 16-bit register control (in the TIMx_PSC register). Because this control register has a buffer, it can be changed at runtime. The parameters of the new prescaler are adopted on the arrival of the next update event.

The following figure gives an example of changing the prescaler parameters while the counter is running.

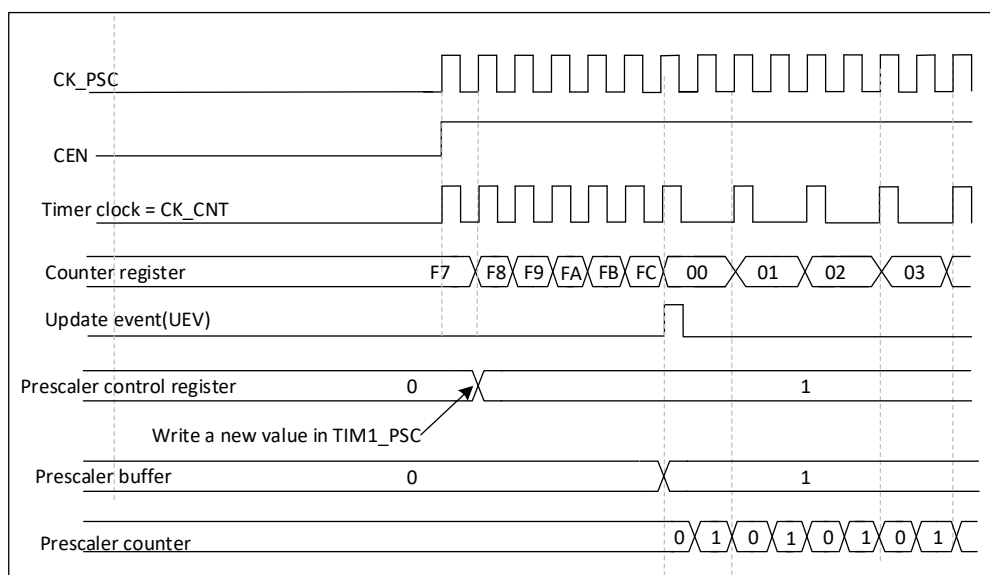


Figure 23-2 Timing diagram of the counter when the prescaler parameter is changed from 1 to 2

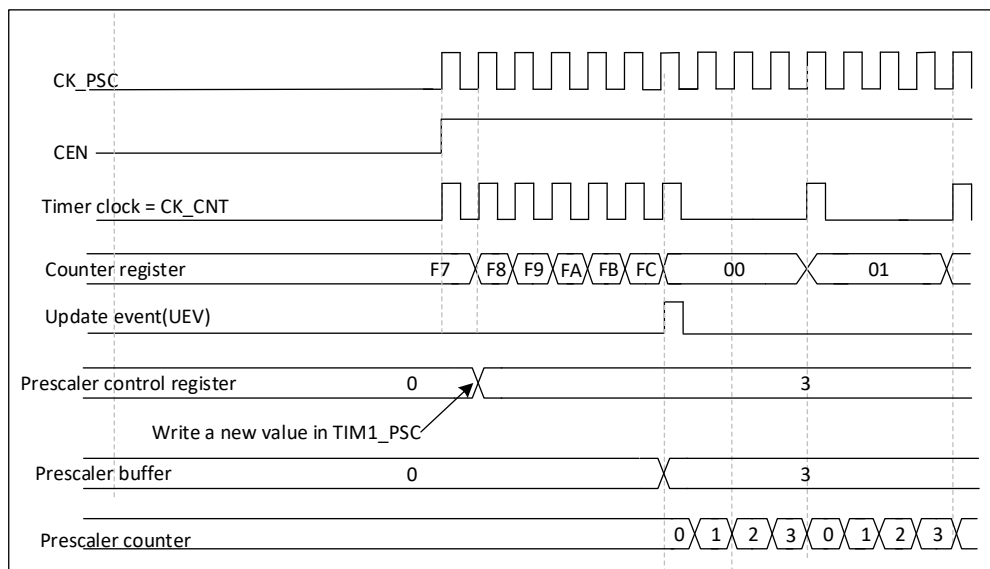


Figure 23-3 Timing diagram of the counter when the prescaler parameter is changed from 1 to 4

Up Count Mode

The counter counts from 0 up to the auto-reload value (the value registered by TIMx_ARR) and then restarts counting from 0 again and generates a counter overflow event.

Setting the UG bit in the TIMx_EGR register (via software) can similarly generate an update event.

Setting the UDIS bit in the TIMx_CR1 register disables the update event; this also prevents the shadow register from being updated when a new value is written to the preloaded register. No update event will be generated until the UDIS bit is cleared. In spite of this, the counter still starts from 0, and at the same time the count of the prescaler is cleared to 0 (but the value of the prescaler remains unchanged). In addition, if the URS bit (select update request) in the TIMx_CR1 register is set, setting the UG bit will generate an update event UEV, but the hardware does not set the UIF flag (i.e., no interrupt or DMA request is generated). This is to avoid generating both update and capture interrupts when clearing the counter in capture mode.

When an update event occurs, all registers are updated and the hardware simultaneously (based on the URS bit) sets the update flag bit (UIF bit in the TIMx_SR register).

- The auto-reload shadow register is reset to the value of the preload register (TIMx_ARR).
- The prescaler buffer is set to the value of the preload register (the contents of the TIMx_PSC register).

The following routines show several counter behaviors at different frequencies when TIMx_ARR=0X36.

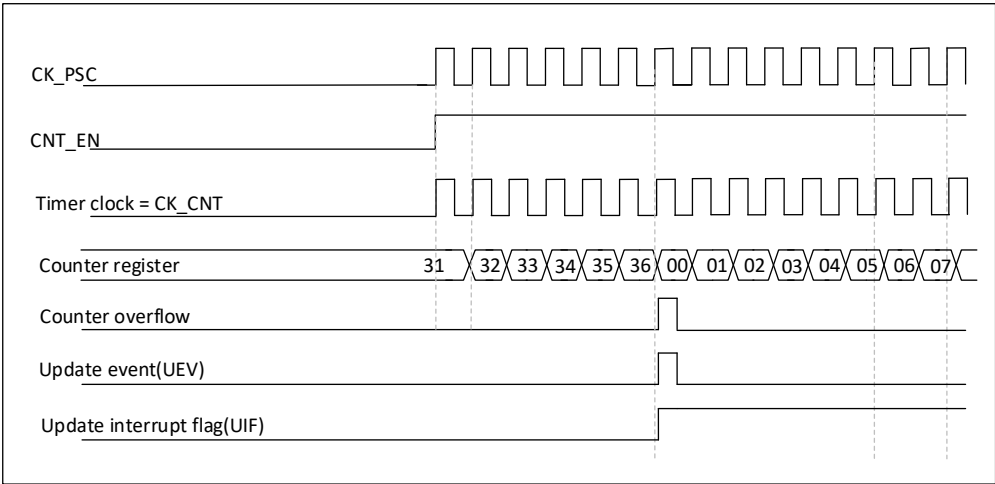


Figure 23-4 Timing diagram of the counter with internal clock division factor of 1

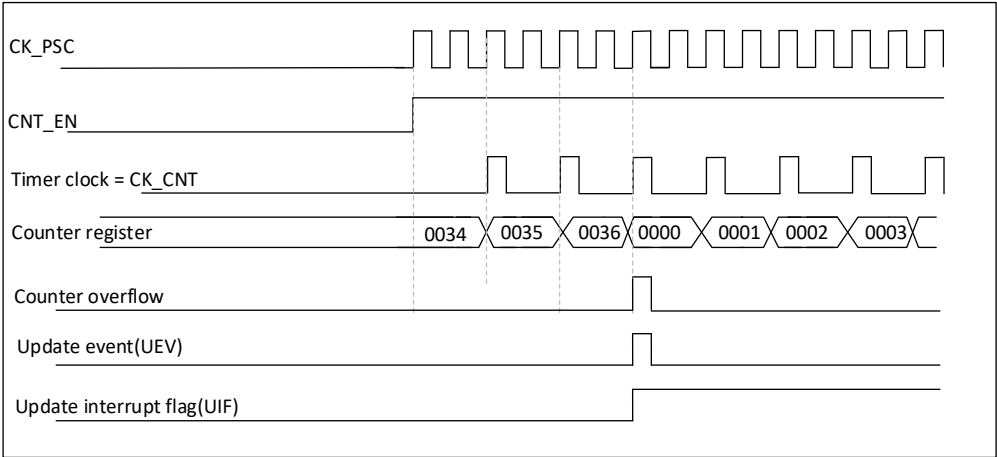


Figure 23-5 Timing diagram of the counter with internal clock divider factor of 2

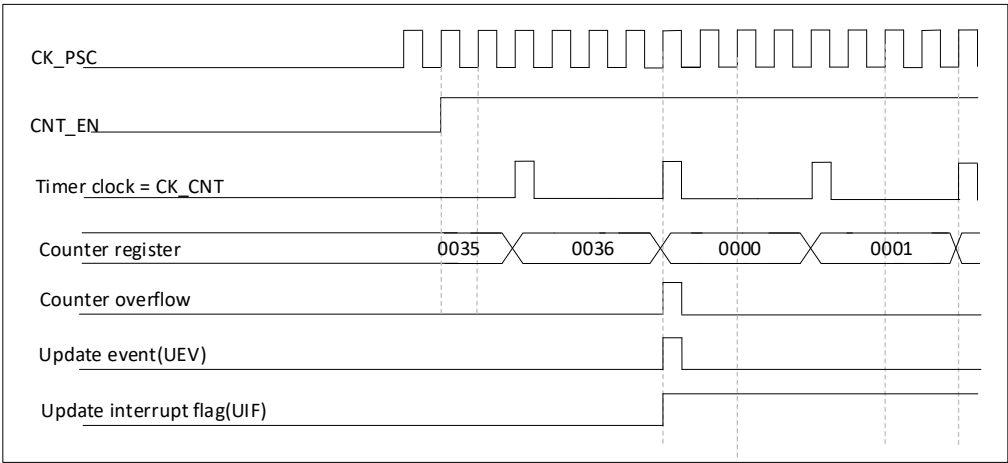


Figure 23-6 Timing diagram of the counter with internal clock divider factor of 4

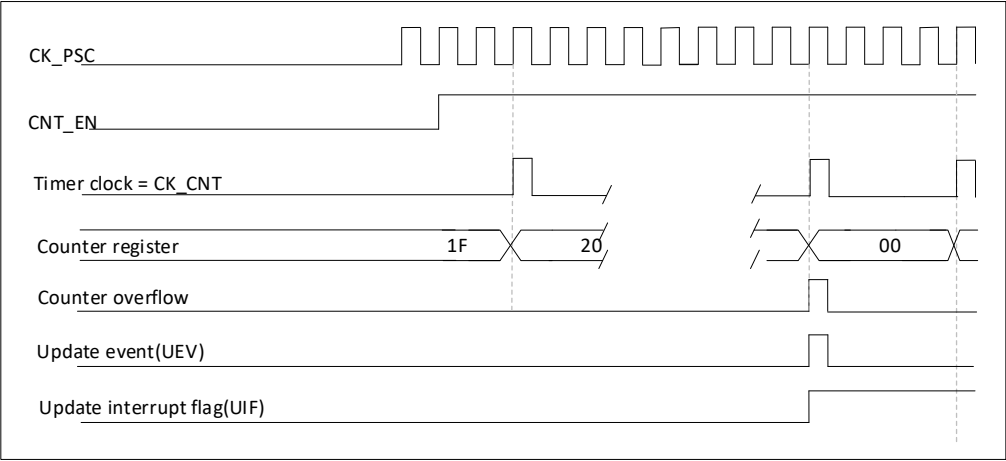


Figure 23-7 Timing diagram of the counter with internal clock dividing factor N

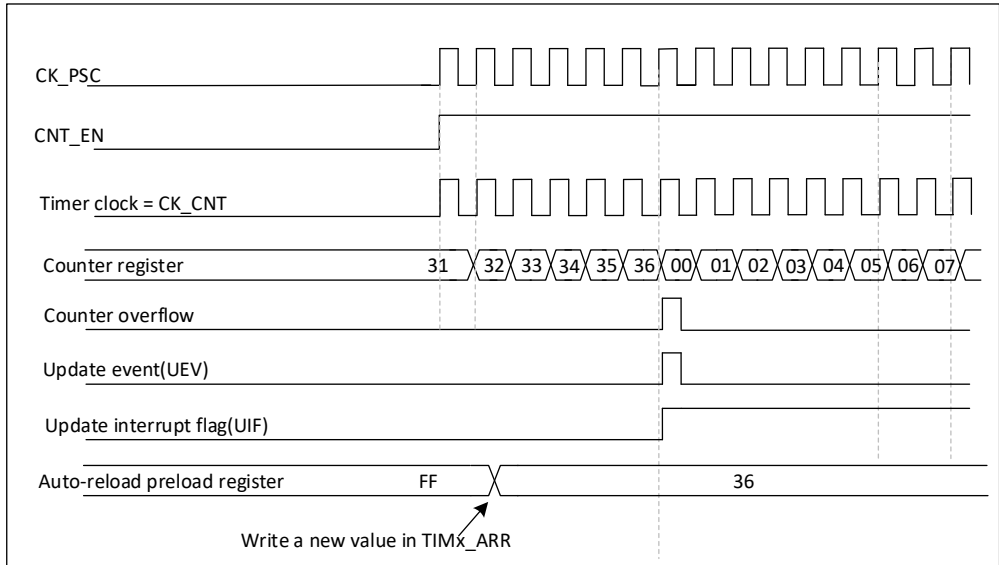


Figure 23-8 Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)

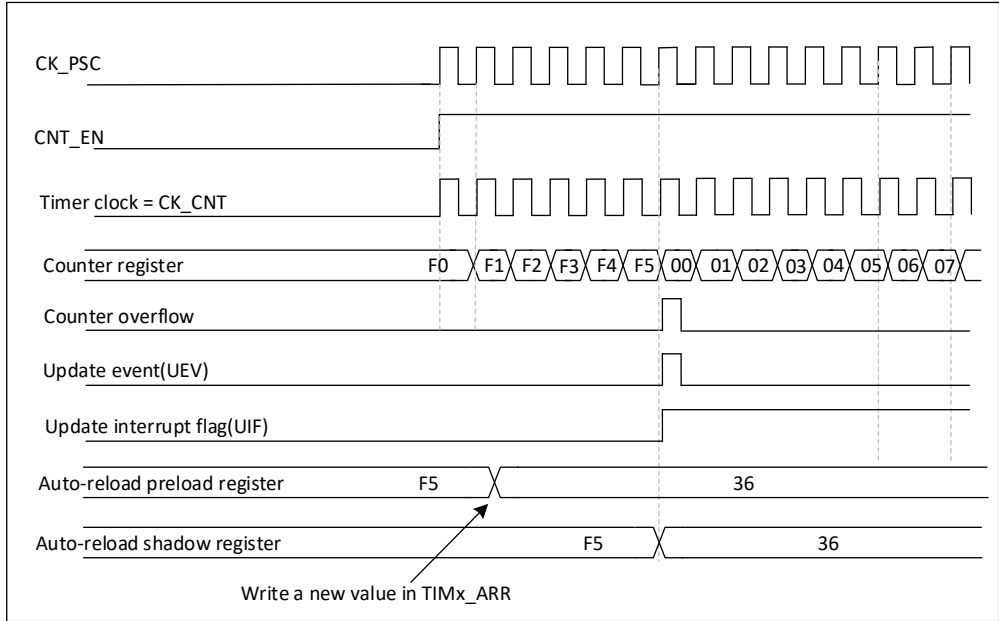


Figure 23-9 Counter timing diagram, update event when ARPE=1 (preloaded with TIMx_ARR)

23.3.2. Clock source

The counter is clocked by the internal clock (CK_INT). The CEN bit of the TIMx_CR1 register and the UG bit of the TIMx_EGR register are the actual control bits (except for the UG bit which is cleared automatically) and they can only be changed by software. Once the CEN bit is set to 1, the internal clock supplies the clock to the divider.

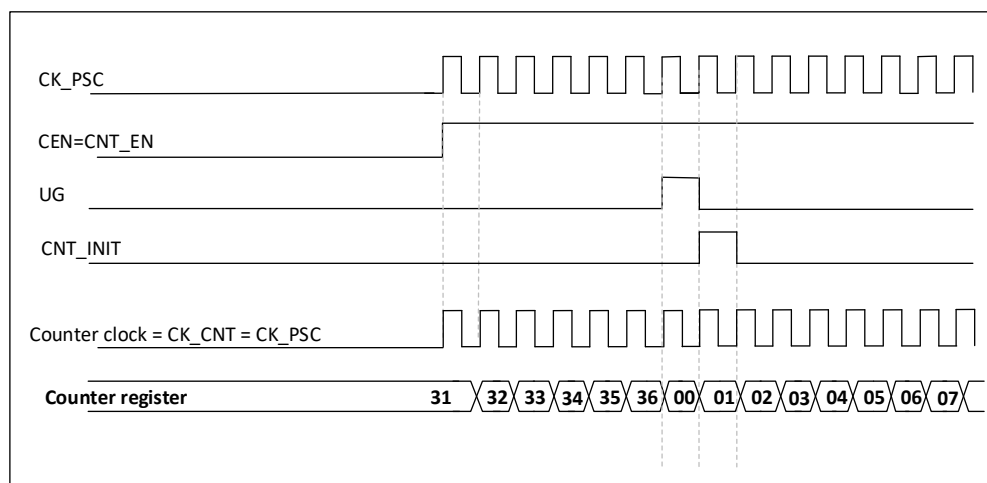


Figure 23-10 Control circuit in general mode with internal clock division factor of 1

23.3.3. Debug mode

When the chip enters the debug mode (M0+STOP), the TIMx counter either continues normal operation or stops depending on the DBG_TIMx_STOP setting in the DBG module.

23.4. TIM6 and TIM7 registers

23.4.1. TIM6 and TIM7 control register 1 (TIMx_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res		ARPE	Res	Res	Res	OPM	URS	UDIS	CEN
-	-	-	-	-	-	-		RW	-	-	-	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 8	Res	-	-	Res
7	ARPE	RW	0	Auto Reload Preload Allowed Bit 0: TIMx_ARR register is not buffered 1: TIMx_ARR register is buffered
6: 4	Res	-	-	Res
3	OPM	RW	0	Single pulse mode (One pulse mode) 0: The counter does not stop when an update event occurs; 1: The counter stops when the next update event (clearing the CEN bit) occurs.
2	URS	RW	0	Update request source The software selects the source of UEV events with this bit 0: If generating an update interrupt or DMA request is allowed, either of the following events generates an update interrupt or DMA request: - Counter overflow/underflow - Setting the UG position - Updates generated from the schema controller 1: If generating update interrupts or DMA requests is allowed, only counter overflow/underflow generates an update interrupt or DMA request
1	UDIS	RW	0	prohibit updating Software allows/disallows the generation of UEV events with this bit

Bit	Name	R/W	Reset Value	Function
				<p>0: UEV is allowed. Update (UEV) events are generated by any of the following events:</p> <ul style="list-style-type: none"> - Counter overflow/underflow - Setting the UG position - Updates generated from the schema controller <p>The cached registers are loaded with their preloaded values.</p> <p>1: UEV is prohibited. No update events are generated and the shadow registers (ARR,PSC,CCR_x) hold their values.</p> <p>If the UG bit is set or a hardware reset is issued from the mode controller, the counter and prescaler are reinitialized.</p>
0	CEN	RW	0	<p>Allowed counters</p> <p>0: Disable counter</p> <p>1: Turn on the counter</p> <p>Note: The external clock, gated mode and encoder mode will not work until the CEN bit is set in software. Trigger mode can be automatically set in hardware with the CEN bit.</p>

23.4.2. TIM6 and TIM7 Control Register 2 (TIM_x_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 7	Res	-	-	Res
6: 4	MMS [2:0]	RW	0	<p>Master Mode Selection</p> <p>These 3 bits are used to select the synchronization message (TRGO) to be sent to the slave timer in master mode. The possible combinations are as follows:</p> <p>000: Reset - The UG bit of the TIM_x_EGR register is used as a trigger output (TRGO). If the reset is generated by a trigger input (the slave mode controller is in reset mode),</p>

Bit	Name	R/W	Reset Value	Function
				<p>the signal on the TRGO is delayed relative to the actual reset.</p> <p>001: Enable - The counter enable signal CNT_EN is used as a trigger output (TRGO). Sometimes it is necessary to start multiple timers at the same time or to control the enabling of slave timers over a period of time. The counter enable signal is generated by a logical or of the CEN control bits and the trigger input signal in gated mode. When the counter enable signal is controlled on the trigger input, there is a delay on TRGO unless master/slave mode is selected (see description of the MSM bit in the TIMx_SMCR register).</p> <p>010: Update - The update event is selected as a trigger input (TRGO). For example, a master timer clock may be used as a prescaler for a slave timer.</p> <p>NOTE: The slave timer and ADC clocks must first be enabled to receive the master timer signal and not be changed on reception.</p>
3: 0	Res	-	-	Res

23.4.3. TIM6 and TIM7 DMA/Interrupt Enable Registers (TIM14_DIER)

Address offset: 0x0C

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 9	Res	-	-	Res
8	UDE	RW	0	<p>UDE: DMA request to allow updates</p> <p>0: DMA requests for updates are disabled</p> <p>1: Allow updated DMA requests</p>
7: 1	Res	-	-	Res
0	UIE	RW	0	<p>UIE: Allow update interruptions</p> <p>0: Disable update interrupt</p> <p>1: Allow updates to be interrupted</p>

23.4.4. TIM6 and TIM7 Status Registers (TIMx_SR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	UIF
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	RC_W0

Bit	Name	R/W	Reset Value	Function
31: 1	Res	-	-	Res
0	UIF	RC_W0	0	<p>Update Interrupt Flag, this bit is set to 1 by hardware when an update event is generated. It is cleared 0 by the software.</p> <p>0: No update events are generated;</p> <p>1: Update event waiting for response. This bit is set to 1 by hardware when the register is updated:</p> <ul style="list-style-type: none"> - If UDIS=0 in the TIMx_CR1 register, an update event overflow is generated; - If UDIS=0 and URS=0 in the TIMx_CR1 register, the update event is generated when UG=1 in the TIMx_EGR register. <p>pieces (software reinitialization of the CNT);</p>

23.4.5. TIM6 and TIM7 Event Generation Registers (TIMx_EGR)

Address offset: 0x14

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 1	Res	-	-	Res

0	UG	W	0	<p>An update event is generated and this bit is set to 1 by software and cleared to 0 automatically by hardware.</p> <p>0: No action; 1: Reinitialize the counter and generate a register update event. Note that the prescaler counter is also cleared to 0 (but the prescaler coefficients remain unchanged).</p>
---	----	---	---	---

23.4.6. TIM6 and TIM7 counters (TIMx_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 16	Res	-	-	Res
15: 0	CNT [15:0]	RW	0	Counter value

23.4.7. TIM6 and TIM7 prescalers (TIMx_PSC)

Address offset: 0x28

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 16	Res	-	-	Res
15: 0	PSC [15:0]	RW	0	<p>Prescaler value</p> <p>The counter clock frequency (CK_CNT) is equal to $f_{CK_PSC}/(PSC[15:0]+1)$.</p> <p>The PSC contains the value loaded into the current prescaler register when the update event is generated; the update event includes the counter</p>

				Cleared to 0 by the UG bit of TIM_EGR or by a slave controller operating in reset mode.
--	--	--	--	---

23.4.8. TIM6 and TIM7 Automatic Reload Registers (TIMx_ARR)

Address offset: 0x2C

Reset value: 0x0000 FFFF

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

Bit	Name	R/W	Reset Value	Function
31: 16	Res	-	-	Res
15: 0	ARR [15:0]	RW	FFFF	<p>Auto-reload values</p> <p>The ARR contains the value that will be loaded into the actual auto-reload register.</p> <p>Refer to 23.3.1: Updates and actions of the time base unit regarding ARR for details.</p> <p>The counter does not work when the value of Auto-Reload is empty.</p>

24. General purpose timer (TIM14)

24.1. TIM14 Introduction

The general-purpose timer TIM14 consists of a 16-bit auto-reload counter driven by a programmable prescaler.

It is suitable for a variety of applications, including measuring the pulse length of an input signal (input capture) or generating an output waveform (output comparison and PWM).

Using the timer prescaler and the RCC clock controller prescaler, the pulse length and waveform period can be adjusted from a few microseconds to a few milliseconds.

The general purpose timers (TIMx) are all completely independent and do not share any resources with each other. They can be operated synchronously together, see the synchronization chapter of TIM2/3.

24.2. TIM14 main features

- 16-bit auto-reload up counter
- 16-bit programmable (can be modified in real time) prescaler, counter clock frequency crossover coefficient of any value between 1 and 65536
- 1 independent channel as:
 - Input Capture
 - Output Comparison
 - PWM generation (edge-aligned mode)

An interrupt is generated when the following events occur

- Update: Counter overflow upwards, counter initialization (via software)
- Input Capture
- Output Comparison



24.3.1. Time base unit (in computing)

The software can read and write counters, auto-reload registers, and prescaler registers, and operate them even when the counters are running.

- Counter register (TIM14_CNT)
- Prescaler register (TIM14_PSC)
- Automatic Reload Register (TIM14_ARR)

The counter is driven by the clock output of the prescaler, CK_CNT, which is valid only when the counter enable bit (CEN) in the counter TIM14_CR1 register is set.

441/755

Prescaler Description:

The prescaler divides the counter clock by any value between 1 and 65536. It is a 16-bit counter based on the 16-bit register control (in the TIM14_PSC register). Because this control register has a buffer, it can be changed at runtime. The parameters of the new prescaler are adopted on the arrival of the next update event.

The following figure gives an example of changing the prescaler parameters while the counter is running.

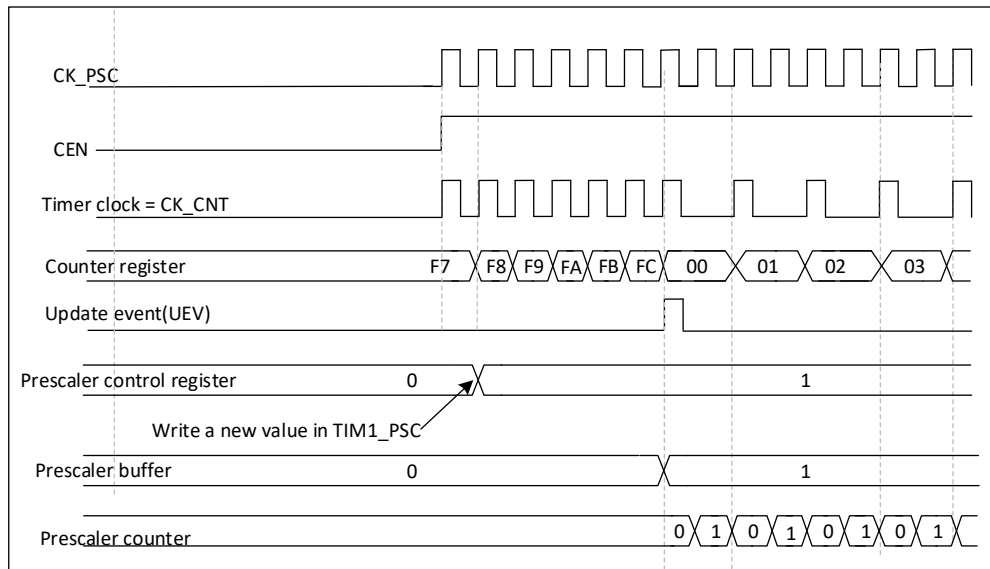


Figure 24-2 Timing diagram of the counter when the prescaler parameter is changed from 1 to 2

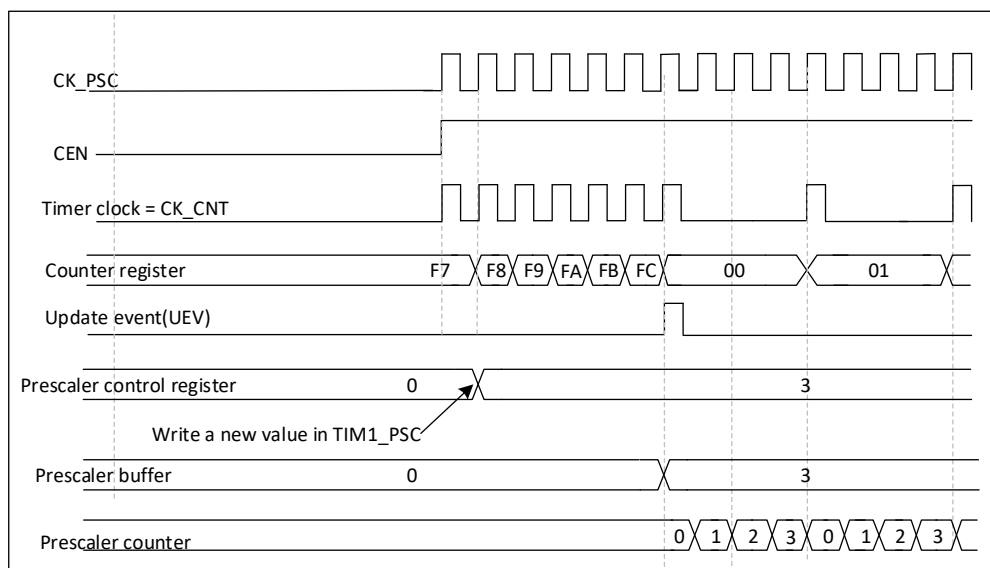


Figure 24-3 Timing diagram of the counter when the prescaler parameter is changed from 1 to 4

Up Count Mode

The counter counts from 0 up to the auto-reload value (the value registered by TIM14_ARR) and then restarts counting from 0 again and generates a counter overflow event.

Setting the UG bit in the TIM14_EGR register (via software) can similarly generate an update event.

Setting the UDIS bit in the TIM14_CR1 register disables the update event; this also prevents the shadow register from being updated when a new value is written to the preloaded register. No update event will be generated until the UDIS bit is cleared. In spite of this, the counter still starts from 0, and at the same time the count of the prescaler is cleared to 0 (but the value of the prescaler remains unchanged). In addition, if the URS bit (select update request) in the TIM14_CR1 register is set, setting the UG bit will generate an update event UEV, but the hardware does not set the UIF flag (i.e., no interrupt or DMA request is generated). This is to avoid generating both update and capture interrupts when clearing the counter in capture mode.

When an update event occurs, all registers are updated and the hardware simultaneously (based on the URS bit) sets the update flag bit (UIF bit in the TIM14_SR register).

- The auto-reload shadow register is reset to the value of the preload register (TIM14_ARR).
- The prescaler buffer is set to the value of the preload register (the contents of the TIM14_PSC register).

The following routines show several counter behaviors at different frequencies when TIMx_ARR=0X36.

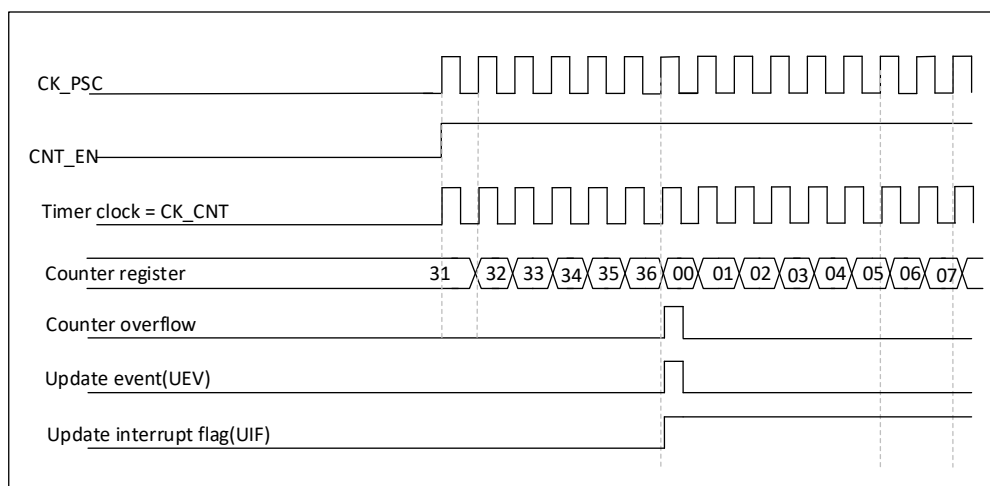


Figure 24-4 Timing diagram of the counter with internal clock division factor of 1

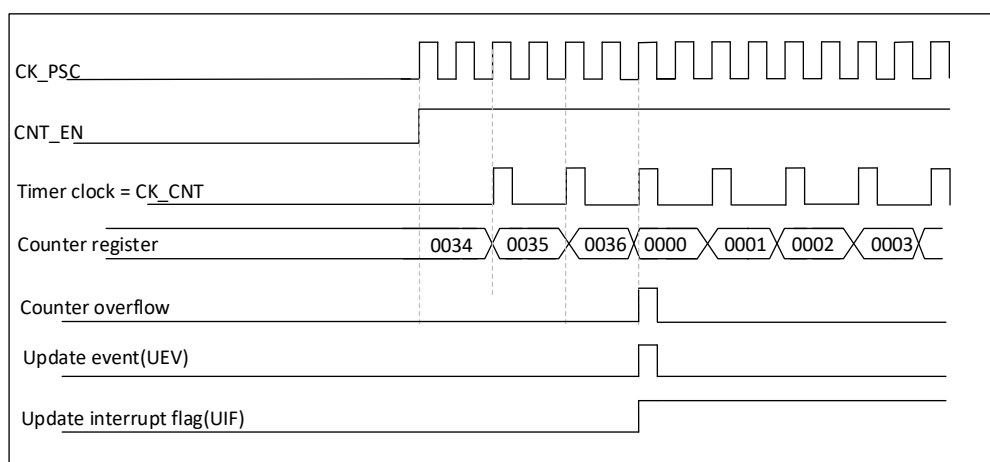


Figure 24-5 Timing diagram of the counter with internal clock divider factor of 2

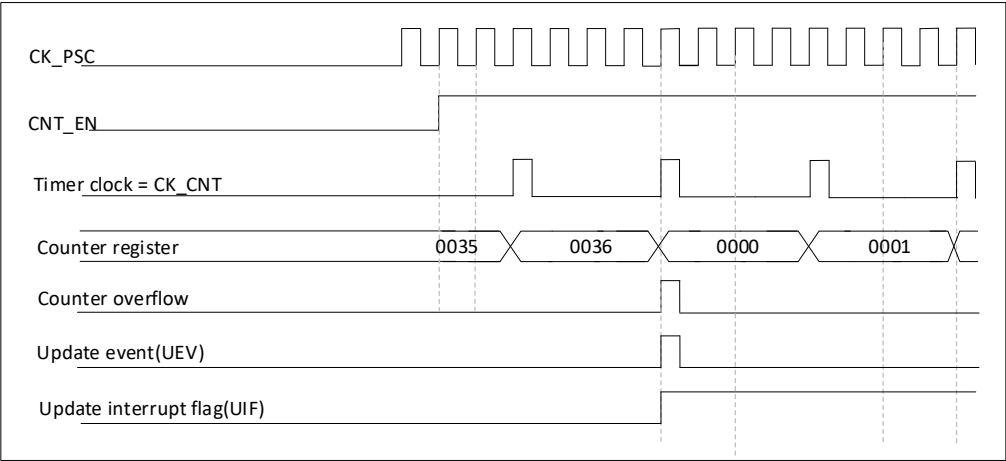


Figure 24-6 Timing diagram of the counter with internal clock divider factor of 4

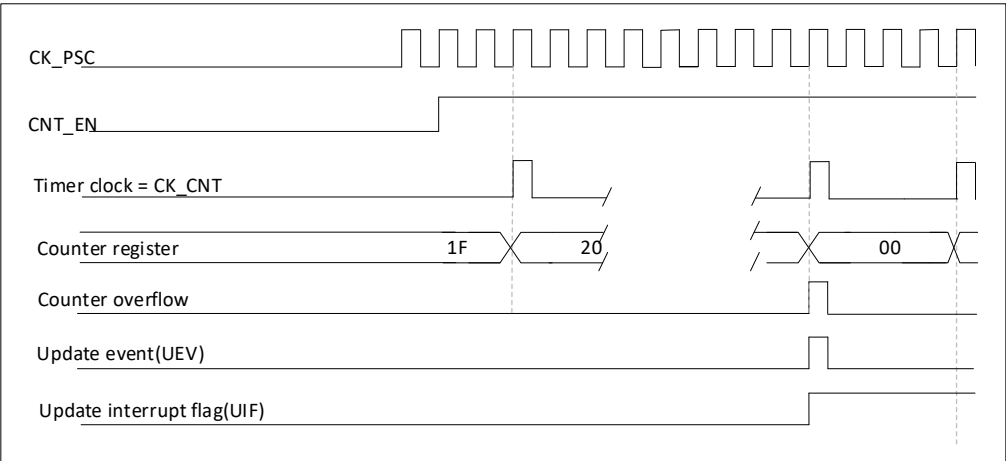


Figure 24-7 Timing diagram of the counter with internal clock dividing factor N

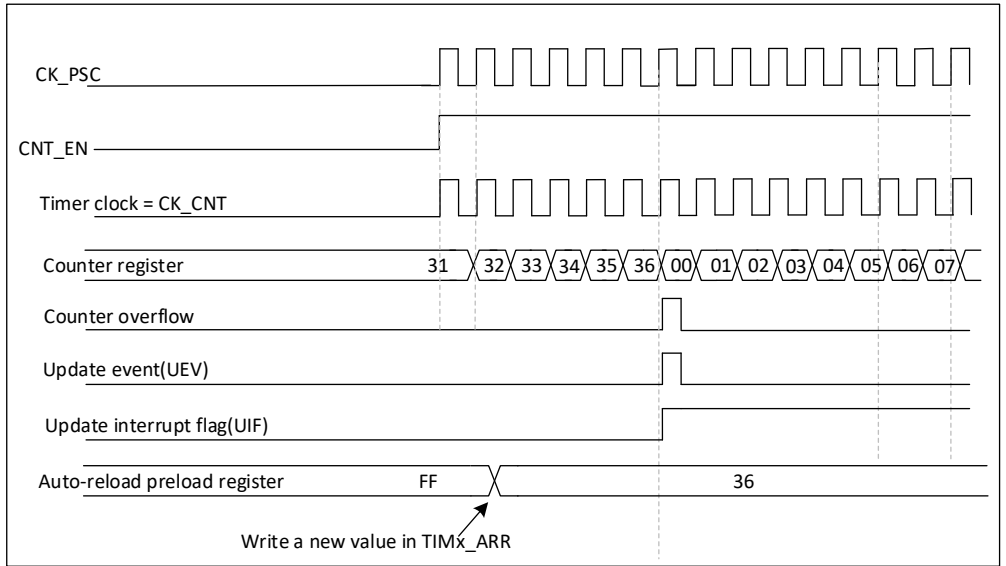


Figure 24-8 Counter timing diagram, update event when ARPE=0 (TIMx_ARR not preloaded)

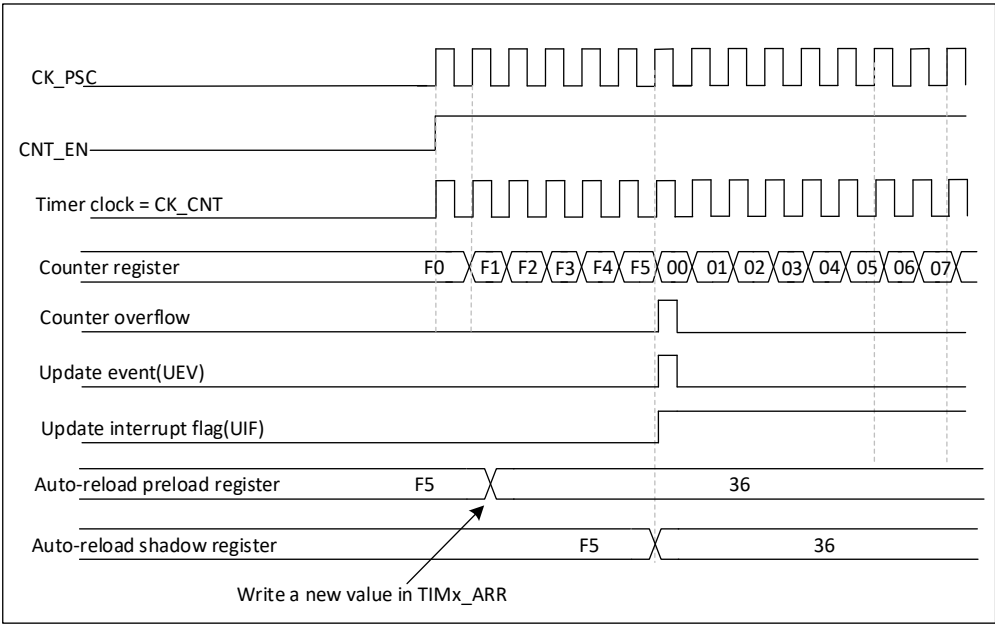


Figure 24-9 Counter timing diagram, update event when ARPE=1 (preloaded with TIMx_ARR)

24.3.2. Clock source

The counter is clocked by the internal clock (CK_INT). The CEN bit of the TIMx_CR1 register and the UG bit of the TIM14_EGR register are the actual control bits (except for the UG bit which is cleared automatically) and they can only be changed by software. Once the CEN bit is set to 1, the internal clock supplies the clock to the divider.

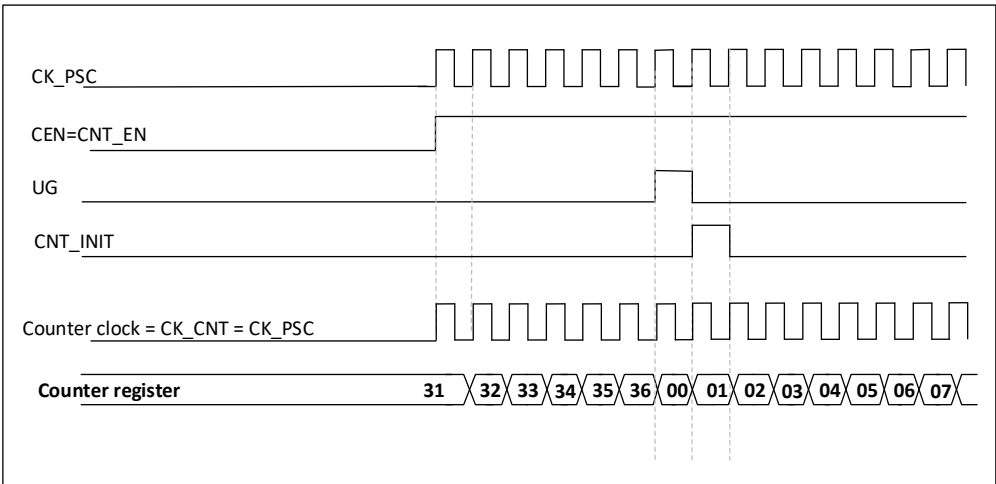


Figure 24-10 Control circuit in general mode with internal clock division factor of 1

24.3.3. Capture/Compare Channel

Each capture/compare channel is organized around a capture/compare register (containing shadow registers), including the input portion of the capture (digital filtering, multiplexing, and prescaler), and the output portion (comparator and output control).

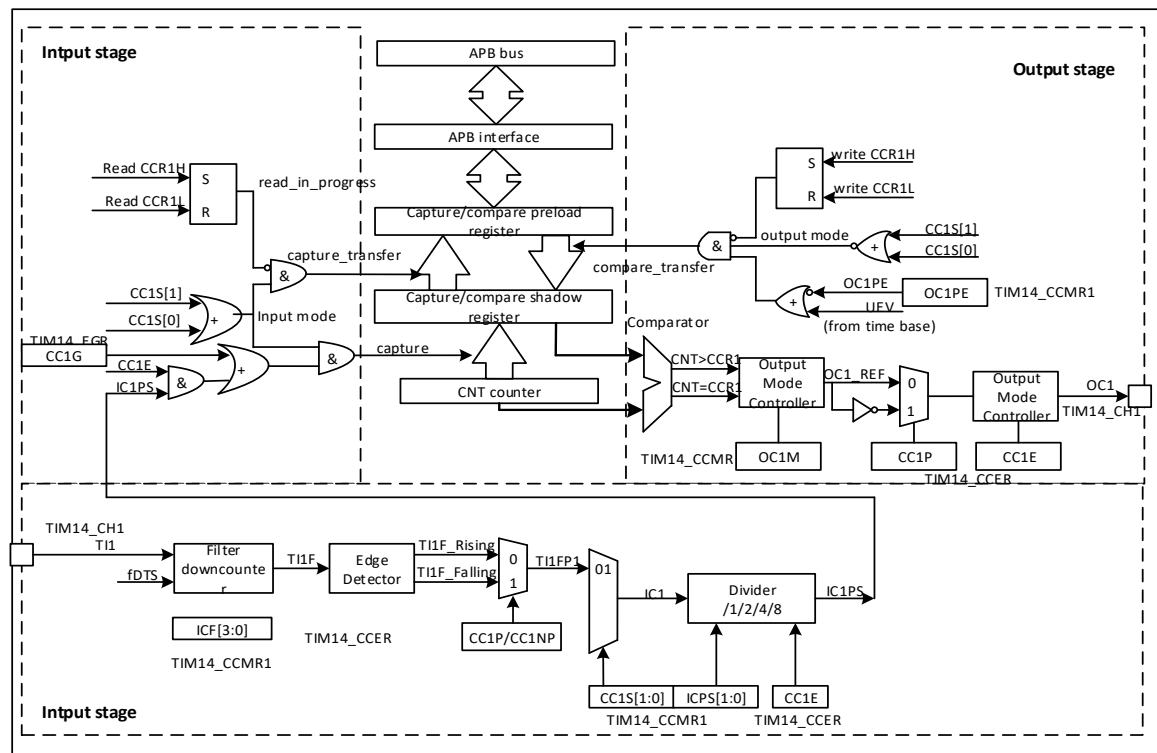


Figure 24-11 TIM14 Capture/Compare Channel Diagram

The input section samples the corresponding TIx input signal and produces a filtered signal, TIxF. An edge detector with polarity selection then generates a signal (TIxFPx) which can be used as an input trigger from the pattern controller or as a capture control. This signal enters the capture register (ICxPS) through pre-divided frequency.

The output section produces an intermediate waveform (highly active) as a reference, and the end of the chain determines the polarity of the final output signal.

The capture/compare module consists of a preloaded register and a shadow register. The read/write process operates only on the preloaded registers.

In capture mode, the capture occurs on the shadow registers, which are then copied to the preloaded registers.

In comparison mode, the contents of the preloaded registers are copied to the shadow registers, and then the contents of the shadow registers are compared to the counter.

24.3.4. Input Capture Mode

In input capture mode, the current value of the counter is latched into the capture/compare register (TIMx_CCRx) when the corresponding edge of the ICx signal is detected. When a capture event occurs, the corresponding CCxIF flag (TIM14_SR register) is set to 1. If the CCxIF flag is already high when the capture event occurs, the repeat capture flag CCxOF (TIMx_SR register) is set to one. Writing CCxIF clears CCxIF, or reading capture data from storage also clears CCxIF. Writing CCxOF=0 clears CCxOF.

The following example shows how to capture the value of the counter into the TIM14_CCR1 register on the rising edge of the TI1 input as follows:

- Select valid outputs: TIM14_CCR1 must be connected to the TI1 input, so write CC1S=01 to the TIM14_CMR1 register, as long as the channel is configured as an input when CC1S is not 00, and the TIM14_CCR1 register becomes read-only.
- Configure the input filter to the desired width based on the characteristics of the input signal (i.e., when the input is TIx, the input filter control bit is the ICxF bit in the TIM14_CMRx register). Assuming that the input signal is dithered for a maximum of 5 internal clock cycles, we must configure the width of the filter to be longer than 5 clock cycles. Therefore, we can (at fDTS frequency) sample 8 times in a row, have confirmed the edge change on TI1 once really, and then write IC1F=0011 in the TIM14_CMR1 register.
- Select the active conversion edge of the TI1 channel by writing CC1P=0 (rising edge) (and CC1NP=0) in the TIMx_CCER register
- Configure the input prescaler. In this example, we want the capture to occur at every valid level-transition moment, so the prescaler is disabled (write IC1PS=00 to the TIMx_CMR1 register).
- Setting CC1E=1 in the TIMx_CCER register allows the value of the capture counter to be captured into the capture register.

If desired, allow the associated interrupt request by setting the CC1E bit in the TIMx_DIER register

When an input capture occurs:

- When a valid level transition is generated, the counter value is transferred to the TIM14_CCR1 register.
- The CC1IF flag is set (interrupt flag). When at least 2 consecutive captures occur without CC1IF having been cleared, CC1OF is also set to 1.
- If the CC1IE bit is set, an interrupt request is generated.

In order to handle overcapture, it is recommended that data be read before the overcapture flag is set, this is to avoid losing capture overflow information that may be generated after the capture overflow flag is read and before the data is read.

Note: Input capture interrupt requests can be generated by software setting the corresponding CCxG bit in TIMx_EGR.

24.3.5. Forced Output Mode

In this mode (CCxS bits = 00 in the TIM14_CMRx register), the output comparison signals (OCxREF and the corresponding OCx) can be set to valid or invalid states directly by software, independent of the result of the comparison between the Output Comparison Register and the counter.

The output compare signal (OCxREF/OCx) can be forced to be active by writing the corresponding OCxM=101 in the TIM14_CMRx register. In this way OCxREF is forced high (OCxREF is always active high), while OCx gets the value of CCxP with the opposite polarity bit.

For example, if CCxP=0 (OCx active high), then OCx is forced high.

Setting OCxM=100 in the TIM14_CMRx register forces the OCxREF signal low.

In this mode, the comparison between the TIM14_CCRx shadow register and the counter is still in progress and the corresponding flags are modified. This will be covered in the Output Comparison Mode section below.

24.3.6. Output Comparison Mode

This function is used to control an output waveform or to indicate that a given period of time has elapsed.

When the counter and the capture/compare registers have the same contents, the output compare job does the following:

- Outputs the values defined by the output comparison mode (OCxM bit in the TIM14_CCMRx register) and output polarity (CCxP bit in the TIMx_CCER register) to the corresponding pins. When comparing matches, the output pin can hold its level (OCxM=000), be set to an active level (OCxM=001), be set to an inactive level (OCxM=010), or be flipped (OCxM=011).
- Set the flag bit in the interrupt status register (CcxIF bit in the TIMx_SR register).
- An interrupt is generated if the corresponding interrupt mask (CCxIE bit in the TIM14_DIER register) is set.

Whether or not the TIM14_CCRx registers require the use of preloaded registers can be selected by configuring the OCxPE bit in TIM14_CMRx.

In output comparison mode, the update event UEV has no effect on the OCxREF and OCx outputs. The synchronization can be accurate up to one counting cycle of the counter. Output compare mode (in single pulse mode) can also be used to output a single pulse.

The TIMx_CCRx registers can be updated by software at any time to control the output waveform, provided that the preloaded registers are not used (OCxPE='0', otherwise the shadow registers of TIMx_CCRx can only be updated when the next update event occurs). An example is given in the figure below.

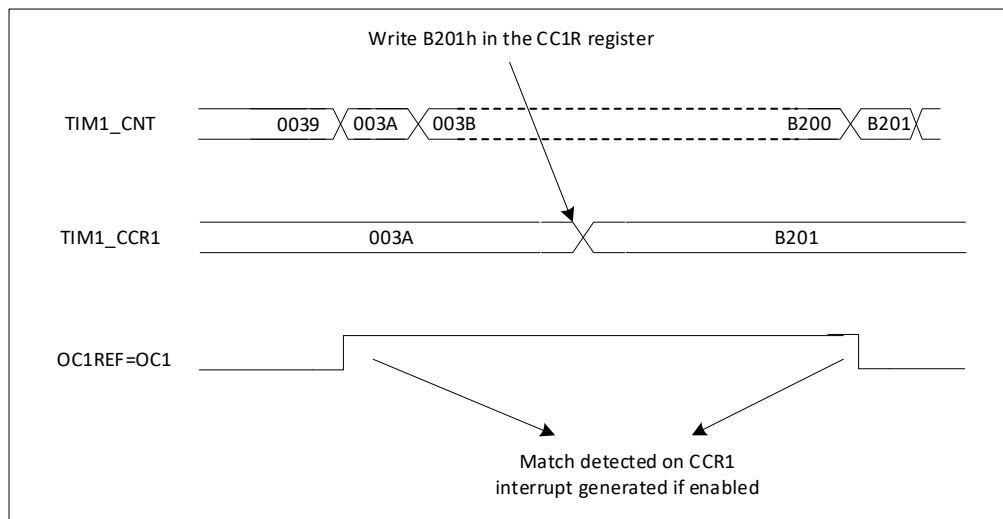


Figure 24-12 Output Compare Mode, Flip OC1

24.3.7. PWM mode

The Pulse Width Adjustment mode generates a signal with a frequency determined by the TIMx_ARR register and a duty cycle determined by the TIMx_CCRx register.

Writing "110" (PWM mode 1) or "111" (PWM mode 2) to the OCxM bit in the TIM14_CMRx register can independently set each OCx output channel to generate one PWM. The TIM14_CCMRx register OCxPE bit must be set to enable the corresponding preload register, and finally the ARPE bit in the TIM14_CR1 register (in count-up or center-symmetric modes) must be set to enable the auto-reload preload register.

Preloaded registers are transferred to shadow registers only when an update event occurs, so all registers must be initialized by setting the UG bit in the TIM14_EGR register before the counter begins counting.

The polarity of the OCx can be set by software in the CCxP bit in the TIM14_CCER register, which can be set to active high or active low. The CcxE bit in the TIM14_CCER register controls the OCx output enable.

In PWM mode (Mode 1 or Mode 2), TIM14_CNT and TIM14_CCRx are always being compared for compliance with $TIM14_CNT \leq TIM14_CCRx$.

The timer is capable of generating PWM signals in edge-aligned mode.

PWM Edge Alignment Mode

Below is an example of PWM mode 1. The PWM reference signal OCxREF is high when $TIM14_CNT < TIMx_CCRx$ and low otherwise. If the comparison value in TIM14_CCRx is greater than the auto-reload value (TIM14_ARR), OCxREF is held to '1'. If the comparison value is 0, OCxREF remains '0'.

The following figure shows an example of an edge-aligned PWM waveform with $TIMx_ARR=8$.

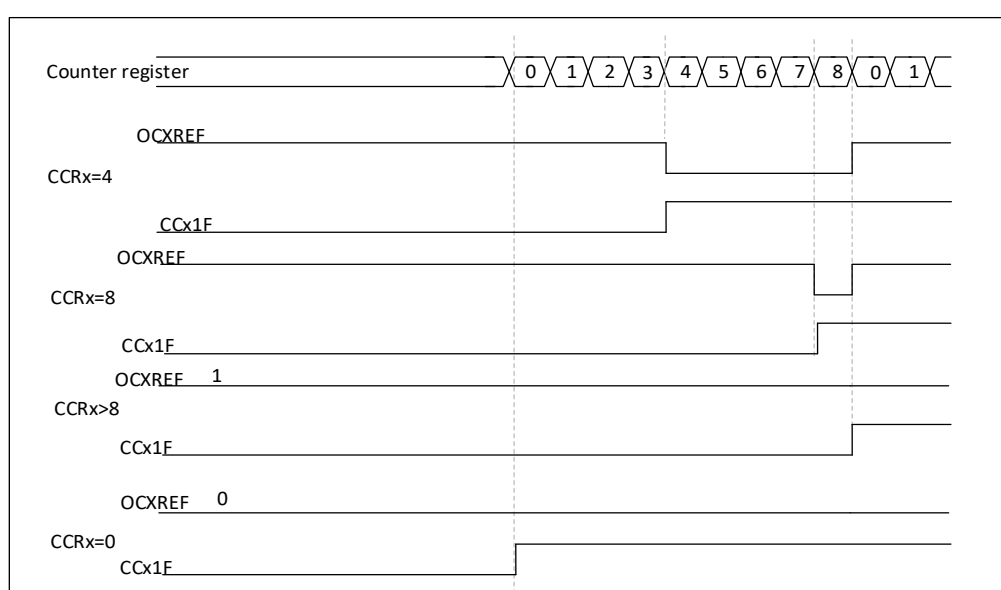


Figure 24-13 Edge-aligned PWM waveform (ARR=8)

24.3.8. Single Pulse Mode

The single pulse mode (OPM) is a special case of one of the many aforementioned modes. This mode allows the counter to respond to an excitation and generate a pulse with a programmable pulse width after a programmable delay.

The counter can be activated from the mode controller to generate waveforms in output comparison mode or PWM mode. Setting the OPM bit in the TIMx_CR1 register will select single-pulse mode, which allows the counter to automatically stop when the next update event, UEV, is generated.

A pulse is generated only when the comparison value is different from the initial value of the counter. Before starting (when the timer is waiting to be triggered), it must be configured as follows:

Upward counting mode: counter $CNT < CCRx \leq ARR$ (in particular, $0 < CCRx$).

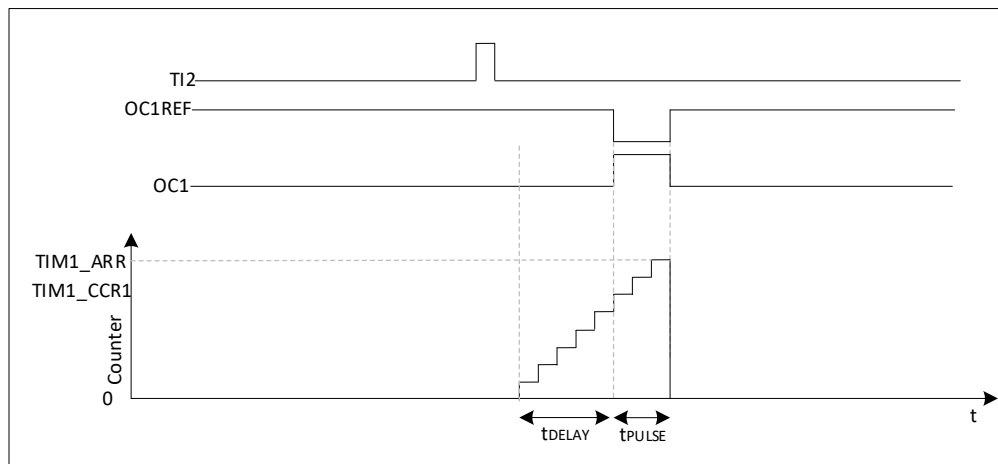


Figure 24-14 Example of single-pulse mode

For example, you need to generate a positive pulse of length t_{PULSE} on OC1 after a delay of t_{DELAY} starting from the detection of a rising edge on the TI2 input pin.

TI2FP2 is assumed as the trigger:

- Set $CC2S=01$ in the $TIMx_CMR1$ register to map TI2FP2 to TI2.
- Set $CC2P=0$ in the $TIMx_CCER$ register to enable the TI2FP2 to detect the rising edge.
- Setting $TS=110$ in the $TIMx_SMCR$ register triggers the TI2FP2 as a slave mode controller (TRGI).
- Setting $SMS=110$ (trigger mode) in the $TIMx_SMCR$ register, the TI2FP2 is used to start the counter.

The OPM waveform is determined by the value written to the comparison register (taking into account the clock frequency and counter prescaler)

- t_{DELAY} is defined by the value in the $TIMx_CCR1$ register.
- t_{PULSE} is defined by the difference between the auto-reload value and the comparison value ($TIMx_ARR - TIMx_CCR1$).
- Assuming that a waveform from 0 to 1 is to be generated when a comparison match occurs, and a waveform from 1 to 0 is to be generated when the counter reaches the preloaded value; firstly, set $OC1M=111$ in $TIMx_CMR1$ register to enter PWM mode 2; selectively enable the preload registers as needed: set $OC1PE=1$ in $TIMx_CMR1$ and $ARPE$ in $TIMx_CR1$ register; then fill the comparison value in $TIMx_CCR1$ register, and set the UG bit to generate an update event. then fill in the comparison value in the $TIMx_CCR1$ register and the auto-reload value in the $TIMx_ARR$ register, set the UG bit to generate an update event, and wait for an external trigger event on TI2. In this example, $CC1P = 0$.

In this example, the DIR and CMS bits in the $TIMx_CR1$ register should be set low.

Since only one pulse is needed, $OPM=1$ in the $TIMx_CR1$ register must be set to stop counting at the next update event (when the counter flips from the auto-reload value to 0). When $OPM=0$, repeat mode is selected.

24.3.9. Timer synchronization

All TIMx timers are connected internally for timer synchronization or linking. When a timer is in master mode, it can perform operations such as resetting, starting, stopping, or providing a clock to the counter of another timer that is in slave mode.

24.3.10. Debug mode

When the chip enters the debug mode (M0+STOP), the TIMx counter either continues normal operation or stops depending on the DBG_TIMx_STOP setting in the DBG module.

24.4. TIM14 Register

24.4.1. TIM14 Control Register 1 (TIM14_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	CKD [1:0]		ARPE	Res	Res	Res	OPM	URS	UDIS	CEN
-	-	-	-	-	-	RW		RW	-	-	-	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 10	Res	-	-	Res
9: 8	CKD [1:0]	RW	00	Clock division factor, these 2 bits define the division ratio between the frequency of the timer clock (CK_INT), the sampling clock used 00: tDTS = tCK_INT01: tDTS = 2 x tCK_INT10: tDTS = 4 x tCK_INT11: Reserved, do not use this configuration!
7	ARPE	RW	0	Auto Reload Preload Allowed Bit 0: TIM14_ARR register is not buffered 1: TIM14_ARR register is buffered
6: 4	Res	-	-	Res
3	OPM	RW	0	Single pulse mode (One pulse mode) 0: The counter does not stop when an update event occurs; 1: The counter stops when the next update event (clearing the CEN bit) occurs.
2	URS	RW	0	Update request source The software selects the source of UEV events with this bit 0: If generating an update interrupt or DMA request is allowed, either of the following events generates an update interrupt or DMA request: - Counter overflow/underflow - Setting the UG position

Bit	Name	R/W	Reset Value	Function
				1: If generating update interrupts or DMA requests is allowed, only counter overflow/underflow generates an update interrupt or DMA request
1	UDIS	RW	0	<p>prohibit updating</p> <p>Software allows/disallows the generation of UEV events with this bit</p> <p>0: UEV is allowed. Update (UEV) events are generated by any of the following events:</p> <ul style="list-style-type: none"> - Counter overflow/underflow - Setting the UG position <p>The cached registers are loaded with their preloaded values.</p> <p>1: UEV is prohibited. No update events are generated and the shadow registers (ARR,PSC,CCR_x) hold their values.</p> <p>If the UG bit is set or a hardware reset is issued from the mode controller, the counters and prescalers are Reinitialization.</p>
0	CEN	RW	0	<p>Allowed counters</p> <p>0: Disable counter</p> <p>1: Turn on the counter</p> <p>Note: The external clock, gated mode and encoder mode will not work until the CEN bit is set in software. Trigger mode can be automatically set in hardware with the CEN bit.</p>

24.4.2. TIM14 DMA/Interrupt Enable Register (TIM14_DIER)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res													CC1IE	UIE	
-													RW	RW	

Bit	Name	R/W	Reset Value	Function
31: 2	Res	-	-	Res

Bit	Name	R/W	Reset Value	Function
1	CC1IE	RW	0	CC1IE: Capture/Compare 1 interrupt allowed 0: Capture/compare 1 interrupt disabled 1: Allow capture/compare 1 interrupt
0	UIE	RW	0	UIE: Allow update interruptions 0: Disable update interrupt 1: Allow updates to be interrupted

24.4.3. TIM14 Status Register (TIM14_SR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res											IC1IF	Res	Res	Res	IC1IR
-											RC_W0	-	-	-	RC_W0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res						CC1OF	Res						CC1F	UIF	
-						RC_W0	-						RC_W0	RC_W0	

Bit	Name	R/W	Reset Value	Function
31: 21	Res	-	-	Res
20	IC1IF	RC_W0	0	Capture 1 flag on falling edge This flag can be set to 1 by hardware only if the corresponding channel is configured for input capture and the capture event is triggered by a falling edge. It is cleared '0' by software or by reading TIMx_CCR1. 0: No falling edge capture is generated; 1: A falling edge capture event occurs.
19: 17	Res	-	-	Res
16	IC1IR	RC_W0	0	Rising edge capture 1 flag This flag can be set to 1 by hardware only if the corresponding channel is configured for input capture and the capture event is triggered by a rising edge. It is cleared '0' by software or by reading TIMx_CCR1. 0: No rising edge capture is generated; 1: A rising edge capture event occurs.
15: 10	Res	-	-	Res
9	CC1OF	RC_W0	0	Capture/Compare1 Overcapture Marker

Bit	Name	R/W	Reset Value	Function
				<p>This flag can be set to 1 by hardware only if the corresponding channel is configured for input capture. Writing a 0 clears the bit.</p> <p>0: No overcapture is generated; 1: When CC1IF is set to 1, the counter value has been captured into the TIM14_CCR1 register.</p>
8: 2	Res	-	-	Res
1	CC1IF	RC_W0	0	<p>Capture/Compare1 Interrupt Marker</p> <p>If channel CC1 is configured for output mode:</p> <p>This bit is set to 1 by hardware when the counter value matches the comparison value one clock cycle later, and it is cleared to 0 by software.</p> <p>0: No match occurred; 1: The value of TIM14_CNT matches the value of TIM14_CCR1.</p> <p>If channel CC1 is configured for input mode:</p> <p>This bit is set to 1 by hardware when a capture event occurs, and it is cleared to 0 by software or by reading TIM14_CCR1.</p> <p>0: No input capture is generated; 1: Input capture is generated and the counter value has been loaded into TIM14_CCR1 (an edge of the same polarity as selected is detected on IC1).</p>
0	UIF	RC_W0	0	<p>Update Interrupt Flag, this bit is set to 1 by hardware when an update event is generated. It is cleared 0 by the software.</p> <p>0: No update events are generated; 1: Update event waiting for response. This bit is set to 1 by hardware when the register is updated:</p> <ul style="list-style-type: none"> - If UDIS=0 in the TIMx_CR1 register, an update event overflow is generated; - If UDIS=0 and URS=0 in the TIMx_CR1 register, the update event is generated when UG=1 in the TIMx_EGR register. <p>pieces (software reinitialization of the CNT);</p>

24.4.4. TIM14 Event Generation Register (TIM14_EGR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res														CC1G	UG
-														W	W

Bit	Name	R/W	Reset Value	Function
31: 2	Res	-	-	Res
1	CC1G	W	0	<p>Generate Capture/Compare 1 Event This bit is set to 1 by software to generate a capture/compare event, which is automatically cleared to 0 by hardware.</p> <p>0: No action;</p> <p>1: Generate a capture/compare event on channel CC1:</p> <p>If channel CC1 is configured as an output:</p> <p>Set CC1IF=1 to generate the corresponding interrupt request if the corresponding interrupt and DMA are turned on.</p> <p>If channel CC1 is configured as an input:</p> <p>The current counter value is captured to the TIM14_CCR1 register, CC1IF=1 is set, and the corresponding interrupt request is generated if the corresponding interrupt is turned on. If CC1IF is already 1, set CC1OF=1.</p>
0	UG	W	0	<p>An update event is generated and this bit is set to 1 by software and cleared to 0 automatically by hardware.</p> <p>0: No action;</p> <p>1: Reinitialize the counter and generate a register update event. Note that the prescaler counter is also cleared to 0 (but the prescaler coefficients remain unchanged).</p>

24.4.5. TIM14 Capture/Compare Mode Register 1 (TIM14_CM1)

Address offset: 0x18

Reset value: 0x0000 0000

Output comparison mode:

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

-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res								Res	OC1M[2: 0]			OC1PE	Res	CC1S[1: 0]	
-								-	RW	RW	RW	RW	-	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 7	Res	-	-	Res
6: 4	OC1M[2: 0]	RW	00	<p>Output Comparison 1 Mode</p> <p>This bit defines the action of the output reference signal OC1REF, which determines the value of OC1, OC1N.</p> <p>OC1REF is active high, and the active level of OC1, OC1N depends on the CC1P, CC1NP bits.</p> <p>000: Frozen. Comparison between output comparison register TIMx_CCR1 and counter TIMx_CNT does not work for OC1REF.</p> <p>Use;</p> <p>001: Set channel 1 as the effective level for matching. Force OC1REF high when the value of counter TIMx_CNT is the same as capture/compare register 1 (TIMx_CCR1).</p> <p>010: Set channel 1 as the invalid level for matching. When the value of counter TIMx_CNT is the same as the capture/compare register, the value of TIMx_CNT is the same as the value of TIMx_CNT.</p> <p>1 (TIMx_CCR1) is the same, force OC1REF low.</p> <p>011: Flip. Flips the level of OC1REF when TIMx_CCR1 = TIMx_CNT.</p> <p>100: Forced to an invalid level. Force OC1REF to be low.</p> <p>101: Forced to active level. Force OC1REF to be high.</p> <p>110: PWM mode 1-</p> <p>In upward counting, channel 1 is valid once TIMx_CNT < TIMx_CCR1, otherwise it is an invalid level; in downward counting, channel 1 is invalid (OC1REF=0) once TIMx_CNT > TIMx_CCR1, otherwise it is a valid level (OC1REF=1).</p> <p>111: PWM mode 2</p> <p>Channel 1 is at an invalid level once TIMx_CNT < TIMx_CCR1, otherwise it is at an active level.</p>

Bit	Name	R/W	Reset Value	Function
				Note: In PWM mode 1 or PWM mode 2, the OC1REF level changes only when the comparison result is changed or when switching from freeze mode to PWM mode in the output comparison mode.
3	OC1PE	RW	0	Output compare 1 preload enable 0: Disable the preload function of TIM14_CCR1 register, which can be written to TIM14_CCR1 register at any time and the new value works immediately. 1: Enable the preload function of TIM14_CCR1 register, read/write operation only operates on the preloaded register, and the preloaded value of TIM14_CCR1 is loaded into the current register on the arrival of an update event.
2	Res	-	-	Res
1: 0	CC1S[1: 0]	RW	00	Capture/Compare1 Select. These 2 bits define the direction of the channel (input/output), and the selection of the input pin: 00: CC1 channel is configured as an output; 01: CC1 channel is configured as an input and IC1 is mapped on TI1; 10: Reserved; 11: Reserved. Note: CC1S is writable only when the channel is closed (CC1E=0 in the TIM14_CCER register).

Input Capture Mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res								IC1F[3: 0]				IC1PSC[1: 0]		CC1S[1: 0]	
-								RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 21	Res	-	-	Res
7: 4	IC1F[3: 0]	RW	0000	Input Capture 1 Filter These bits define the sampling frequency of the TI1 input and the digital filter length. The digital filter consists of an event counter that It records N events and then produces a jump in output:

				<p>0000: no filter, sampled at fDTS</p> <p>0001: Sampling frequency fSAMPLING=fCK_INT, N=2</p> <p>0010: Sampling frequency fSAMPLING=fCK_INT, N=4</p> <p>0011: Sampling frequency fSAMPLING=fCK_INT, N=8</p> <p>0100: Sampling frequency fSAMPLING=fDTS/2, N=6</p> <p>0101: Sampling frequency fSAMPLING = fDTS/2, N=8</p> <p>0110: Sampling frequency fSAMPLING = fDTS/4, N=6</p> <p>0111: Sampling frequency fSAMPLING=fDTS/4, N=8</p> <p>1000: Sampling frequency fSAMPLING = fDTS/8, N=6</p> <p>1001: Sampling frequency fSAMPLING = fDTS/8, N=8</p> <p>1010: Sampling frequency fSAMPLING = fDTS/16, N=5</p> <p>1011: Sampling frequency fSAMPLING=fDTS/16, N=6</p> <p>1100: Sampling frequency fSAMPLING = fDTS/16, N=8</p> <p>1101: Sampling frequency fSAMPLING=fDTS/32, N=5</p> <p>1110: Sampling frequency fSAMPLING = fDTS/32, N=6</p> <p>1111: Sampling frequency fSAMPLING = fDTS/32, N=8</p>
3: 2	IC1PSC[1: 0]	RW	00	<p>Capture/Compare 1 prescaler</p> <p>These 2 bits define the prescaling factor for the CC1 input (IC1). Once CC1E=0 (in the TIMx_CCER register), the prescaler is reset.</p> <p>00: No prescaler, each edge detected on the capture input triggers a capture;</p> <p>01: Capture is triggered every 2 events;</p> <p>10: Capture is triggered every 4 events;</p> <p>11: Capture is triggered every 8 events.</p>
1: 0	CC1S[1: 0]	RW	00	<p>CC1S[1:0]: capture/compare 1 selection.</p> <p>These 2 bits define the direction of the channel (input/output), and the selection of the input pin:</p> <p>00: CC1 channel is configured as an output;</p> <p>01: CC1 channel is configured as an input and IC1 is mapped on TI1;</p> <p>10: Res</p> <p>11: Res</p> <p>Note: CC1S is writable only when the channel is closed (CC1E=0 in the TIM14_CCER register).</p>

24.4.6. TIM14 Capture/Compare Enable Register (TIM14_CCER)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	CC1NP	Res	CC1P	CC1E
												RW		RW	RW

Bit	Name	R/W	Reset Value	Function
31: 4	Res	-	-	Res
3	CC1NP	RW	0	<p>Capture/compare 1 complementary output polarity</p> <p>The CC1 channel is configured as an output:</p> <p>CC1NP must remain 0.</p> <p>The CC1 channel is configured as an input:</p> <p>CC1NP and CC1P are used in conjunction to define TI1FP1 polarity (refer to CC1P description)</p>
2	Res	-	-	Res
1	CC1P	RW	0	<p>Capture/Compare 1 Output Polarity</p> <p>The CC1 channel is configured as an output:</p> <p>0: OC1 active high;</p> <p>1: OC1 active low.</p> <p>The CC1 channel is configured as an input:</p> <p>The CC1NP/CC1P bits select the polarity of TI1FP1 and TI2FP1 used as trigger or capture signals.</p> <p>00: No inversion/rising edge:</p> <p>TIxFP1 rising edge active (triggered in capture mode);</p> <p>01: Inverted/falling edge:</p> <p>TIxFP1 falling edge active (triggered in capture mode);</p> <p>10: Reserved, do not use this configuration.</p> <p>11: Non-inverting/double-edge</p> <p>TIxFP1 is valid on both rising and falling edges (triggered in capture mode);</p>
0	CC1E	RW	0	<p>Capture/compare 1 output enable</p> <p>The CC1 channel is configured as an output:</p> <p>0: off - OC1 disable output</p> <p>1: On - OC1 signal output to corresponding output pin CC1</p> <p>The channel is configured as an input:</p> <p>This bit determines whether the counter value can be captured into the TIMx_CCR1 register.</p> <p>0: Capture prohibited</p>

				1: Capture Enable
--	--	--	--	--------------------------

CcxEN bit	OCx output State
0	Output inhibit (OCx=0,OCx_EN=0)
1	OCx=OCxREF+Polarity,OCx_EN=1

24.4.7. TIM14 Counter (TIM14_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 16	Res	-	-	Res
15: 0	CNT [15:0]	RW	0	Counter value

24.4.8. TIM14 Prescaler (TIM14_PSC)

Address offset: 0x28

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 16	Res	-	-	Res
15: 0	PSC [15:0]	RW	0	Prescaler value The counter clock frequency (CK_CNT) is equal to $f_{CK_PSC}/(PSC[15:0]+1)$. The PSC contains the value loaded into the current pre-scaler register when the update event is generated; the update event includes the counter

				Cleared to 0 by the UG bit of TIM_EGR or by a slave controller operating in reset mode.
--	--	--	--	---

24.4.9. TIM14 Automatic Reload Register (TIM14_ARR)

Address offset: 0x2c

Reset value: 0x0000 FFFF

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

Bit	Name	R/W	Reset Value	Function
31: 16	Res	-	-	Res
15: 0	ARR [15:0]	RW	FFFF	<p>Auto-reload values</p> <p>The ARR contains the value that will be loaded into the actual auto-reload register.</p> <p>Refer to 24.3.1: Updates and actions of the time base unit regarding ARR for details.</p> <p>The counter does not work when the value of Auto-Reload is empty.</p>

24.4.10. TIM14 Capture/Compare Register 1 (TIM14_CCR1)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15: 0]															
RW															

Bit	Name	R/W	Reset Value	Function
31: 16	Res	-	-	Res
15: 0	CCR1[15: 0]	RW	0	<p>Capture/compare the value of 1</p> <p>If the CC1 channel is configured as an output:</p>

Bit	Name	R/W	Reset Value	Function
				<p>CCR1 contains the value loaded into the current Capture/Compare 1 register (preloaded value).</p> <p>If the preload feature is not selected in the TIMx_CMR1 register (OC1PE bit), it is always loaded into the current register.</p> <p>Otherwise, this preloaded value is loaded into the current Capture/Compare 1 register only when an update event occurs.</p> <p>The current capture/compare register contains the value to be compared with counter TIMx_CNT and is signaled on the OC1 port.</p> <p>If the CC1 channel is configured as an input:</p> <p>CCR1 contains the counter value transmitted by the last input capture 1 event (IC1).</p>

24.4.11. TIM14 option register (TIMx_OR)

Address offset: 0x50

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res														TI1_RMP	
-														RW	

Bit	Name	R/W	Reset Value	Function
31: 2	Res	-	-	Res
1: 0	TI1_RMP	RW	0	<p>Timer input 1 Remap</p> <p>Set and cleared by software.</p> <p>00: TIM14 channel 1 is connected to the GPIO, refer to the datasheet for the multiplexing function.</p> <p>01: TIM14 channel 1 connected to RTCCLK.</p> <p>10: TIM14 channel 1 connected to HSE/32 clock</p> <p>11: TIM14 channel 1 is connected to the MCU clock output (MCO). This configuration is determined by the setting of MCO[2:0] in the RCC_CFG register.</p>

25. General-purpose timers (TIM15/16/17)

25.1. Introduction

General-purpose timers (TIM15_16_17) consist of a 16-bit auto-reload counter driven by a programmable prescaler, applicable in various scenarios, including measuring pulse length of input signals (input capture) or generating output waveforms (output compare, output PWM, complementary PWM with dead-time insertion).

Pulse length and waveform period can be modulated from microseconds to milliseconds using timer prescalers and RCC clock control dividers. General-purpose timers (TIM15_16_17) and general-purpose (TIMx) timers are completely independent and do not share any resources. TIM15_16_17 can be synchronized with other TIMERS.

25.2. TIM15 main features

- 16-bit up-counting auto-reload counter
- 16-bit programmable prescaler, allowing clock frequency division from 1 to 65536 for the counter
- Up to 2 independent channels
 - Input capture
 - Output compare
 - PWM generation (edge-aligned mode)
 - Single-pulse mode output
- Complementary outputs with programmable dead time (Channel 1 only)
- Synchronization circuit using external signals to control timers and timer interconnections
- Repetition counter, which updates the timer's registers only after counting a specified number of cycles
- The brake input can set the timer's output signal to a reset state or a known state
- Interrupt/DMA generation on the following events
 - Update: Counter overflow, counter initialization (via software or internal/external trigger)
 - Trigger event
 - Input capture
 - Output compare
 - Brake input

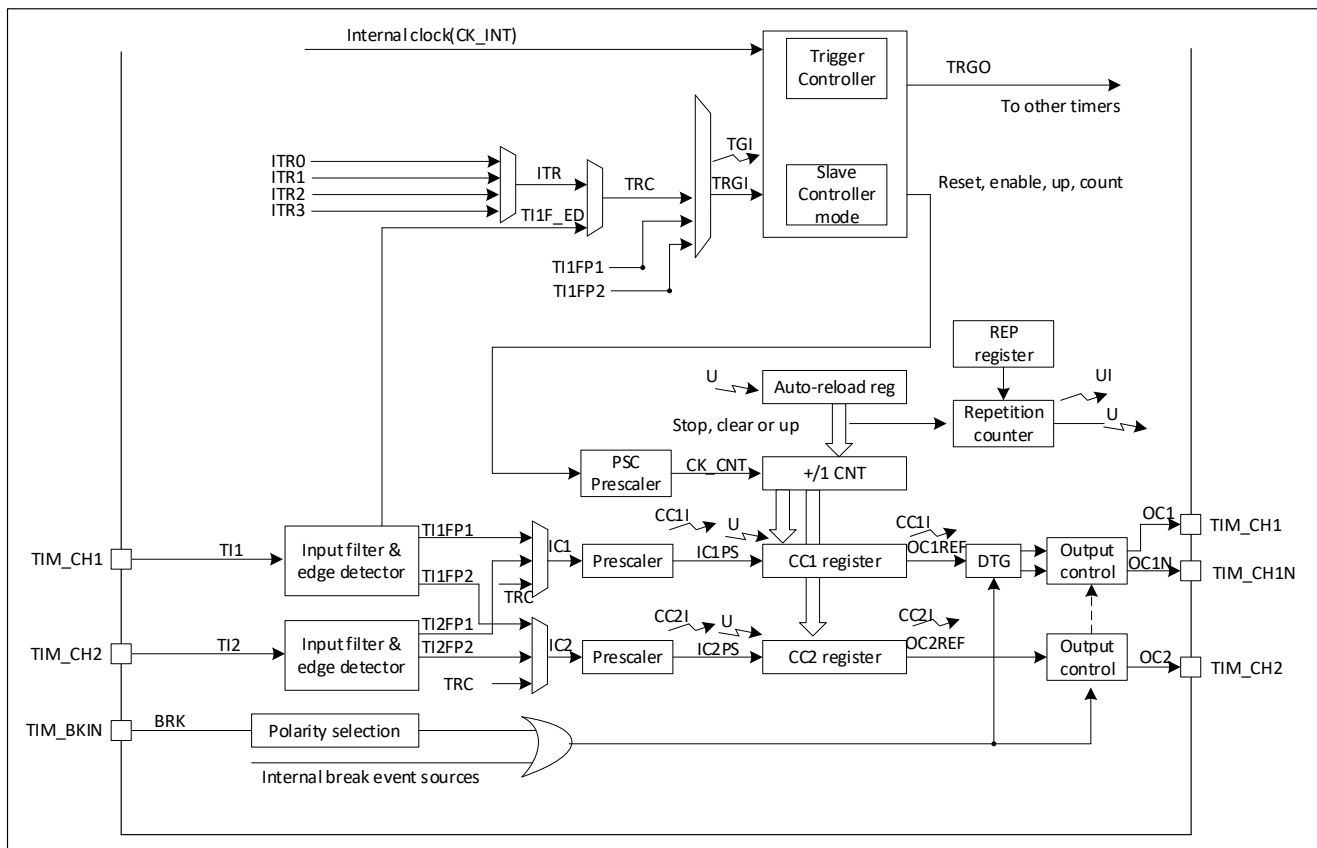


Figure 25-1 Block diagram of general-purpose control timer (TIM15) architecture

25.3. TIM16_17 main features

- 16-bit up-counting auto-reload counter
- 16-bit programmable prescaler, allowing clock frequency division from 1 to 65536 for the counter
- 1 independent channel
 - Input capture
 - Output compare
 - PWM generation (edge-aligned mode)
 - Single-pulse mode output
- Complementary outputs with programmable dead-time
- Repetition counter, which updates the timer's registers only after counting a specified number of cycles
- The brake input can set the timer's output signal to a reset state or a known state
- Interrupt/DMA generation on the following events
 - Update: Counter overflow
 - Input capture
 - Output compare
 - Brake input

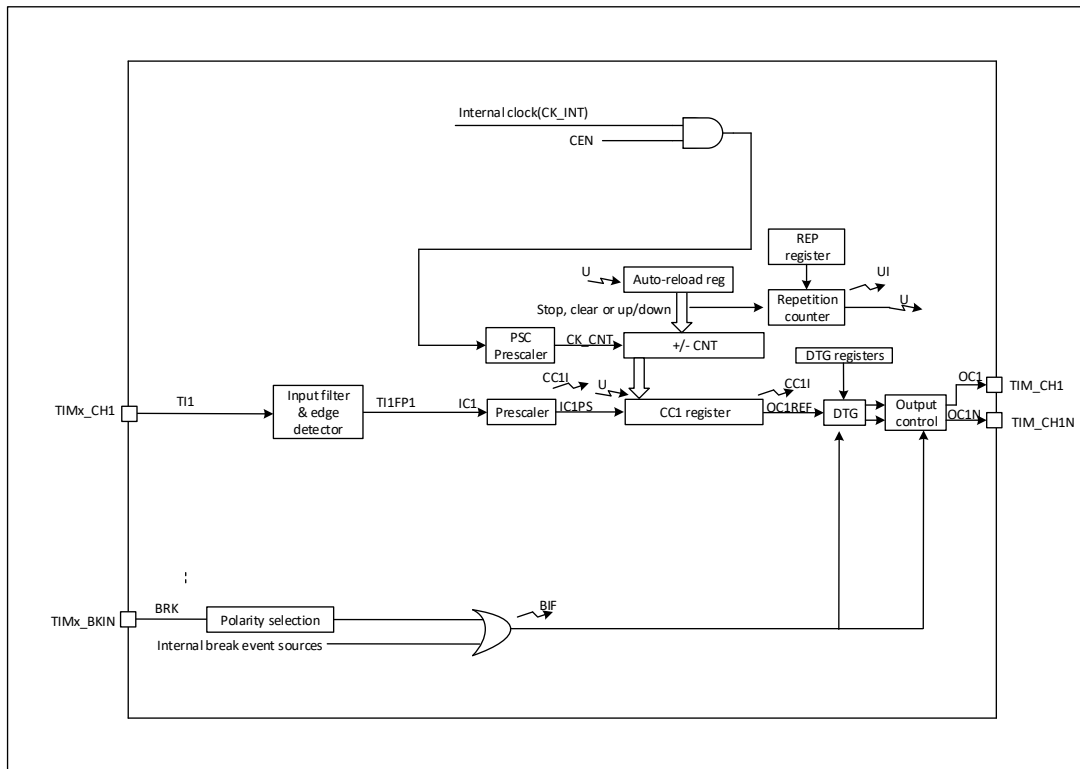


Figure 25-2 Block diagram of general-purpose control timer (TIM16_17) architecture

25.4. TIM15_16_17 functional description

25.4.1. Time base unit

The main component of a programmable general-purpose control timer is a 16-bit counter and its associated auto-reload register. This counter can count up. This counter clock is derived from the prescaler. The counter, auto-reload register, and prescaler register can be read or written by software, even while the counter is running.

The time base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Auto-reload register (TIMx_ARR)
- Repetition counter register (TIMx_RCR)

The auto-reload register is preloaded; writing to or reading from the auto-reload register accesses the preload register. Depending on the setting of the auto-reload preload enable bit (ARPE) in the TIMx_CR1 register, the contents of the preload register are transferred to the shadow register immediately or at each update event (UEV). An update event is generated when the counter overflows and the UDIS bit in the TIMx_CR1 register is 0. Update events can also be generated by software.

The counter is driven by the prescaler's clock output CK_CNT, which is only active when the counter enable bit (CEN) in the TIMx_CR1 register is set.

Note that the counter starts counting one clock cycle after the CEN bit in the TIMx_CR register is set.

Prescaler description

The prescaler can divide the counter clock by any value between 1 and 65536. It is a 16-bit counter based on a 16-bit register control (in the TIMx_PSC register). Because this control register has a buffer, it can be modified during operation. The new prescaler parameter is applied when the next update event occurs.

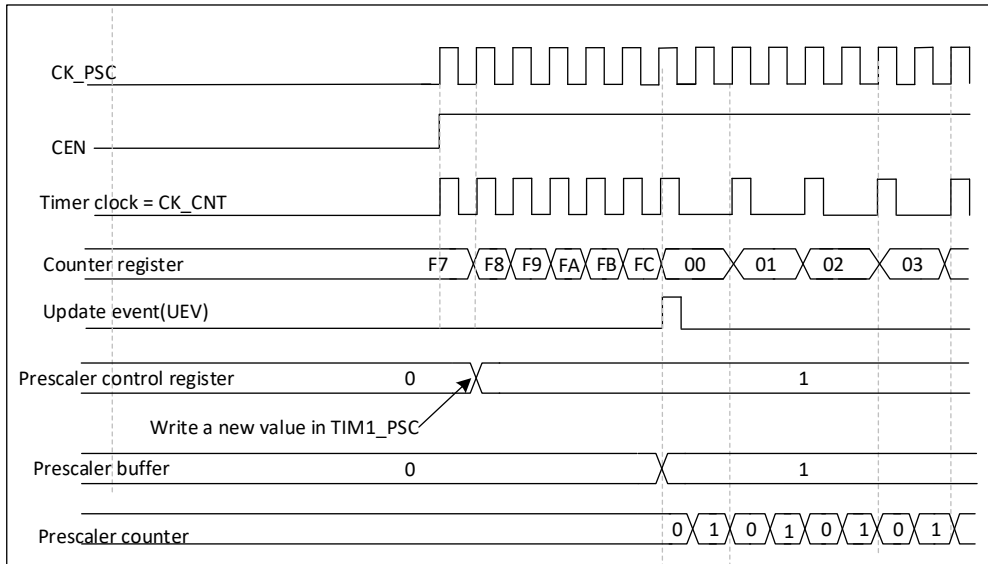


Figure 25-3 Timing diagram of the counter when the prescaler parameter changes from 1 to 2

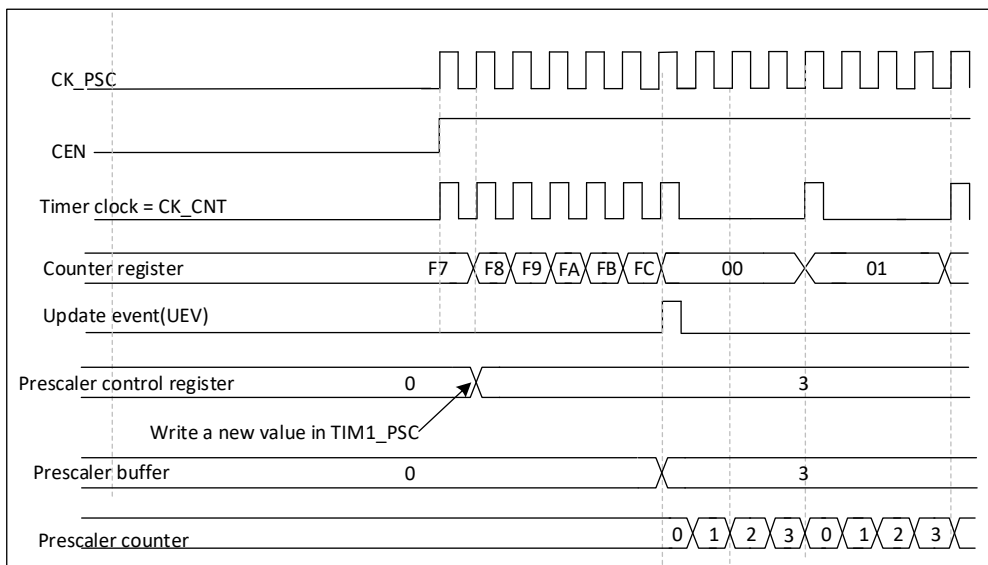


Figure 25-4 Timing diagram of the counter when the prescaler parameter changes from 1 to 4

25.4.2. Counter mode

Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value, then restarts from 0 and generates a counter overflow event.

If the repetition counter is used, an update event is generated after the number of overflows reaches the configured repetition count register value plus one (i.e., $TIMx_RCR+1$). Otherwise, an update event is generated at each counter overflow.

Setting the UG bit in $TIMx_EGR$ register (either by software or using the slave mode controller) can also generate an update event.

Setting the UDIS bit in $TIMx_CR1$ register disables update events; this also prevents updating shadow registers when writing new values to preload registers. No update events will be generated until the UDIS bit is cleared. Even so, when an update event should occur, the counter is cleared to '0' and the prescaler count is reset to 0 (though the prescaler ratio remains unchanged). Additionally, if the URS bit in $TIMx_CR1$ register is set (update request selection), setting the UG bit generates an update event UEV but hardware does not set the UIF flag (no interrupt or DMA request). This prevents simultaneous update and capture interrupts when clearing the counter in capture mode.

When an update event occurs, all registers are updated, and hardware sets the update flag (UIF bit in $TIMx_SR$ register) simultaneously (according to URS bit).

- The repetition counter is reloaded with the contents of the $TIMx_RCR$ register.
- The auto-reload shadow register is reloaded with the preload register value ($TIMx_ARR$).
- The prescaler buffer is loaded with the value from the preload register (contents of $TIMx_PSC$ register).

The following diagram shows examples of counter behavior at different clock frequencies when $TIMx_ARR=0x36$.

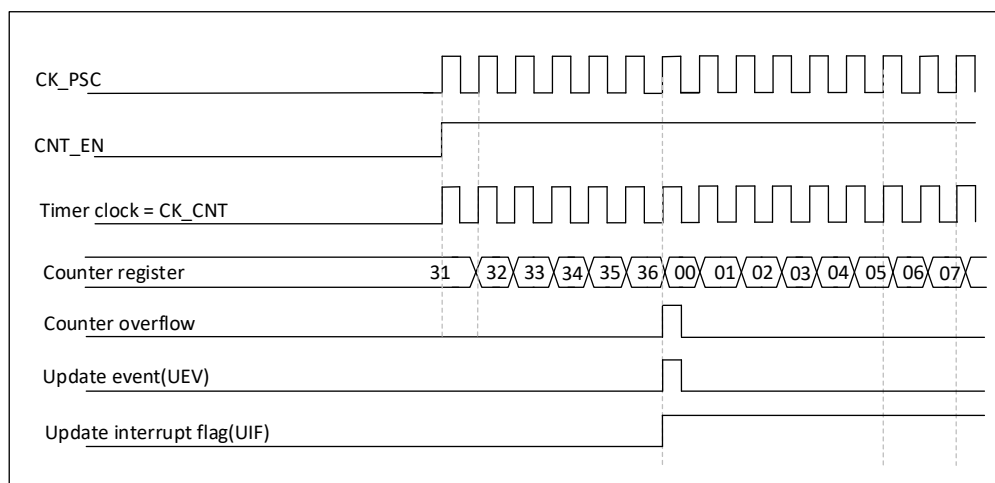


Figure 25-5 Counter timing diagram with internal clock division factor 1

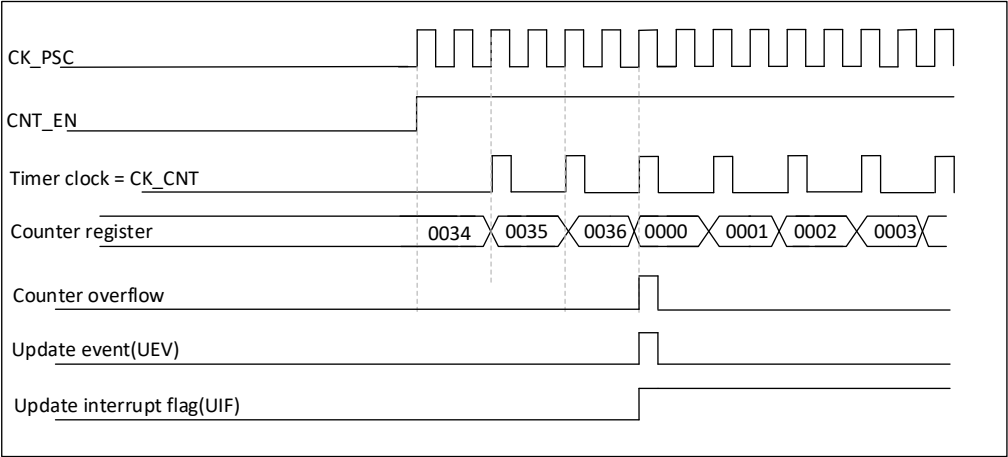


Figure 25-6 Counter timing diagram with internal clock division factor 2

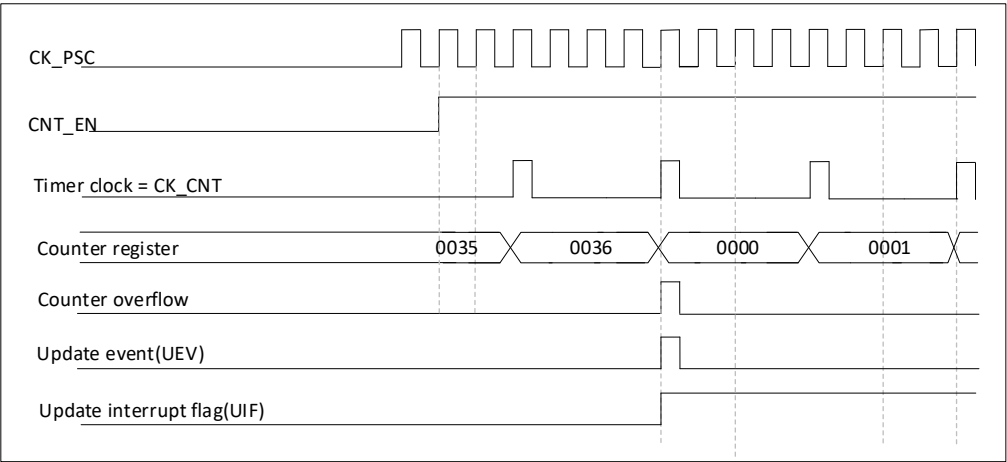


Figure 25-7 Counter timing diagram with internal clock division factor 4

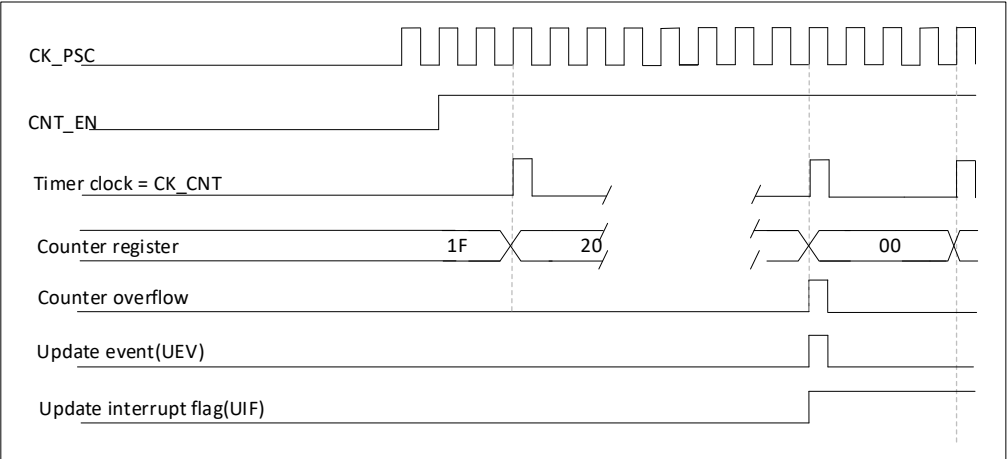


Figure 25-8 Counter timing diagram with internal clock division factor N

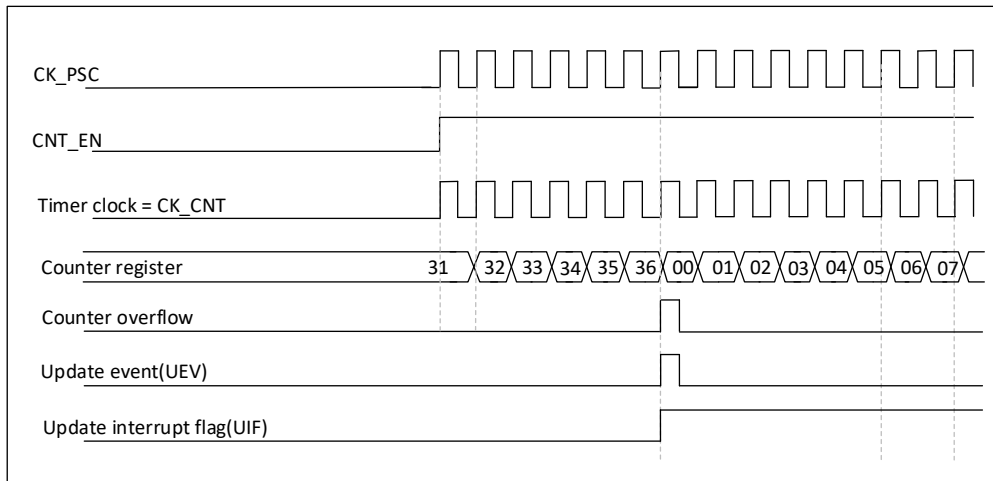


Figure 25-9 Counter timing diagram showing update events when ARPE=0 (TIMx_ARR not preloaded)

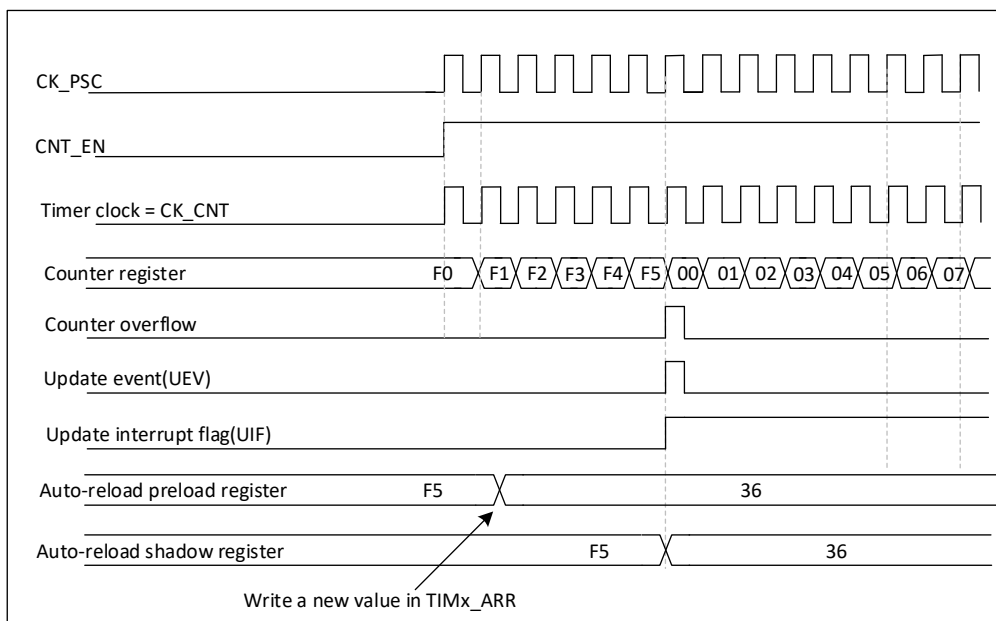


Figure 25-10 Counter timing diagram showing update events when ARPE=1 (TIMx_ARR preloaded)

25.4.3. Repetition counter

The time base unit describes how update events are generated upon counter overflow. It is actually generated only when the repetition counter counts down to zero. This is also useful when generating PWM signals.

This means that on every N+1 count overflow, data is transferred from the preload register to the shadow register (TIMx_ARR auto-reload register, TIMx_PSC preload register, and the capture/compare register TIMx_CCRx in compare mode), where N is the value in the TIMx_RCR repetition counter register.

The repetition counter decrements when the following condition is met:

- On each overflow in up-counting mode

The repetition counter is auto-loaded, and the repetition rate is defined by the value in the TIMx_RCR register. When an update event is generated by software (by setting the UG bit in TIMx_EGR) or by the

hardware slave mode controller, an update event occurs immediately regardless of the value of the repetition counter, and the content of the TIMx_RCR register is reloaded into the repetition counter.

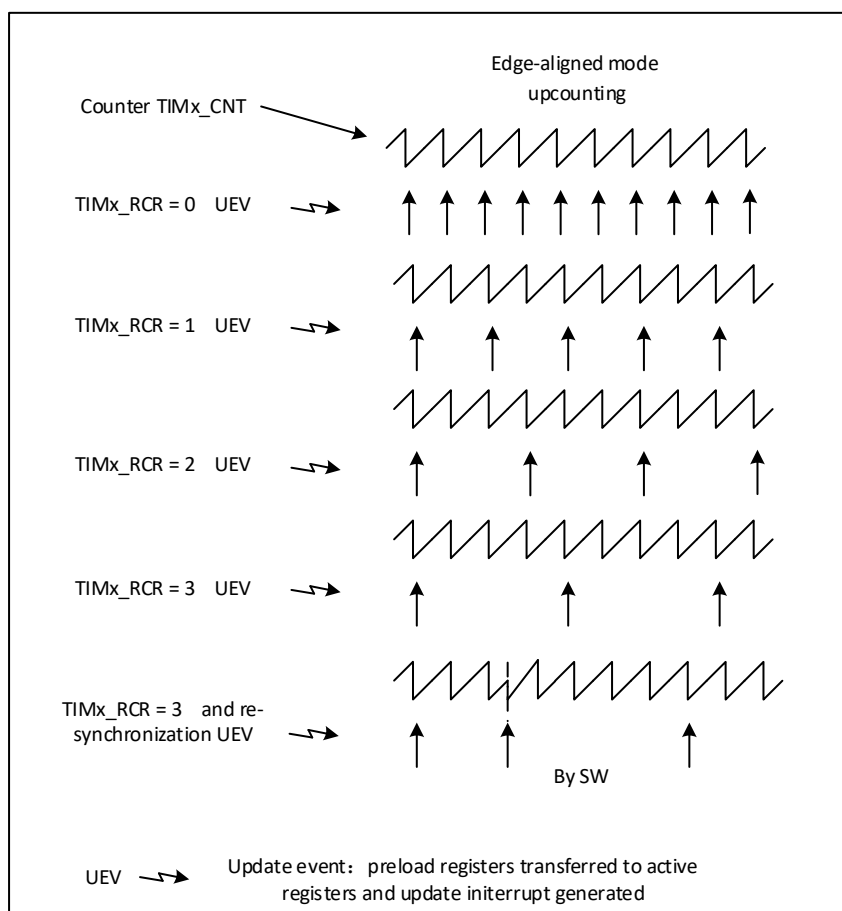


Figure 25-11 Example of update rates in different modes, and TIMx_RCR register settings

25.4.4. Clock source

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

- Internal clock (CK_INT)
- External clock mode 1: External input pin (TMI15 only)
- Internal trigger input (ITRx): Use one timer as a prescaler for another timer. For example, Timer1 can be configured as a prescaler for another timer, Timer2. (TMI15 only)

Internal clock source (CK_INT)

If the slave mode controller is disabled, the CEN, DIR (TIMx_CR1 register), and UG bits (TIMx_EGR register) are the de facto control bits and can only be modified by software. As long as the CEN bit is written as 1, the clock of the prescaler is provided by the internal clock CK_INT.

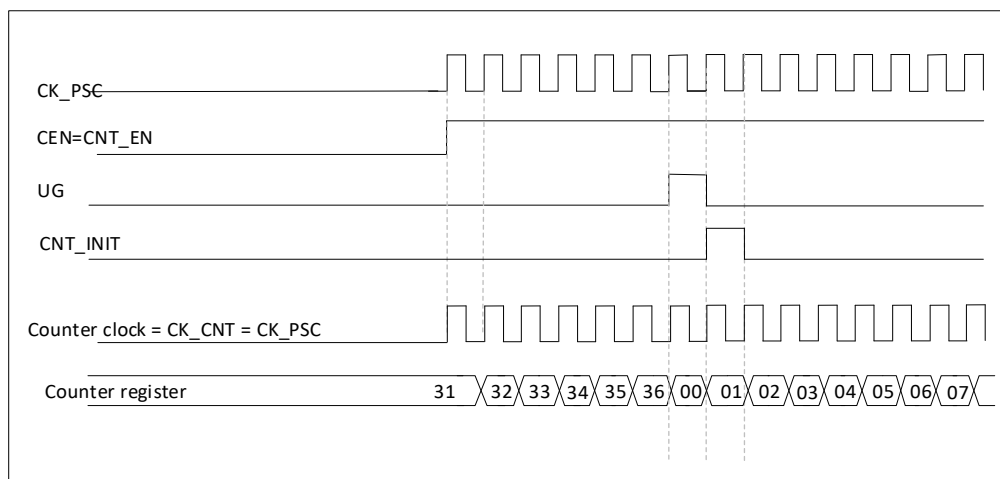


Figure 25-12 Control circuit in normal mode, internal clock division factor is 1

External clock source mode 1 (TIM15 only)

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

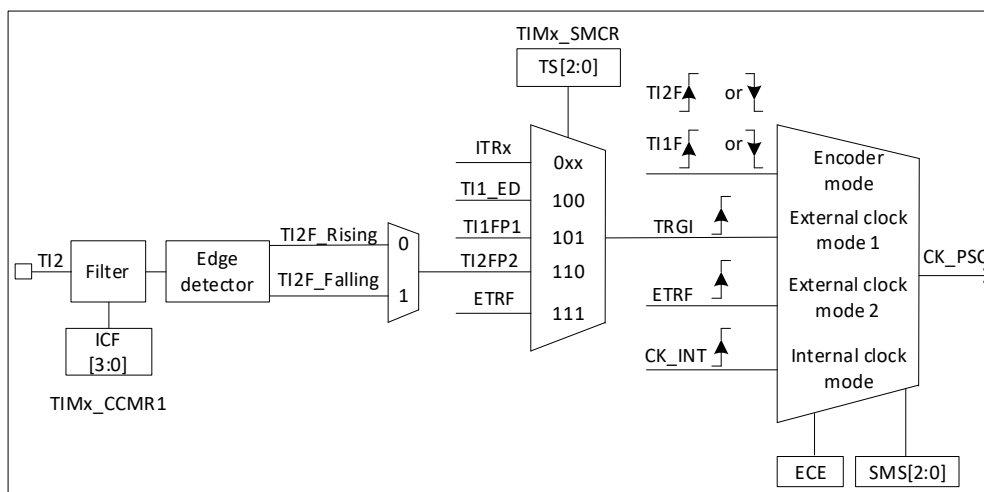


Figure 25-13 Example of TI2 external clock connection

For example, to configure the counter to count up on the rising edge at the T12 input, use the following steps:

1. Configure the TIMx_CCMR1 register CC2S=01 to make channel 2 detect the rising edge at the TI2 input;
2. Configure IC2F[3:0] in the TIMx_CCMR1 register to select the input filter bandwidth (if no filter is needed, keep IC2F=0000).
3. Configure CC2P=0 in the TIMx_CCER register to select rising edge polarity.
4. Configure SMS=111 in the TIMx_SMCR register to set the timer to external clock mode 1.
5. Configure TS=110 in the TIMx_SMCR register to select TI2 as the trigger input source.
6. Set CEN=1 in the TIMx_CR1 register to start the counter.

Note: The capture prescaler is not used for triggering, so no configuration is required for it.

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

The delay between the rising edge of TI2 and the actual clock of the counter depends on the resynchronization circuit at the TI2 input.

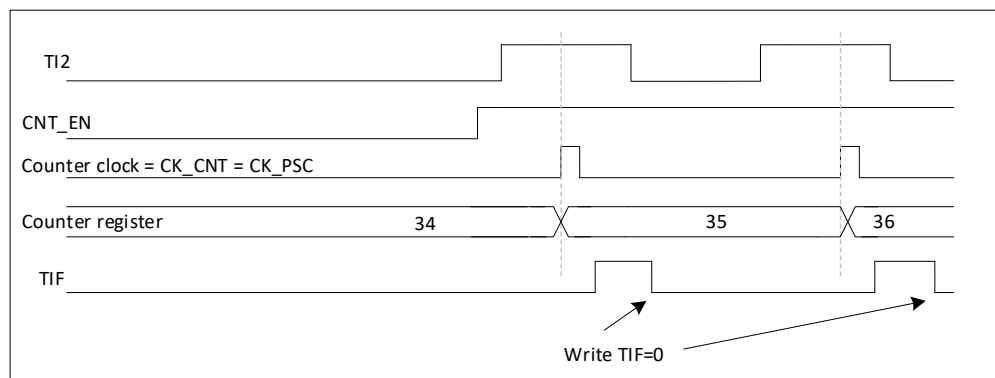


Figure 25-14 Control circuit in external clock mode 1

25.4.5. Capture/Compare Channel

Each capture/compare channel is built around a capture/compare register (including a shadow register), comprising an input section for capture (digital filter, multiplexer, and prescaler), and an output section (comparator and output control).

The input section samples the corresponding TIx input signal and produces a filtered signal TIxF. Then, an edge detector with polarity selection generates a signal (TIxFPx), which can serve as an input trigger for the slave mode controller or as capture control. This signal enters the capture register (ICxPS) through the prescaler.

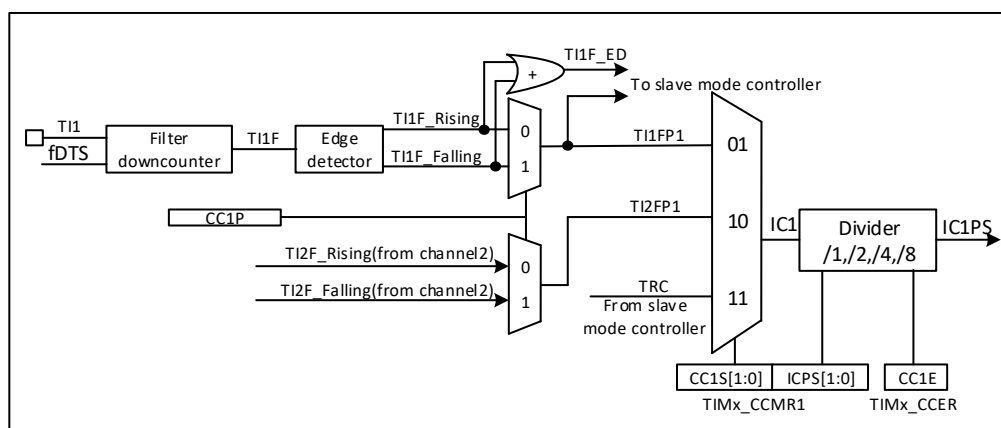


Figure 25-15 Capture/compare channel (e.g., input section of Channel 1)

The output section generates an intermediate waveform OCxREF (active high) as a reference, and the end of the chain determines the polarity of the final output signal.

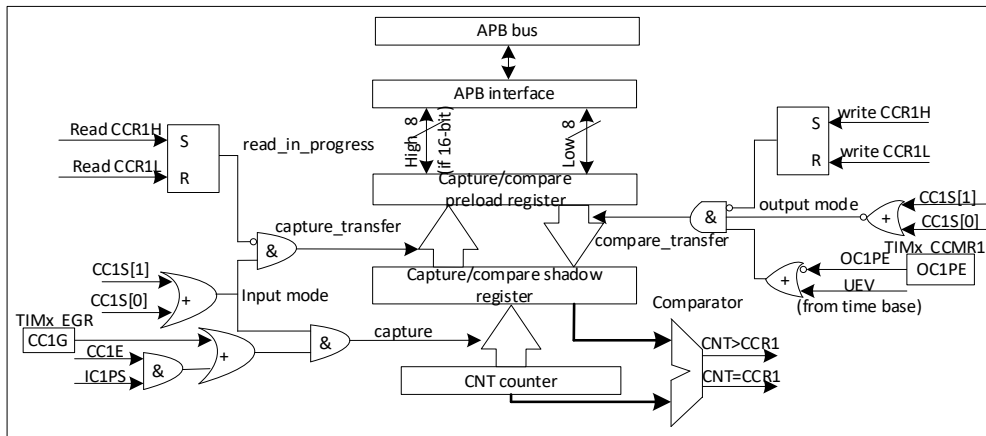


Figure 25-16 Main circuit of capture/compare channel 1

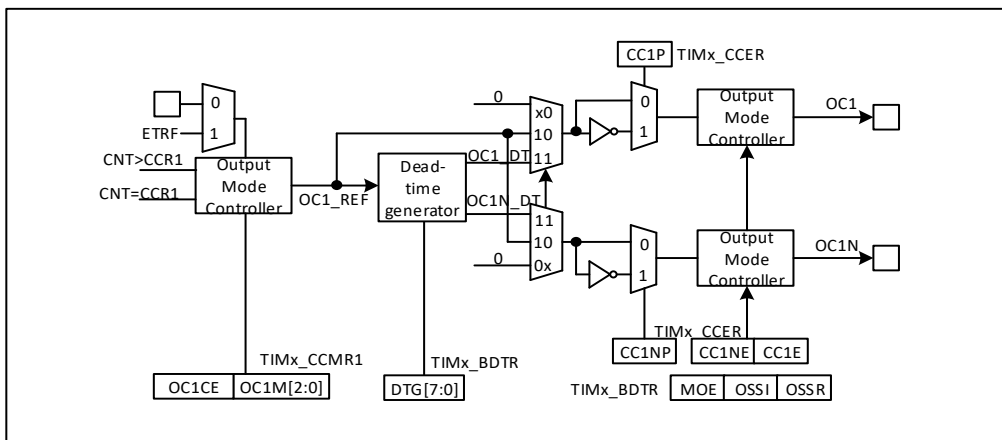


Figure 25-17 Output section of the capture/compare channel (Channels 1 to 3)

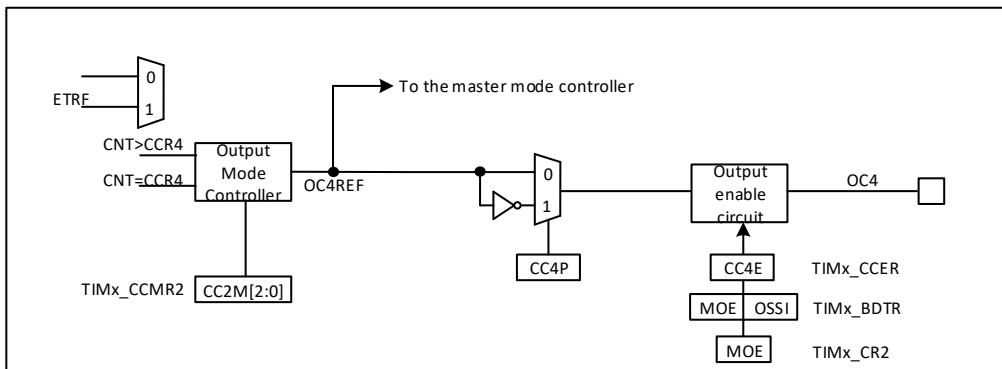


Figure 25-18 Output section of the capture/compare channel (Channel 4)

The capture/compare module consists of a preload register and a shadow register. Read and write operations only access the preload register.

In capture mode, the capture occurs on the shadow register before being copied to the preload register.

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

25.4.6. Input Capture mode

In input capture mode, when the corresponding edge is detected on the ICx signal, the current value of the counter is latched into the capture/compare register. When a capture event occurs, the corresponding

CCxIF flag (TIMx_SR register) is set to 1. If interrupts or DMA operations are enabled, an interrupt or DMA request will be generated. If the CCxIF flag is already high when a capture event occurs, the repeat capture flag CCxOF (TIMx_SR register) is set to 1. Writing CCxIF=0 clears CCxIF, or reading the captured data stored in the TIMx_CCRx register also clears CCxIF. Writing CCxOF=0 clears CCxOF.

The following example illustrates how to capture the counter value into the TIMx_CCR1 register on the rising edge of the TI1 input. The steps are as follows:

- Select the active input: TIMx_CCMR1 must be connected to the TI1 input, so write CC1S=01 in the TIMx_CCMR1 register. As long as CC1S is not '00', the channel is configured as an input, and the TIMx_CCR1 register becomes read-only.
- Based on the characteristics of the input signal, configure the input filter to the desired bandwidth (i.e., when the input is TIx, the input filter control bits are the ICxF bits in the TIMx_CCMRx register). Assuming the input signal jitters within a maximum of 5 internal clock cycles, we must configure the filter bandwidth to be longer than 5 clock cycles; therefore, we can sample 8 times consecutively (at fDTS frequency) to confirm a real edge transition on TI1, i.e., write IC1F=0011 in the TIMx_CCMR1 register.
- Select the active transition edge for the TI1 channel by writing CC1P=0 (rising edge) in the TIMx_CCER register.
- Configure the input prescaler. In this example, we want the capture to occur at every valid level transition, so the prescaler is disabled (write IC1PS=00 in the TIMx_CCMR1 register).
- Set CC1E=1 in the TIMx_CCER register to allow capturing the counter value into the capture register.
- If needed, enable the corresponding interrupt request by setting the CC1IE bit in the TIMx_DIER register, and enable the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:

- When a valid level transition occurs, the counter value is transferred to the TIMx_CCR1 register.
- The CC1IF flag is set (interrupt flag). When at least two consecutive captures occur and CC1IF has not been cleared, CC1OF is also set to 1.
- If the CC1IE bit is set, an interrupt will be generated.
- If the CC1DE bit is set, a DMA request will also be generated.

To handle capture overflow, it is recommended to read the data before reading the capture overflow flag to avoid losing capture overflow information that may occur after reading the flag and before reading the data.

Note: Setting the corresponding CCxG bit in the TIMx_EGR register can generate an input capture interrupt and/or DMA request via software.

25.4.7. PWM input mode (TIM15 only)

This mode is a special case of the input capture mode. Except for the following differences, the operation is the same as the input capture mode:

- Both ICx signals are mapped to the same TIx input.
- These two ICx signals are edge-sensitive but have opposite polarities.

- One of the TIxFP signals is used as the trigger input signal, and the slave mode controller is configured to reset mode.

For example, when measuring the length (TIMx_CCR1 register) and duty cycle (TIMx_CCR2 register) of a PWM signal input to TI1, the specific steps are as follows (depending on the frequency of CK_INT and the prescaler value):

- Select the valid input for TIMx_CCR1: Set CC1S=01 in the TIMx_CCMR1 register (select TI1).
- Select the active polarity of TI1FP1 (to capture data to TIMx_CCR1 and clear the counter): Set CC1P=0 (rising edge active).
- Select the valid input for TIMx_CCR2: Set CC2S=10 in the TIMx_CCMR1 register (select TI1).
- Select the active polarity of TI1FP2 (to capture data to TIMx_CCR2): Set CC2P=1 (falling edge active).
- Select a valid trigger input signal: Set TS=101 in the TIMx_SMCR register (select TI1FP1).
- Configure the slave mode controller to reset mode: Set SMS=100 in TIMx_SMCR.
- Enable capture: Set CC1E=1 and CC2E=1 in the TIMx_CCER register.

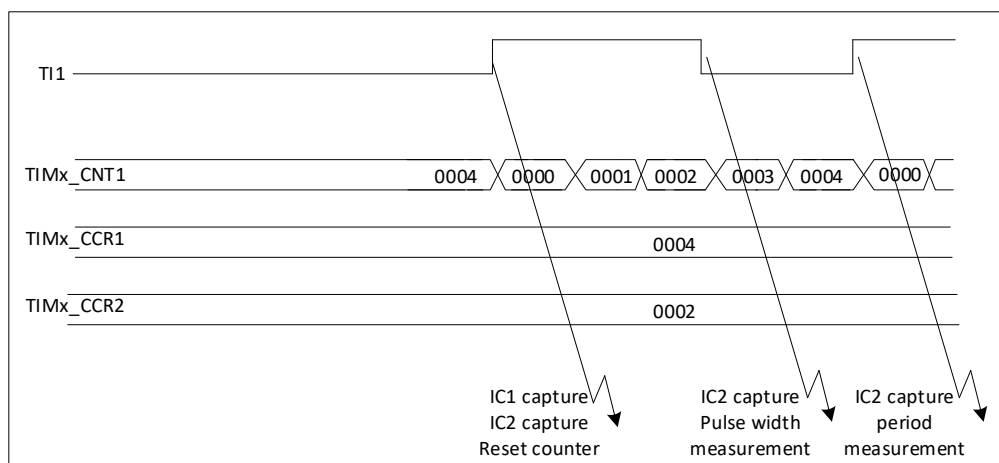


Figure 25-19 PWM input mode timing diagram

25.4.8. Forced output mode

In output mode (CCxS=00 in the TIMx_CCMRx register), the output compare signal (OCxREF and the corresponding OCx/OCxN) can be directly forced to an active or inactive state by software, independent of the comparison result between the output compare register and the counter. Set the corresponding OCxM=101 in the TIMx_CCMRx register to force the output compare signal (OCxREF/OCx) to an active state. Thus, OCxREF is forced high (OCxREF is always active high), while OCx receives a signal with the opposite polarity of CCxP.

For example: CCxP=0 (OCx active high), then OCx is forced high. Set OCxM=100 in the TIMx_CCMRx register to force the OCxREF signal low.

In this mode, the comparison between the TIMx_CCRx shadow register and the counter still occurs, and the corresponding flags are also modified. Therefore, the corresponding interrupt and DMA request will still be generated. This will be described in the output compare mode section below.

25.4.9. Output compare mode

This function is used to control an output waveform or indicate that a given time period has elapsed.

When the counter matches the content of the capture/compare register, the output compare function performs the following operations:

- Output the value defined by the output compare mode (OCxM bit in TIMx_CCMRx register) and output polarity (CCxP bit in TIMx_CCER register) to the corresponding pin. Upon compare match, the output pin can maintain its level (OCxM=000), be set to active level (OCxM=001), be set to inactive level (OCxM=010), or toggle (OCxM=011).
- Set the flag bit in the interrupt status register (CCxIF bit in TIMx_SR register).
- If the corresponding interrupt mask is set (CCxIE bit in TIMx_DIER register), an interrupt is generated.
- If the corresponding enable bit is set (CCxDE bit in TIMx_DIER register, CCDS bit in TIMx_CR2 register selects DMA request function), a DMA request is generated.

The OCxPE bit in TIMx_CCMRx selects whether the TIMx_CCRx register needs to use a preload register. In output compare mode, the update event UEV has no effect on OCxREF and OCx output.

The synchronization accuracy can reach one counter cycle. Output compare mode (in one-pulse mode) can also be used to output a single pulse.

Configuration steps for output compare mode:

1. Select the counter clock (internal, external, prescaler).
2. Write the corresponding data to the TIMx_ARR and TIMx_CCRx registers.
3. If an interrupt request needs to be generated, set the CCxIE bit.
4. Select the output mode, for example:
 - When the counter matches CCRx, toggle the output pin OCx by setting OCxM=011
 - Set OCxPE = 0 to disable the preload register.
 - Set CCxP = 0 to select active-high polarity.
 - Set CCxE = 1 to enable the output.
5. Set the CEN bit in the TIMx_CR1 register to start the counter.

The TIMx_CCRx register can be updated at any time via software to control the output waveform, provided the preload register is not used (OCxPE = '0'; otherwise, the shadow register of TIMx_CCRx can only be updated upon the next update event). An example is given below.

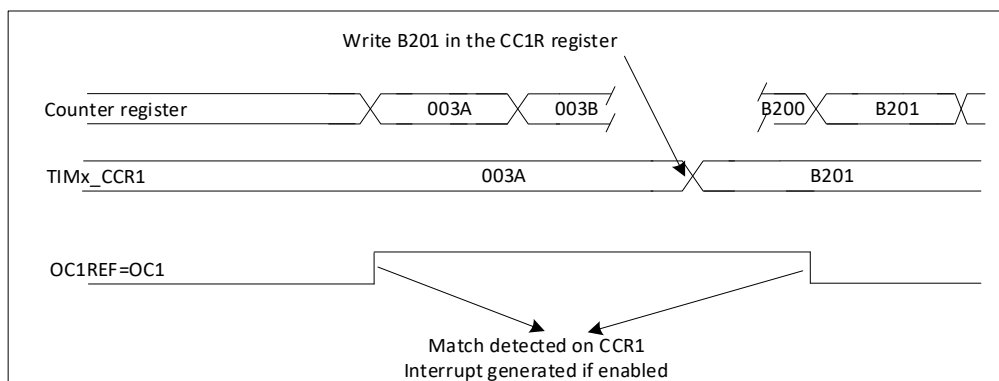


Figure 25-20 Output compare mode, toggle OC1

25.4.10. PWM mode

Pulse Width Modulation mode allows the generation of a signal with a frequency determined by the TIMx_ARR register and a duty cycle determined by the TIMx_CCRx register.

Writing "110" (PWM Mode 1) or "111" (PWM Mode 2) to the OCxM bits in the TIMx_CCMRx register allows each OCx output channel to independently generate a PWM signal. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register. Finally, the ARPE bit in the TIMx_CR1 register must be set to enable the auto-reload preload register (in upcounting or center-aligned mode).

The preload registers can only be transferred to the shadow registers upon an update event. Therefore, all registers must be initialized by setting the UG bit in the TIMx_EGR register before the counter starts counting.

The polarity of OCx can be set via software using the CCxP bit in the TIMx_CCER register, allowing it to be configured as active high or active low. The output enable for OCx is controlled by the combination of the CCxE, CCxNE, MOE, OSSI, and OSSR bits (in the TIMx_CCER and TIMx_BDTR registers). For details, refer to the description of the TIMx_CCER register.

In PWM mode (Mode 1 or Mode 2), TIMx_CNT and TIMx_CCRx are continuously compared (based on the counter's counting direction) to determine whether $\text{TIMx_CCRx} \leq \text{TIMx_CNT}$ or $\text{TIMx_CNT} \leq \text{TIMx_CCRx}$.

Depending on the state of the CMS bit in the TIMx_CR1 register, the timer can generate either edge-aligned or center-aligned PWM signals.

PWM edge-aligned mode

■ Upcounting configuration

Upcounting is performed when the DIR bit in the TIMx_CR1 register is low. Refer to the following example of PWM Mode 1. When $\text{TIMx_CNT} < \text{TIMx_CCRx}$, the PWM reference signal OCxREF is high; otherwise, it is low. If the compare value in TIMx_CCRx is greater than the auto-reload value (TIMx_ARR), OCxREF remains at '1'. If the compare value is 0, OCxREF remains at '0'. The figure below shows an example of edge-aligned PWM waveforms when TIMx_ARR=8.

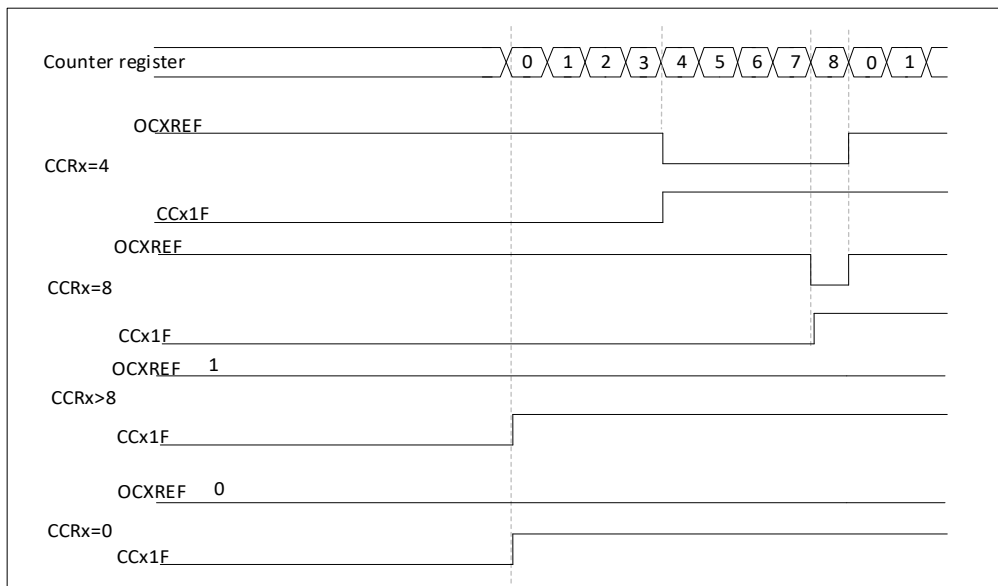


Figure 25-21 Edge-aligned PWM output, up-counting (ARR=8)

25.4.11. Complementary outputs and dead-time insertion

The general-purpose control timers (TIM15_16_17) can output two complementary signals and manage the instantaneous turn-off and turn-on of the outputs. This period is commonly referred to as dead-time, and the user should adjust the dead-time duration based on the connected output devices and their characteristics (level transition delay, power switch delay, etc.).

The polarity (main output OCx or complementary output OCxN) can be independently selected for each output by configuring the CCxP and CCxNP bits in the TIMx_CCER register.

The complementary signals OCx and OCxN are controlled by the combination of the following control bits: CCxE and CCxNE bits in the TIMx_CCER register, MOE, OISx, OISxN, OSS1, and OSSR bits in the TIMx_BDTR and TIMx_CR2 registers. For details, refer to the control bits of the complementary output channels OCx and OCxN with break function in TIMx_CCER. In particular, dead-time is activated during the transition to IDLE state (MOE falling to 0).

Setting both CCxE and CCxNE bits inserts dead-time, and if a break circuit is present, the MOE bit must also be set. Each channel has an 8-bit dead-time generator DTG[7:0]. The 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, except that its rising edge is delayed relative to the rising edge of the reference signal.
- The OCxN output signal is the opposite of the reference signal, except that its rising edge is delayed relative to the falling edge of the reference signal.

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

The following figures show the relationship between the output signals of the dead-time generator and the reference signal OCxREF. (Assuming CCxP=0, CCxNP=0, MOE=1, CCxE=1 and CCxNE=1)

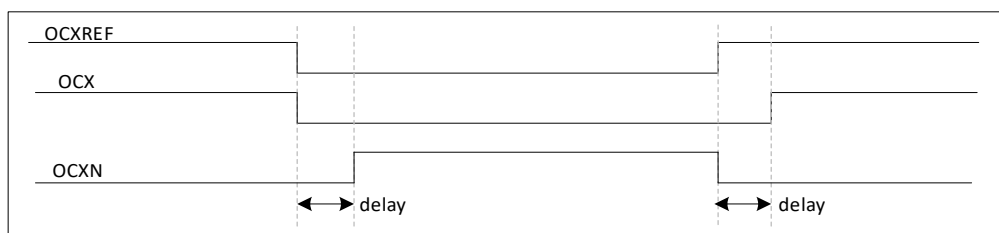


Figure 25-22 Complementary outputs with dead-time insertion

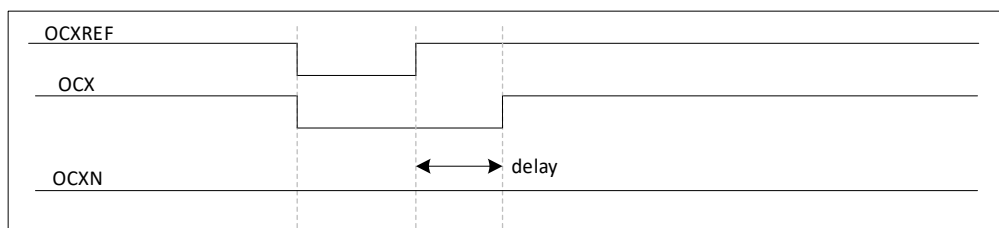


Figure 25-23 Dead-time waveform with delay greater than negative pulse

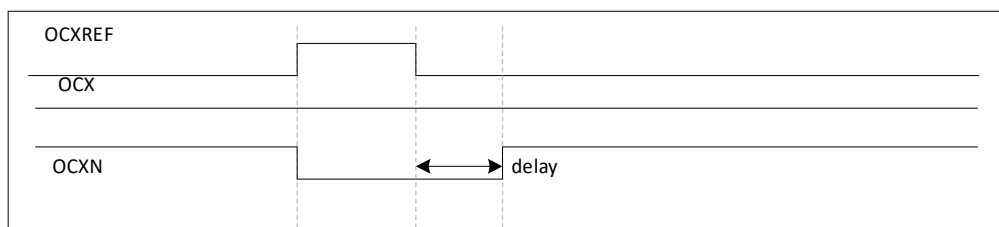


Figure 25-24 Dead-time waveform with delay greater than positive pulse

The dead-time duration is identical for all channels and is programmed with the DTG bits in the TIMx_BDTR register.

Redirect OCxREF to OCx or OCxN

In output mode (forced, output compare, or PWM), OCxREF can be redirected to the output of OCx or OCxN by configuring the CCxE and CCxNE bits in the TIMx_CCER register. This function allows sending a special waveform (e.g., PWM or static active level) on one output while the complementary output is at an inactive level. Another effect is to have both outputs at an inactive level simultaneously or at an active level with complementary outputs including dead time.

Note: When only OCxN is enabled (CCxE=0, CCxNE=1), it is not inverted and immediately goes high when OCxREF is active. For example, if CCxNP=0, then OCxN=OCxREF. On the other hand, when both OCx and OCxN are enabled (CCxE=CCxNE=1), OCx is active when OCxREF is high; conversely, OCxN becomes active when OCxREF is low.

25.4.12. Using the brake function

When using the brake function, both the output enable signal and the inactive level signal are modified based on additional control bits (MOE, OSSI, and OSSR bits in the TIMx_BDTR register, and OISx and OISxN bits in the TIMx_CR2 register). Under no circumstances can the OCx and OCxN outputs be at an active level simultaneously.

The brake source can be either the brake input pin or the following internal sources:

- Core LOCKUP output

- PVD output
- Clock failure event generated by CSS monitoring
- Output from the comparator

After a system reset, the brake circuit is disabled, and the MOE bit is low. Setting the BKE bit in the TIMx_BDTR register enables the brake function, and the polarity of the brake input signal can be selected by configuring the BKP bit in the same register. BKE and BKP can be modified simultaneously. When writing the BKE and BKP bits, there is a delay of 1 APB clock cycle before the actual write occurs, so it is necessary to wait for one APB clock cycle before correctly reading back the written bits.

Since the falling edge of MOE can be asynchronous, a resynchronization circuit is implemented between the actual signal (acting on the output) and the synchronous control bit (in the TIMx_BDTR register). This resynchronization circuit introduces a delay between the asynchronous and synchronous signals. Specifically, if MOE=1 is written when it is low, a delay (NOP instruction) must be inserted before reading it to obtain the correct value. This is because writing is an asynchronous signal while reading is a synchronous signal.

When a brake occurs (the selected level appears at the brake input), the following actions take place:

- The MOE bit is cleared asynchronously, placing the outputs in the inactive state, idle state, or reset state (selected by the OSSR bit). This feature remains valid even when the MCU's oscillator is turned off.
- Once MOE=0, each output channel outputs the level set by the OISx bit in the TIMx_CR2 register. If OSSR=0, the timer releases the enable output; otherwise, the enable output remains high.
- When using complementary outputs:
 - The output is first placed in the reset state, i.e., the inactive state (depending on the polarity). This is asynchronous operation, and this feature remains valid even when the timer has no clock.
 - If the timer clock is still present, the dead-time generator will reactivate, driving the output ports to the levels indicated by the OISx and OISxN bits after the dead time. Even in this case, OCx and OCxN cannot be driven to active levels simultaneously. Note: Due to MOE resynchronization, the dead time is slightly longer than usual (approximately 2 ck_tim clock cycles).
 - If OSSR=0, the timer releases the enable output; otherwise, the enable output remains high, or it goes high once either CCxE or CCxNE becomes high.
- If the BIE bit in the TIMx_DIER register is set, an interrupt is generated when the brake status flag (BIF bit in the TIMx_SR register) is '1'. If the BDE bit in the TIMx_DIER register is set, a DMA request is generated.
- If the AOE bit in the TIMx_BDTR register is set, the MOE bit is automatically set on the next update event UEV; for example, this can be used for shaping. Otherwise, MOE remains low until it is set to '1' again; this feature can be used for safety purposes, where you can connect the brake input to a power driver alarm output, thermal sensor, or other safety device.

Note: The brake input is level-sensitive. Therefore, when the brake input is active, MOE cannot be set (automatically or via software) simultaneously. *Meanwhile, the status flag BIF cannot be cleared.*

The brake can be generated by the BRK input, its active polarity is programmable, and it is enabled by the BKE bit in the TIMx_BDTR register. Braking can also be generated by software by setting the BG bit in the TIMx_EGR register.

In addition to brake input and output management, the brake circuit also implements write protection to ensure application safety. It allows users to freeze several configuration parameters (dead-time length, OCx/OCxN polarity and disabled state, OCxM configuration, brake enable and polarity). Users can select one of three protection levels via the LOCK bit in the TIMx_BDTR register. The LOCK bit can only be modified once after MCU reset.

The following figure shows an example of output response to braking.

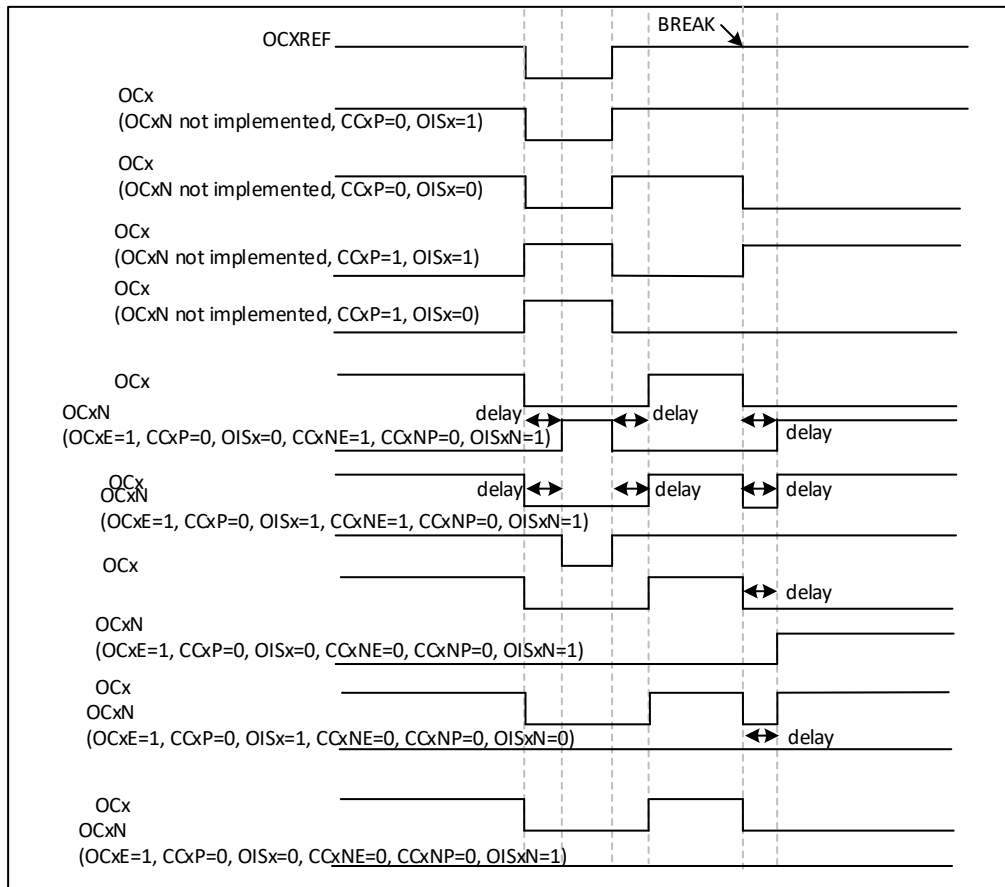


Figure 25-25 Output Response to Brake

25.4.13. One-pulse mode

Single Pulse Mode (OPM) is a special case of the previously described modes. This mode allows the counter to respond to a stimulus and generate a pulse with a programmable width after a programmable delay.

The counter can be started by the slave mode controller to generate waveforms in output compare mode or PWM mode. Setting the OPM bit in the TIMx_CR1 register selects the single pulse mode, allowing the counter to automatically stop when the next update event UEV is generated.

A pulse can only be generated if the compare value differs from the counter's initial value. Before starting (when the timer is waiting for a trigger), the following configuration must be set:

- Up-counting mode: counter $CNT < CCRx \leq ARR$ (specifically, $0 < CCRx$).

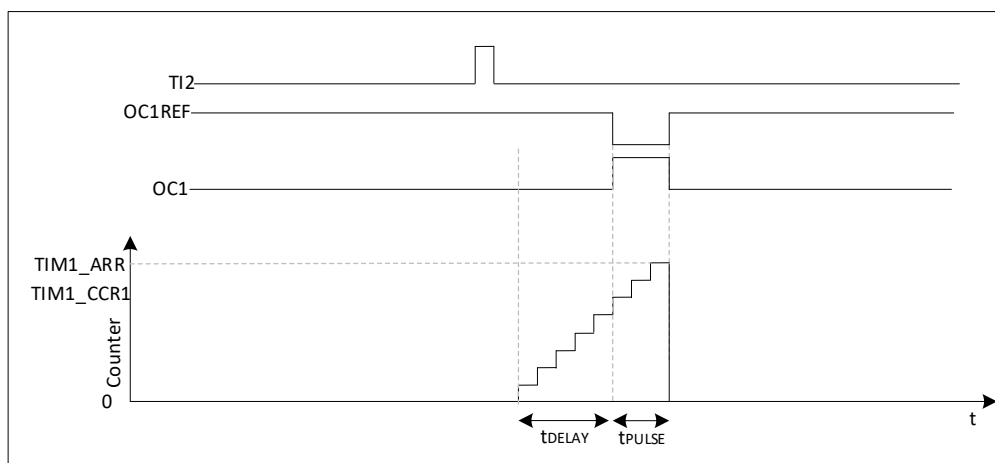


Figure 25-26 Example of Single Pulse Mode

For example, when a rising edge needs to be detected on the TI2 input pin, a positive pulse of length t_{PULSE} is generated on OC1 after a delay of t_{DELAY} .

Using TI2FP2 as trigger 1:

- Set CC2S=01 in the TIMx_CCMR1 register to map TI2FP2 to TI2.
- Set CC2P=0 in the TIMx_CCER register to enable TI2FP2 to detect rising edges.
- Set TS=110 in the TIMx_SMCR register, and TI2FP2 serves as the trigger (TRGI) for the slave mode controller.

- Set SMS=110 (trigger mode) in the TIMx_SMCR register, and TI2FP2 is used to start the counter.

The waveform of OPM is determined by the value written to the compare register (taking into account the clock frequency and counter prescaler).

- t_{DELAY} is defined by the value in the TIMx_CCR1 register.
- t_{PULSE} is defined by the difference between the auto-reload value and the compare value ($TIMx_ARR - TIMx_CCR1$).
- Assume that a waveform transition from 0 to 1 is to be generated upon a compare match, and a transition from 1 to 0 is to be generated when the counter reaches the preload value. First, set OC1M=111 in the TIMx_CCMR1 register to enter PWM mode 2. Optionally enable the preload register as needed: set OC1PE=1 in TIMx_CCMR1 and ARPE in the TIMx_CR1 register. Then, fill the compare value in the TIMx_CCR1 register and the auto-reload value in the TIMx_ARR register, set the UG bit to generate an update event, and wait for an external trigger event on TI2. In this case, CC1P=0.

In this example, the DIR and CMS bits in the TIMx_CR1 register should be set low.

Since only one pulse is needed, OPM=1 must be set in the TIMx_CR1 register to stop counting at the next update event (when the counter rolls over from the auto-reload value to 0).

Special case: OCx fast enable:

In one-pulse mode, the edge detection logic on the Tix input pin sets the CEN bit to start the counter. The comparison between the counter and the compare value then generates the output transition. However, these operations require a certain number of clock cycles, which limits the achievable minimum delay t_{DELAY} .

To output waveforms with minimal delay, set the OCxFE bit in the TIMx_CCMRx register; in this case, OCxREF (and OCx) directly responds to the stimulus without relying on the comparison result, and the output waveform is the same as during a comparison match. OCxFE is only effective when the channel is configured in PWM1 or PWM2 mode.

25.5. TIMx timer and external trigger synchronization (TIM15 only)

The TIMx timer can be synchronized with an external trigger in various modes: reset mode, gated mode, and trigger mode.

25.5.1. Slave mode: Reset mode

When a trigger input event occurs, the counter and its prescaler can be reinitialized; meanwhile, if the URS bit in TIMx_CR1 register is low, an update event UEV is also generated; then all the preload registers (TIMx_ARR, TIMx_CCRx) are updated.

In the following example, a rising edge on the TI1 input causes the up-counter to be cleared:

- Configure channel 1 to detect the rising edge of TI1. Configure the bandwidth of the input filter (in this case, no filter is needed, so keep IC1F=0000). The capture prescaler is not used in trigger operations, so no configuration is needed. The CC1S bit only selects the input capture source, i.e., CC1S=01 in TIMx_CCMR1 register. Set CC1P=0 in TIMx_CCER register to determine the polarity (only rising edge detection).
- Set SMS=100 in TIMx_SMCR register to configure the timer in reset mode; set TS=101 in TIMx_SMCR register to select TI1 as the input source.
- Set CEN=1 in the TIMx_CR1 register to start the counter.

The counter starts counting based on the internal clock, then operates normally until a rising edge appears on TI1; at this point, the counter is cleared and starts counting again from 0. Meanwhile, the trigger flag (TIF bit in TIMx_SR register) is set, generating an interrupt request or a DMA request based on the settings of TIE (interrupt enable) bit and TDE (DMA enable) bit in TIMx_DIER register.

The figure below shows the operation when the auto-reload register TIMx_ARR=0x36. The delay between the rising edge of TI1 and the actual reset of the counter depends on the resynchronization circuit at the TI1 input.

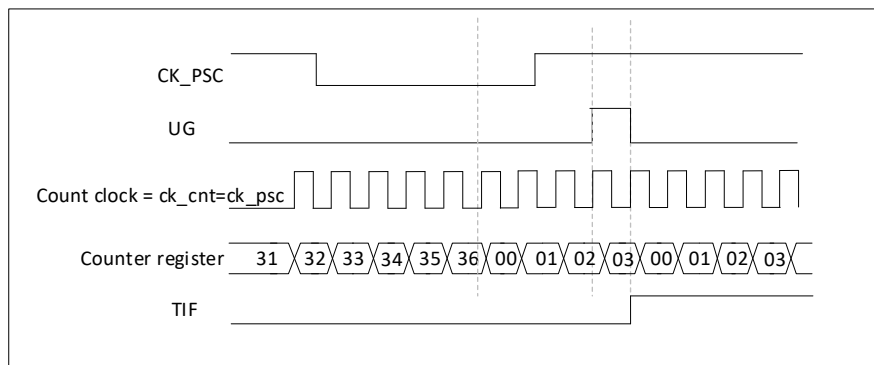


Figure 25-27 Control circuit in reset mode

25.5.2. Slave mode: Gated mode

The counter is enabled based on the level of the selected input.

In the following example, the counter counts up only when TI1 is low:

- Configure channel 1 to detect the low level on TI1. Configure the input filter bandwidth (in this example, no filtering is needed, so keep IC1F=0000). The capture prescaler is not used in trigger operations, so no configuration is needed. The CC1S bit is used to select the input capture source; set CC1S=01 in the TIMx_CCMR1 register. Set CC1P=1 in the TIMx_CCER register to determine polarity (only detects low level).
- Set SMS=101 in the TIMx_SMCR register to configure the timer in gated mode; set TS=101 in the TIMx_SMCR register to select TI1 as the input source.
- Set CEN=1 in the TIMx_CR1 register to start the counter. In gated mode, if CEN=0, the counter cannot start regardless of the trigger input level.

As long as TI1 is low, the counter counts based on the internal clock; it stops counting once TI1 goes high. The TIF flag in TIMx_SR is set whenever the counter starts or stops.

The delay between the rising edge of TI1 and the actual stopping of the counter depends on the resynchronization circuit at the TI1 input.

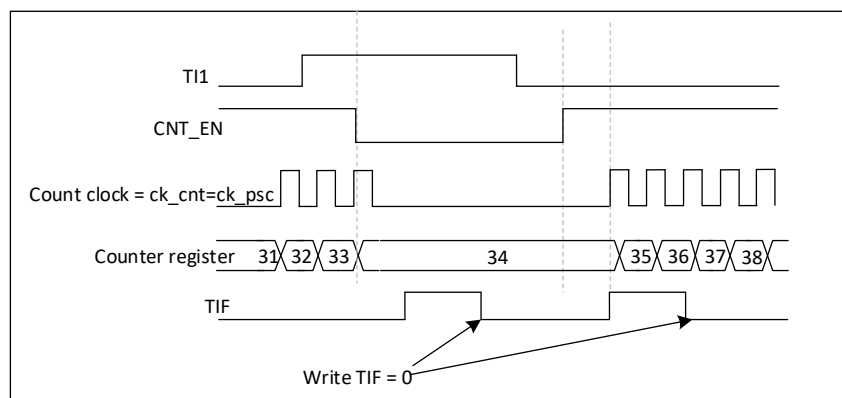


Figure 25-28 Control circuit in gated mode

25.5.3. Slave mode: Trigger mode

The selected event at the input enables the counter.

In the following example, the counter starts counting up on the rising edge of the TI2 input:

- Configure channel 2 to detect the rising edge of TI2. Configure the input filter bandwidth (in this example, no filter is needed; keep IC2F=0000). The capture prescaler is not used in trigger operations and does not require configuration. The CC2S bit is only used to select the input capture source; set CC2S=01 in the TIMx_CCMR1 register. Set CC2P=1 in the TIMx_CCER register to determine polarity (only detects low level).
- Set SMS=110 in the TIMx_SMCR register to configure the timer in trigger mode; set TS=110 in the TIMx_SMCR register to select TI2 as the input source.

When a rising edge occurs on TI2, the counter starts counting driven by the internal clock, and the TIF flag is set.

The delay between the rising edge of TI2 and the counter starting to count depends on the resynchronization circuit at the TI2 input.

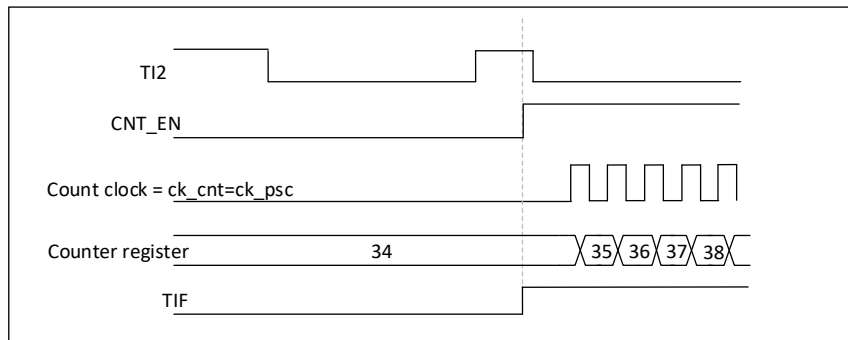


Figure 25-29 Control circuit in trigger mode

25.5.4. Slave mode: External clock mode 2 + Trigger mode

External clock mode 2 can be used with another slave mode (except external clock mode 1 and encoder mode). Here, the ETR signal is used as the input for the external clock, and another input can be selected as the trigger input in reset mode, gated mode, or trigger mode. It is not recommended to use the TS bit of the TIMx_SMCR register to select ETR as TRGI.

In the following example, once a rising edge occurs on TI1, the counter increments once at every rising edge of ETR:

1. Configure external trigger input circuit via TIMx_SMCR register:
 - ETF=0000: No filtering
 - ETPS=00: No prescaler used
 - ETP=0: Detect the rising edge of ETR. Set ECE=1 to enable external clock mode 2.
2. Configure Channel 1 as follows to detect rising edge of TI:
 - IC1F=0000: No filtering
 - The capture prescaler is not used in trigger operation, no configuration is needed
 - Set CC1S=01 in TIMx_CCMR1 register to select the input capture source
 - Set CC1P=0 in the TIMx_CCER register to determine the polarity (only rising edges are detected).
3. Set SMS=110 in TIMx_SMCR register to configure the timer in trigger mode. Set TS=101 in the TIMx_SMCR register to select TI1 as the input source.

When a rising edge occurs on TI1, the TIF flag is set, and the counter starts counting at the rising edge of ETR.

The delay between the rising edge of the ETR signal and the actual counter reset depends on the resynchronization circuit at the ETRP input.

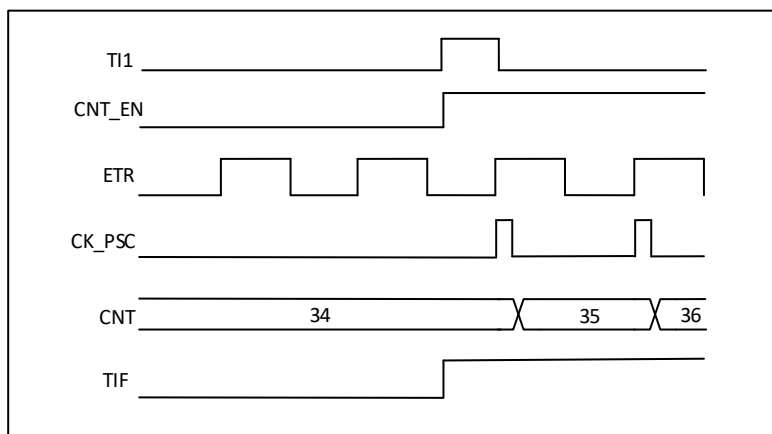


Figure 25-30 Control circuit in external clock mode 2 + trigger mode

25.5.5. TIM and external trigger synchronization

The TIMx timer can be synchronized with an external trigger in various modes: reset mode, gated mode, and trigger mode.

Slave mode: Reset mode

When a trigger input event occurs, the counter and its prescaler can be reinitialized; meanwhile, if the URS bit in TIMx_CR1 register is low, an update event UEV is also generated; then all the preload registers (TIMx_ARR, TIMx_CCRx) are updated.

In the following example, a rising edge on the TI1 input causes the up-counter to be cleared:

- Configure channel 1 to detect the rising edge of TI1. Configure the bandwidth of the input filter (in this case, no filter is needed, so keep IC1F=0000). The capture prescaler is not used in trigger operations, so no configuration is needed. The CC1S bit only selects the input capture source, i.e., CC1S=01 in TIMx_CCMR1 register. Set CC1P=0 in TIMx_CCER register to determine the polarity (only rising edge detection).
- Set SMS=100 in TIMx_SMCR register to configure the timer in reset mode; set TS=101 in TIMx_SMCR register to select TI1 as the input source.
- Set CEN=1 in the TIMx_CR1 register to start the counter.

The counter starts counting based on the internal clock, then operates normally until a rising edge appears on TI1; at this point, the counter is cleared and starts counting again from 0. Meanwhile, the trigger flag (TIF bit in TIMx_SR register) is set, generating an interrupt request or a DMA request based on the settings of TIE (interrupt enable) bit and TDE (DMA enable) bit in TIMx_DIER register.

The figure below shows the operation when the auto-reload register TIMx_ARR=0x36. The delay between the rising edge of TI1 and the actual reset of the counter depends on the resynchronization circuit at the TI1 input.

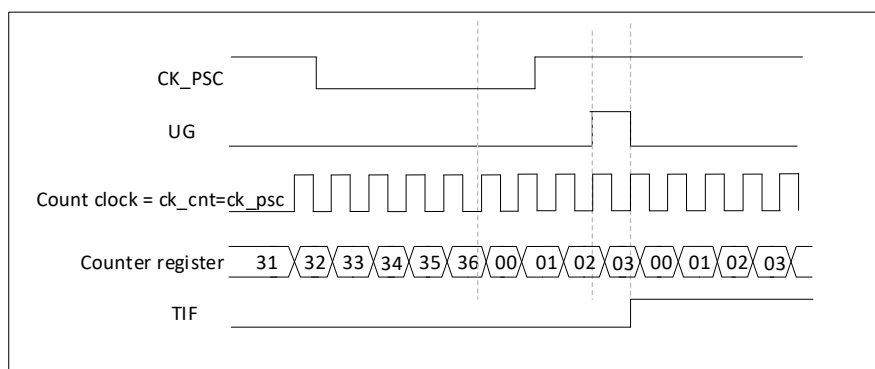


Figure 25-31 Control circuit in reset mode

Slave mode: Gated mode

The counter is enabled based on the level of the selected input.

In the following example, the counter counts up only when TI1 is low:

- Configure channel 1 to detect the low level on TI1. Configure the input filter bandwidth (in this example, no filtering is needed, so keep IC1F=0000). The capture prescaler is not used in trigger operations, so no configuration is needed. The CC1S bit is used to select the input capture source; set CC1S=01 in the TIMx_CCMR1 register. Set CC1P=1 in the TIMx_CCER register to determine polarity (only detects low level).
- Set SMS=101 in the TIMx_SMCR register to configure the timer in gated mode; set TS=101 in the TIMx_SMCR register to select TI1 as the input source.
- Set CEN=1 in the TIMx_CR1 register to start the counter. In gated mode, if CEN=0, the counter cannot start regardless of the trigger input level.

As long as TI1 is low, the counter counts based on the internal clock; it stops counting once TI1 goes high. The TIF flag in TIMx_SR is set whenever the counter starts or stops.

The delay between the rising edge of TI1 and the actual stopping of the counter depends on the resynchronization circuit at the TI1 input.

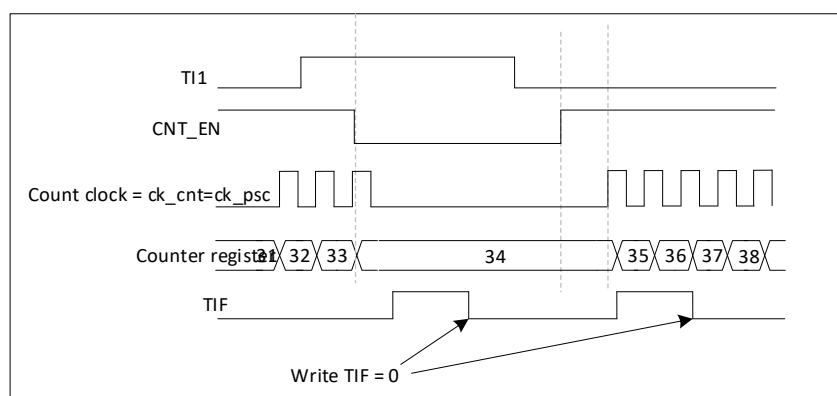


Figure 25-32 Control circuit in gated mode

The selected event at the input enables the counter.

In the following example, the counter starts counting up on the rising edge of the TI2 input:

- c) Configure channel 2 to detect the rising edge of TI2. Configure the input filter bandwidth (in this example, no filter is needed; keep IC2F=0000). The capture prescaler is not used in trigger operations and does not require configuration. The CC2S bit is only used to select the input capture source; set

CC2S=01 in the TIMx_CCMR1 register. Set CC2P=1 in the TIMx_CCER register to determine polarity (only detects low level).

- d) Set SMS=110 in the TIMx_SMCR register to configure the timer in trigger mode; set TS=110 in the TIMx_SMCR register to select TI2 as the input source.

When a rising edge occurs on TI2, the counter starts counting driven by the internal clock, and the TIF flag is set. The delay between the rising edge of TI2 and the counter starting to count depends on the re-synchronization circuit at the TI2 input.

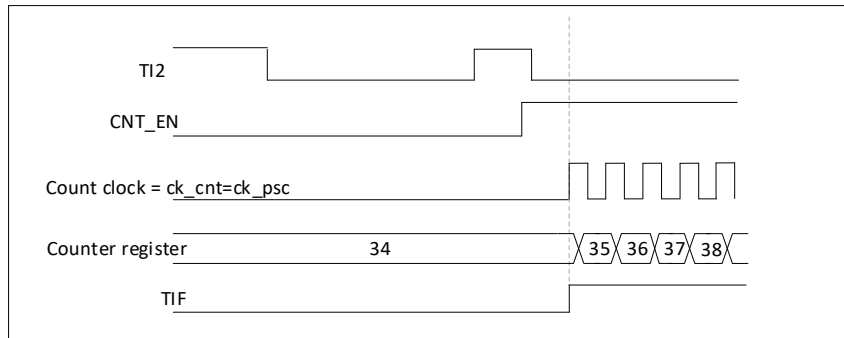


Figure 25-33 Control circuit in gated mode

Slave mode: External clock mode 2 + trigger mode

External clock mode 2 can be used with another slave mode (except external clock mode 1 and encoder mode). Here, the ETR signal is used as the input for the external clock, and another input can be selected as the trigger input in reset mode, gated mode, or trigger mode. It is not recommended to use the TS bit of the TIMx_SMCR register to select ETR as TRGI.

In the following example, once a rising edge occurs on TI1, the counter increments once at every rising edge of ETR:

- Configure the external trigger input circuit via the TIMx_SMCR register:
 - ETF=0000: No filtering.
 - ETPS=00: No prescaler used.
 - ETP=0: Detect the rising edge of ETR. Set ECE=1 to enable external clock mode 2.
- Configure channel 1 as follows to detect the rising edge of TI:
 - IC1F=0000: No filtering.
 - The capture prescaler is not used in trigger operation and does not need to be configured.
 - Set CC1S=01 in the TIMx_CCMR1 register to select the input capture source.
 - Set CC1P=0 in the TIMx_CCER register to determine the polarity (only rising edges are detected).
- Set SMS=110 in the TIMx_SMCR register to configure the timer in trigger mode. Set TS=101 in the TIMx_SMCR register to select TI1 as the input source.

When a rising edge occurs on TI1, the TIF flag is set, and the counter starts counting at the rising edge of ETR. The delay between the rising edge of the ETR signal and the actual counter reset depends on the resynchronization circuit at the ETRP input.

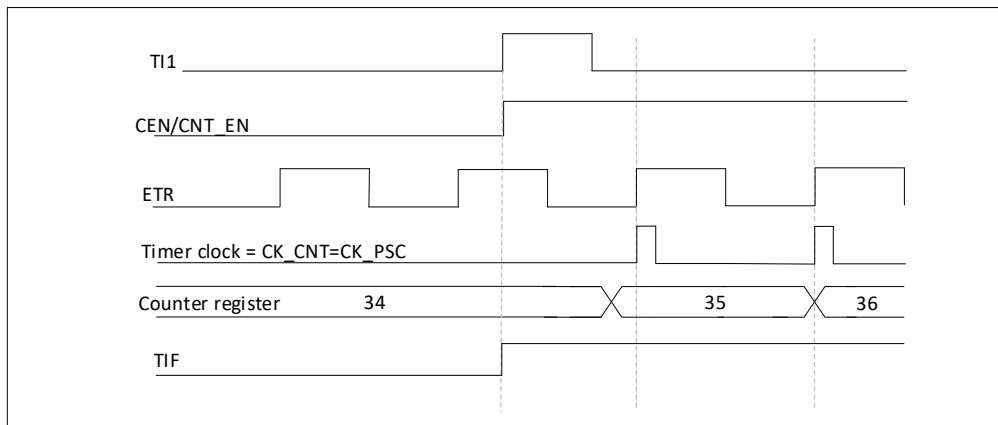


Figure 25-34 Control circuit in external clock mode 2 + trigger mode

25.6. Timer synchronization (TIM15 only)

All TIMx timers are internally connected for timer synchronization or chaining. When a timer is in master mode, it can reset, start, stop, or provide clock to another timer's counter in slave mode.

The figure below shows an overview of the trigger selection and master mode selection module.

25.6.1. Using one timer as a prescaler for another timer

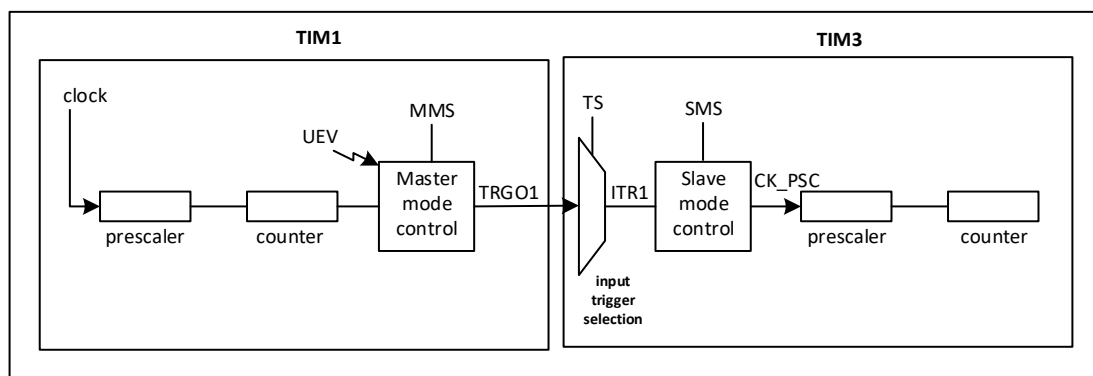


Figure 25-35 Example of Master/Slave Timers

For example, Timer 1 can be configured as a prescaler for Timer 2. Refer to Figure 4-48 and perform the following operations:

- Configure Timer 1 as master mode, which can output a periodic trigger signal at each update event UEV. When MMS='010' in TIM15_16_17_CR2 register, a rising edge signal is output on TRGO1 whenever an update event occurs.
- Connect Timer 1's TRGO1 output to Timer 2, set TS='000' in TIM2_SMCR register, and configure Timer 2 as a slave using ITR1 as internal trigger.
- Then set the slave mode controller to external clock mode 1 (SMS=111 in TIM2_SMCR register); this allows Timer 2 to be driven by Timer 1's periodic rising edge (i.e., Timer 1's counter overflow) signal.
- Finally, the respective CEN bits (in TIMx_CR1 register) must be set to start both timers separately.

Note: If OCx has been selected as Timer 1's trigger output (MMS=1xx), its rising edge is used to drive Timer 2's counter.

25.6.2. Using one timer to enable another timer

In this example, Timer 2's enable is controlled by Timer 1's output compare. Refer to the connection in Figure 4-48. Timer 2 counts the divided internal clock only when Timer 1's OC1REF is high. The clock frequency of both timers is obtained by dividing CK_INT by 3 using the prescaler ($f_{CK_CNT}=f_{CK_INT}/3$).

- Configure Timer 1 as master mode, outputting its output compare reference signal (OC1REF) as trigger output (MMS=100 in TIM15_16_17_CR2 register).
- Configure the OC1REF waveform of Timer 1 (TIM15_16_17_CCMR1 register).
- Configure Timer 2 to receive input trigger from Timer 1 (TS=000 in TIM2_SMCR register).
- Configure Timer 2 in gated mode (SMS=101 in the TIM2_SMCR register)
- Set CEN=1 in the TIM2_CR1 register to enable Timer 2.
- Set CEN=1 in the TIM15_16_17_CR1 register to start Timer 1.

Note: The clock of Timer 2 is not synchronized with the clock of Timer 1; this mode only affects the enable signal of Timer 2's counter.

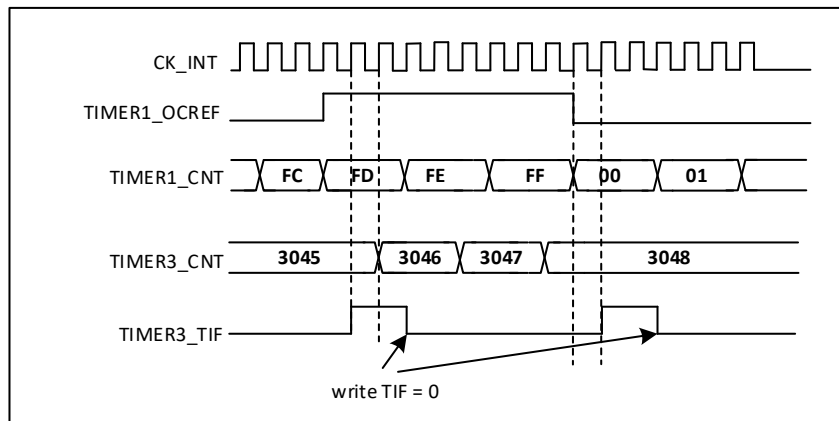


Figure 25-36 Timer 1's OC1REF controls Timer 2

In the example of Figure 4-47, before Timer 2 is started, their counters and prescalers are not initialized, so they start counting from their current values. Both timers can be reset before starting Timer 1 to make them start from given values, i.e., writing any desired value to the timer counters. Writing to the UG bit of the TIMx_EGR register resets the timer.

In the next example, Timer 1 and Timer 2 need to be synchronized. Timer 1 is in master mode and starts from 0, while Timer 2 is in slave mode and starts from 0xE7; both timers have the same prescaler coefficient. Writing '0' to the CEN bit of TIM15_16_17_CR1 will disable Timer 1, and Timer 2 will subsequently stop.

- Configure Timer 1 in master mode, sending the output compare 1 reference signal (OC1REF) as the trigger output (MMS=100 in the TIM15_16_17_CR2 register).
- Configure the OC1REF waveform of Timer 1 (TIM15_16_17_CCMR1 register).
- Configure Timer 2 to receive input trigger from Timer 1 (TS=000 in TIM2_SMCR register).
- Configure Timer 2 in gated mode (SMS=101 in the TIM2_SMCR register)
- Set UG='1' in the TIM15_16_17_EGR register to reset Timer 1.
- Set UG='1' in the TIM2_EGR register to reset Timer 2.

- Write '0xE7' to the counter of Timer 2 (TIM2_CNT) to initialize it to 0xE7.
- Set CEN='1' in the TIM2_CR1 register to enable Timer 2.
- Set CEN='1' in the TIM15_16_17_CR1 register to start Timer 1.
- Set CEN='0' in the TIM15_16_17_CR1 register to stop Timer 1.

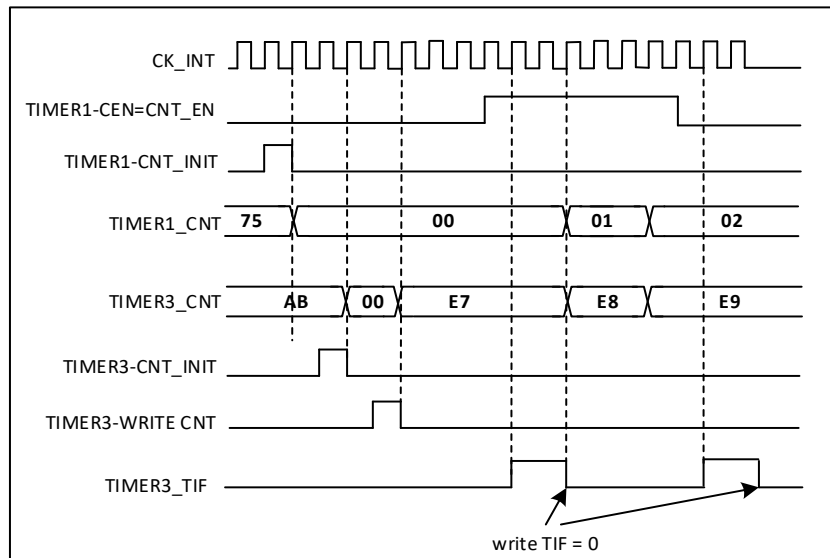


Figure 25-37 Timer 2 can be controlled by enabling Timer 1

25.6.3. Using one timer to start another timer

In this example, the update event of Timer 1 is used to enable Timer 2. Refer to the connection in Figure 4-47. Once Timer 1 generates an update event, Timer 2 starts counting from its current value (which may be non-zero) based on the divided internal clock. Upon receiving the trigger signal, the CEN bit of Timer 2 is automatically set to '1', and the counter starts counting until '0' is written to the CEN bit of the TIM2_CR1 register. Both timers' clock frequencies are derived from CK_INT divided by 3 via the prescaler ($f_{CK_CNT}=f_{CK_INT}/3$).

- Configure Timer 1 as master, sending its update event (UEV) as trigger output (MMS=010 in TIM15_16_17_CR2 register).
- Configure Timer 1's period (TIM15_16_17_ARR register).
- Configure Timer 2 to receive input trigger from Timer 1 (TS=000 in TIM2_SMCR register).
- Configure Timer 2 in trigger mode (SMS=110 in TIM2_SMCR register)
- Set CEN=1 in TIM15_16_17_CR1 register to start Timer 1.

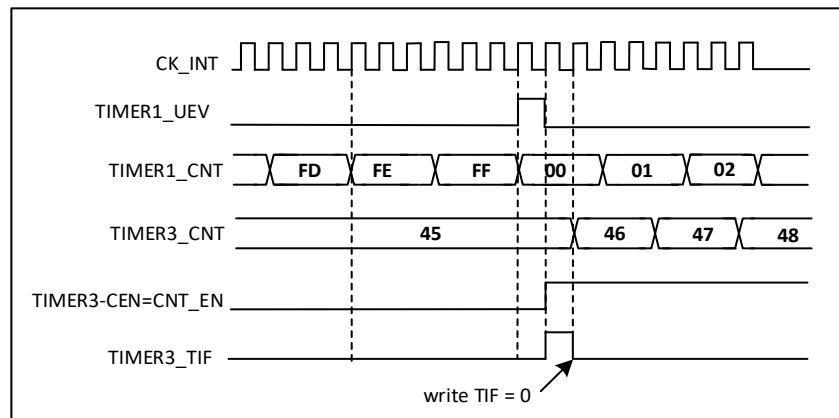


Figure 25-38 Using Timer 1's update event to trigger Timer 2

In the previous example, both counters can be initialized before starting the count. Shows the operation when using trigger mode instead of gated mode (SMS=110 in TIM2_SMCR register) with the same configuration as in case 0.

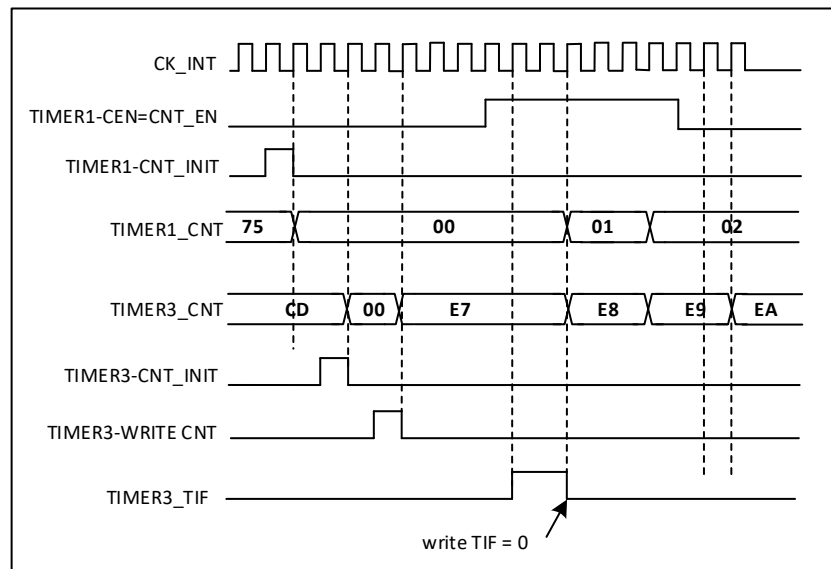


Figure 25-39 Using Timer 1's enable to trigger Timer 2

25.6.4. Using an external trigger to synchronously start two timers

In this example, Timer 1 is enabled when a rising edge occurs on its TI1 input, and enabling Timer 1 simultaneously enables Timer 2. See Figure 25-40. To ensure counter alignment, Timer 1 must be configured in master/slave mode (slave for TI1, master for Timer 2):

- Configure Timer 1 as master, sending its enable as trigger output (MMS=001 in TIM15_16_17_CR2 register).
- Configure Timer 1 as slave, receiving input trigger from TI1 (TS=100 in TIM15_16_17_SMCR register).
- Configure Timer 1 in trigger mode (SMS=110 in TIM15_16_17_SMCR register).
- Configure Timer 1 in master/slave mode (MSM=1 in TIM15_16_17_SMCR register).
- Configure Timer 2 to receive input trigger from Timer 1 (TS=000 in TIM2_SMCR register).

- Configure Timer 2 in trigger mode (SMS=110 in TIM2_SMCR register).

When a rising edge occurs on TI1 of Timer 1, both timers start counting synchronously with the internal clock, and both TIF flags are set simultaneously.

Note: In this example, both timers are initialized before startup (with corresponding UG bits set), both counters start from 0, but an offset can be inserted between timers by writing to either counter register (TIMx_CNT). The figure below shows a delay between CNT_EN and CK_PSC of timer 1 in master/slave mode.

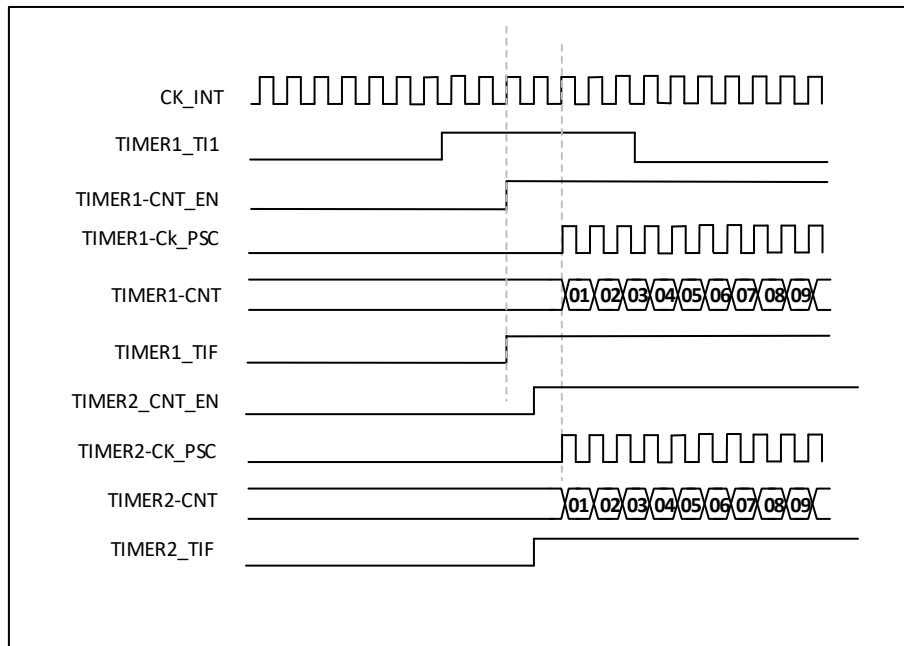


Figure 25-40 Using timer 1's TI1 input to trigger timer 1 and timer 2

25.6.5. Debug mode

When the chip enters debug mode, the TIMx counter can continue to operate normally or stop working according to the DBG_TIMx_STOP setting in the DBG module.

25.7. TIM15 register description

0x4001 4800 - 0x4001 4BFF TIM17

0x4001 4400 - 0x4001 47FF TIM16

0x4001 4000 - 0x4001 43FF TIM15

25.7.1. TIM15 control register 1 (TIMx_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	CKD[1:0]		ARPE	Res	Res	Res	OPM	URS	UDIS	CEN
-	-	-	-	-	-	RW		RW	-	-	-	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15: 10	Reserved	-	-	Reserved
9: 8	CKD[1:0]	RW	00	Clock division factor These two bits define the division ratio between the timer clock (CK_INT) frequency, dead-time, and the sampling clock used by the dead-time generator and digital filter (ETR, Tix). 00: tDTS = tCK_INT 01: tDTS = 2 x tCK_INT 10: tDTS = 4 x tCK_INT 11: Reserved, do not use this configuration.
7	ARPE	RW	0	Auto-reload preload enable bit 0: TIMx_ARR register is not buffered 1: TIMx_ARR register is buffered
6: 4	Reserved	-	-	Reserved
3	OPM	RW	0	One-pulse mode 0: The counter does not stop when an update event occurs 1: The counter stops at the next update event (clearing the CEN bit).
2	URS	RW	0	Update request source Software uses this bit to select the source of UEV event 0: If update interrupt or DMA request is enabled, any of the following events generates an update interrupt or DMA request: – Counter overflow/underflow – Setting the UG bit – Update generated by the slave mode controller 1: If update interrupt or DMA request is enabled, only counter overflow/underflow generates an update interrupt or DMA request. break or DMA request
1	UDIS	RW	0	Disable update The software uses this bit to enable/disable the generation of UEV events 0: Enable UEV. An update (UEV) event is generated by any of the following events: – Counter overflow/underflow – Setting the UG bit – Update generated by the slave mode controller The buffered registers are loaded with their preload values. 1: Disable UEV. No update event is generated, and the shadow registers (ARR, PSC, CCRx) retain their values.

				If the UG bit is set or a hardware reset is issued by the slave mode controller, the counter and prescaler are re-initialized.
0	CEN	RW	0	<p>Enable counter</p> <p>0: Disable counter</p> <p>1: Enable counter</p> <p>Note: After the CEN bit is set in the software, the external clock, gated mode, and encoder mode can operate. The trigger mode can automatically set the CEN bit through hardware.</p>

25.7.2. TIM15 control register 2 (TIMx_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res					OIS2	OIS1N	OIS1	Res	MMS[2:0]			CCDS	CCUS	Res	CCPC
-					RW	RW	RW	-	RW	RW	RW	RW	RW	-	RW

Bit	Name	R/W	Reset Value	Function
15: 11	Reserved	-	-	Reserved
10	OIS2	RW	0	Output idle state 2 (OC2 output). Refer to OIS1 bit
9	OIS1N	RW	0	<p>Output idle state 1 (OC1N output).</p> <p>0: When MOE=0, OC1N=0 after dead time</p> <p>1: When MOE=0, OC1N=1 after dead time</p> <p>Note: This bit cannot be modified after LOCK (TIMx_BDTR register) levels 1, 2, or 3 have been set.</p>
8	OIS1	RW	0	<p>Output idle state 1 (OC1 output).</p> <p>0: When MOE=0, if OC1N is implemented, OC1=0 after dead time</p> <p>1: When MOE=0, if OC1N is implemented, OC1=1 after dead time</p> <p>Note: This bit cannot be modified after LOCK (TIMx_BDTR register) levels 1, 2, or 3 have been set.</p>
7	Reserved	-	-	Reserved
6: 4	MMS[2:0]	RW	000	<p>Master mode selection</p> <p>These 3 bits are used to select the synchronization information (TRGO) sent to slave timers in master mode. Possible combinations are as follows:</p>

Bit	Name	R/W	Reset Value	Function
				<p>000: Reset – The UG bit of TIMx_EGR register is used as trigger output (TRGO). If the trigger input (slave mode controller in reset mode) generates a reset, the signal on TRGO will have a delay relative to the actual reset. There will be a delay.</p> <p>001: Enable – The counter enable signal CNT_EN is used as trigger output (TRGO). Sometimes it is necessary To start multiple timers simultaneously or control a window of slave timers. The counter enable signal is generated by the logical OR of the CEN control bit and the trigger input signal in gated mode. When the counter enable signal is controlled</p> <p>There will be a delay on TRGO during trigger input unless master/slave mode is selected (see description of MSM bit in TIMx_SMCR register).</p> <p>010: Update – The update event is selected as trigger input (TRGO). For example, the clock of a master timer can be used as a prescaler for a slave timer.</p> <p>011: Compare pulse – Upon a capture or compare match occurrence, when the CC1IF flag is to be set (i.e. it is already high), the trigger output sends a positive pulse (TRGO).</p> <p>100: Compare – The OC1REF signal is used as trigger output (TRGO).</p> <p>101: Compare – The OC2REF signal is used as a trigger output (TRGO).</p> <p>Note:</p> <ol style="list-style-type: none"> 1. The clocks of the slave timer and ADC must first be enabled to receive the signal from the master timer and should not be changed during reception. 2. If the master and slave timers are not on the same bus, the master mode should be configured to a width that can be sampled by the slave timer.
3	CCDS	RW	0	<p>Capture/Compare DMA selection</p> <p>0: When a CCx event occurs, send a DMA request for CCx.</p> <p>1: When an update event occurs, send a DMA request for CCx.</p>
2	CCUS	RW	0	Capture/Compare Control Update Selection

Bit	Name	R/W	Reset Value	Function
				<p>0: If the capture/compare control bits are preloaded (CCPC=1), they can only be updated by setting the COM bit.</p> <p>1: If the capture/compare control bits are preloaded (CCPC=1), they can be updated by setting the COM bit or on TRGI a rising edge updates them.</p> <p>Note: This bit only affects channels with complementary outputs.</p>
1	Reserved	-	-	Reserved
0	CCPC	RW	0	<p>Capture/Compare Preload Control Bit</p> <p>0: CCxE, CCxNE, and OCxM bits are not preloaded.</p> <p>1: CCxE, CCxNE, and OCxM bits are preloaded; after setting this bit, they are only updated when the COM bit is set.</p> <p>Note: This bit only affects channels with complementary outputs.</p>

25.7.3. TIM15 slave mode control register (TIMx_SMCR)

Address offset: 0x08

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res								MSM	TS[2:0]			Res	SMS[2:0]		
-								RW	RW			-	RW		

Bit	Name	R/W	Reset Value	Function
15: 8	Reserved	-	-	Reserved
7	MSM	RW	0	<p>Master/Slave mode</p> <p>0: No effect</p> <p>1: The event on the trigger input (TRGI) is delayed to allow synchronization between the current timer (via TRGO) and its current timer and slave timer (via TRGO). This is very useful when multiple timers need to be synchronized to a single external event.</p>
6: 4	TS[2:0]	RW	000	<p>Trigger selection. These 3 bits select the trigger input used to synchronize the counter.</p> <p>000: TIM2 (ITR0)</p> <p>001: TIM3 (ITR1)</p> <p>010: TIM16 (ITR2)</p>

				<p>011: TIM17 (ITR3)</p> <p>100: Edge detector of TI1 (TI1F_ED)</p> <p>101: Filtered timer input 1 (TI1FP1)</p> <p>110: Filtered timer input 2 (TI2FP2)</p> <p>111: External trigger input (ETRF)</p> <p>Note: To avoid false edge detection during signal transitions, these bits must be modified when they are unused.</p>
3	Reserved	-	-	Reserved
2: 0	SMS[2:0]	RW	000	<p>Slave mode selection. When an external signal is selected, the active edge of the trigger signal (TRGI) depends on the polarity of the selected external input (see descriptions of the input control register and control register).</p> <p>000: Slave mode disabled</p> <p>If CEN=1, the prescaler is directly driven by the internal clock.</p> <p>100: Reset mode</p> <p>The rising edge of the selected trigger input (TRGI) reinitializes the counter and generates a register update signal.</p> <p>101: Gated mode</p> <p>When the trigger input (TRGI) is high, the counter clock is enabled. Once the trigger input goes low, the counter stops (but does not reset). Both the start and stop of the counter are controlled.</p> <p>110: Trigger mode</p> <p>The counter starts (but does not reset) on the rising edge of trigger input TRGI. Only the counter start is controlled.</p> <p>111: External clock mode 1</p> <p>The rising edge of the selected trigger input (TRGI) drives the counter.</p> <p>Note: If TI1F_EN is selected as the trigger input (TS=100), do not use gated mode. This is because TI1F_ED outputs a pulse every time TI1F changes, whereas gated mode is meant to check the level of the trigger input.</p> <p>Note: In encoder mode, do not use uev as trgo output signal (i.e., mms cannot be configured as 010)</p>

25-41 TIM15 internal trigger connection

Slave TIM	ITR0 (TS=000)	ITR1 (TS=001)	ITR2 (TS=010)	ITR3 (TS=011)
TIM15	TIM2_TRGO	TIM3_TRGO	TIM16_OC1	TIM17_OC

25.7.4. TIM15 DMA/Interrupt Enable Register (TIMx_DIER)

Address offset: 0x0C

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	TDE	COMDE	Res		CC2DE	CC1DE	UDE	BIE	TIE	COMIE	Res		CC2IE	CC1IE	UIE
-	RW	RW	-	-	RW						-	-	RW		

Bit	Name	R/W	Reset Value	Function
15	Reserved	-	-	Reserved
14	TDE	RW	0	TDE: Enable triggering DMA request 0: Disable triggering DMA request 1: Enable triggering DMA request
13	COMDE	RW	0	COMDE: Enable DMA request for COM 0: Disable DMA request for COM 1: Enable DMA request for COM
11: 12	Reserved	-	-	Reserved
10	CC2DE	RW	0	CC2DE: Enable capture/compare 2 DMA request 0: Disable capture/compare 2 DMA request 1: Enable capture/compare 2 DMA request
9	CC1DE	RW	0	CC1DE: Enable capture/compare 1 DMA request 0: Disable capture/compare 1 DMA request 1: Enable capture/compare 1 DMA request
8	UDE	RW	0	UDE: Enable update DMA request 0: Disable update DMA request 1: Enable update DMA request
7	BIE	RW	0	BIE: Enable brake interrupt 0: Disable brake interrupt 1: Enable brake interrupt
6	TIE	RW	0	TIE: Enable interrupt triggering 0: Disable interrupt triggering 1: Enable interrupt triggering
5	COMIE	RW	0	COMIE: COM interrupt enable 0: COM interrupt disabled 1: COM interrupt enabled
4: 3	Reserved	-	-	Reserved
2	CC2IE	RW	0	CC2IE: Capture/compare 2 interrupt enable 0: Capture/compare 2 interrupt disabled 1: Capture/compare 2 interrupt enabled

1	CC1IE	RW	0	CC1IE: Capture/Compare 1 interrupt enable 0: Capture/Compare 1 interrupt disabled 1: Capture/Compare 1 interrupt enabled
0	UIE	RW	0	UIE: Update interrupt enable 0: Update interrupt disabled 1: Update interrupt enabled

25.7.5. TIM15 status register (TIMx_SR)

Address offset: 0x010

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	CC2OF	CC1OF	Res	BIF	TIF	COMIF	Res	Res	CC2IF	CC1F	UIF
-	-	-	-	-	RC_W0		-	RC_W0			-	-	RC_W0		

Bit	Name	R/W	Reset Value	Function
15: 11	Reserved	-	-	Reserved
10	CC2OF	RC_W0	0	Capture/Compare 2 Overcapture Flag Refer to CC1OF description
9	CC1OF	RC_W0	0	Capture/Compare 1 over-capture flag This flag can only be set to 1 by hardware when the corresponding channel is configured as input capture. Writing 0 can clear this bit. 0: No over-capture generated; 1: When CC1OF is set to 1, the counter value has been captured into the TIMx_CCR1 register.
8	Reserved	-	-	Reserved
7	BIF	RC_W0	0	Brake interrupt flag Once the brake input is valid, this bit is set to 1 by hardware. If the brake input is invalid, this bit can be cleared to 0 by software. 0: No brake event generated; 1: A valid level is detected on the brake input.
6	TIF	RC_W0	0	Trigger interrupt flag When a trigger event occurs (when the slave mode controller is in modes other than gated mode, a valid edge is detected at the TRGI input terminal This bit is set to 1 by hardware when a valid edge or any edge in gated mode is detected. It is cleared by software. 0: No trigger event occurred; 1: Trigger interrupt pending

Bit	Name	R/W	Reset Value	Function
5	COMIF	RC_W0	0	<p>COM interrupt flag</p> <p>This bit is set to 1 by hardware once a COM event occurs (when CcxE, CcxNE, OCxM have been updated). It is cleared by software.</p> <p>0: No COM event generated; 1: COM interrupt pending</p>
4: 3	Reserved	-	-	Reserved
2	CC2IF	RC_W0	0	<p>Capture/Compare 2 interrupt flag</p> <p>Refer to CC1IF description</p>
1	CC1IF	RC_W0	0	<p>Capture/Compare 1 interrupt flag</p> <p>If channel CC1 is configured in output mode: This bit is set by hardware when the counter value matches the compare value, except in center-aligned mode (refer to the TIMx_CR1 register the CMS bit of the register). It is cleared by software. 0: No match occurs; 1: The value of TIMx_CNT matches the value of TIMx_CCR1.</p> <p>If channel CC1 is configured in input mode: This bit is set by hardware when a capture event occurs. It is cleared by software or by reading TIMx_CCR1. 0: No input capture occurred; 1: An input capture occurred and the counter value has been loaded into TIMx_CCR1 (a edge with the selected polarity was detected on IC1).</p> <p>Note: This bit is also set when CEN is enabled.</p>
0	UIF	RC_W0	0	<p>Update interrupt flag</p> <p>This bit is set by hardware when an update event occurs. It is cleared by software.</p> <p>0: No update event generated; 1: An update event is pending. This bit is set by hardware when the register is updated:</p> <ul style="list-style-type: none"> – If UDIS=0 in TIMx_CR1 register, an update event is generated when REP_CNT=0 (counter overflow); – If UDIS=0 and URS=0 in TIMx_CR1 register, an update event is generated when UG=1 in TIMx_EGR register (software reinitializes CNT);

Bit	Name	R/W	Reset Value	Function
				<p>– If UDIS=0 and URS=0 in TIMx_CR1 register, an update event is generated when CNT is reinitialized by a trigger event.</p> <p>(Refer to Slave Mode Control Register (TIMx_SMCR))</p>

25.7.6. TIM15 event generation register (TIMx_EGR)

Address offset: 0x14

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	BG	TG	COMG	Res	Res	CC2G	CC1G	UG
-	-	-	-	-	-	-	-	W	W	W	-	-	W	W	W

Bit	Name	R/W	Reset Value	Function
15: 8	Reserved	-	-	Reserved
7	BG	W	0	<p>Generate brake event</p> <p>This bit is set to 1 by software to generate a brake event and is automatically cleared to 0 by hardware.</p> <p>0: No action;</p> <p>1: Generates a brake event. When MOE=0 and BIF=1, if the corresponding interrupt and DMA are enabled, the corresponding interrupt and DMA will be generated.</p>
6	TG	W	0	<p>Generate trigger event</p> <p>This bit is set to 1 by software to generate a trigger event and is automatically cleared to 0 by hardware.</p> <p>0: No action;</p> <p>1: When TIF of TIMx_SR register equals 1, if the corresponding interrupt and DMA are enabled, it will generate the corresponding interrupt and DMA.</p>
5	COMG	W	0	<p>Capture/compare event generates a control update.</p> <p>This bit is set to 1 by software and automatically cleared to 0 by hardware.</p> <p>0: No action;</p> <p>1: When CCPC=1, allows updating CcxNE, CcxNE, OCxM bits.</p> <p>Note: This bit is only valid for channels with complementary outputs.</p>
4: 3	Reserved	-	-	Reserved
2	CC2G	W	0	Generate capture/compare 2 event

Bit	Name	R/W	Reset Value	Function
				Refer to CC1G description
1	CC1G	W	0	<p>Generate capture/compare 1 event</p> <p>This bit is set to 1 by software to generate a capture/compare event and is automatically cleared by hardware.</p> <p>0: No action;</p> <p>1: Generate a capture/compare event on channel CC1:</p> <p>If channel CC1 is configured as output:</p> <p>Set CC1IF=1, and if the corresponding interrupt and DMA are enabled, the corresponding interrupt and DMA are generated.</p> <p>If channel CC1 is configured as input:</p> <p>The current counter value is captured into the TIMx_CCR1 register, CC1IF is set to 1, and if the corresponding interrupt and DMA are enabled, the corresponding interrupt and DMA are generated. If CC1IF is already set to 1, then set CC1OF=1.</p>
0	UG	W	0	<p>Generate an update event. This bit is set by software and cleared automatically by hardware.</p> <p>0: No action;</p> <p>1: Reinitialize the counter and generate an update event.</p> <p>Note: The prescaler counter is also cleared (but the prescaler</p> <p>The coefficient remains unchanged).</p>

25.7.7. TIM15 Capture/Compare Mode Register 1 (TIMx_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	OC2M[2: 0]			OC2PE	CO2FE	CC2S[1: 0]		Res	OC1M[2: 0]			OC1PE	OC1FE	CC1S[1: 0]	
IC2F[3: 0]				IC2PSC[1: 0]		0]		IC1F[3: 0]				IC1PSC[1: 0]		0]	
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Output compare mode:

Bit	Name	R/W	Reset Value	Function
15	Reserved	-	-	Reserved
14: 12	OC2M[2: 0]	RW	0	Output Compare 2 Mode Selection
11	OC2PE	RW	0	Output Compare 2 Preload Enable
10	OC2FE	RW	0	Output Compare 2 Fast Enable
9: 8	CC2S[1: 0]	RW	00	Capture/Compare 2 selection

Bit	Name	R/W	Reset Value	Function
				<p>This bit defines the channel direction (input/output) and the selection of the input pin:</p> <p>00: CC2 channel is configured as output;</p> <p>01: CC2 channel is configured as input, IC2 is mapped on TI2;</p> <p>10: CC2 channel is configured as input, IC2 is mapped on TI1;</p> <p>11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode only works when the internal trigger input is selected</p> <p>(Selected by the TS bit in the TIMx_SMCR register).</p> <p>Note: CC2S is writable only when the channel is closed (CC2E=0 in the TIMx_CCER register).</p>
7	Reserved	-	-	Reserved
6: 4	OC1M[2: 0]	RW	00	<p>Output compare 1 mode</p> <p>These 3 bits define the action of output reference signal OC1REF, which determines the values of OC1 and OC1N. OC1REF is active high, while the active levels of OC1 and OC1N depend on CC1P and CC1NP bits.</p> <p>000: Freeze. The comparison between output compare register TIMx_CCR1 and counter TIMx_CNT has no effect on OC1REF.</p> <p>;</p> <p>001: Set channel 1 to active level upon match. When the value of the counter TIMx_CNT matches capture/compare register 1 (TIMx_CCR1), force OC1REF high.</p> <p>010: Set channel 1 to inactive level upon match. When the value of the counter TIMx_CNT matches the capture/compare register</p> <p>Force OC1REF to low when TIMx_CNT matches TIMx_CCR1.</p> <p>011: Toggle. Toggle OC1REF level when TIMx_CCR1 = TIMx_CNT.</p> <p>100: Force to inactive level. Force OC1REF to low.</p> <p>101: Force to active level. Force OC1REF to high.</p> <p>110: PWM Mode 1 - During up-counting, channel 1 is at active level when TIMx_CNT < TIMx_CCR1, otherwise it is at inactive level;</p>

Bit	Name	R/W	Reset Value	Function
				<p>111: PWM Mode 2 - During up-counting, channel 1 is at inactive level when TIMx_CNT < TIMx_CCR1, otherwise it is at active level;</p> <p>Note 1: Once the LOCK level is set to 3 (LOCK bit in the TIMx_BDTR register) and CC1S=00 (the channel is configured as an output), this bit cannot be modified.</p> <p>Note 2: In PWM mode 1 or PWM mode 2, the OC1REF level only changes when the compare result changes or when switching from freeze mode to PWM mode in output compare mode.</p>
3	OC1PE	RW	0	<p>Output Compare 1 Preload Enable</p> <p>0: Disable the preload function of the TIMx_CCR1 register. The TIMx_CCR1 register can be written at any time, and the new value takes effect immediately.</p> <p>1: Enable the preload function of the TIMx_CCR1 register. Read and write operations only affect the preload register. The preload value of TIMx_CCR1 is loaded into the current register when an update event occurs.</p> <p>Note 1: Once the LOCK level is set to 3 (LOCK bit in the TIMx_BDTR register) and CC1S=00 (the channel is configured as an output), this bit cannot be modified.</p> <p>Note 2: In single-pulse mode only, PWM mode can be used without acknowledging the preload register; otherwise, its behavior is undefined.</p>
2	OC1FE	RW	0	<p>Output Compare 1 Fast Enable</p> <p>This bit is used to speed up the CC output's response to trigger input events.</p> <p>0: CC1 operates normally based on the counter and CCR1 value, even if the trigger is enabled. When there is an active edge at the trigger input, the minimum delay to activate the CC1 output is 5 clock cycles.</p> <p>1: The active edge input to the trigger acts as if a compare match has occurred. Therefore, OC is set as the comparison level and</p> <p>Independent of the comparison result. The delay between the active edge of the sampling trigger and the CC1 output is reduced to 3 clock cycles.</p> <p>OCFE only takes effect when the channel is configured in PWM1 or PWM2 mode.</p>

Bit	Name	R/W	Reset Value	Function
1: 0	CC1S[1: 0]	RW	00	<p>Capture/Compare 1 selection.</p> <p>These 2 bits define the channel direction (input/output) and the selection of the input pin:</p> <p>00: CC1 channel is configured as output;</p> <p>01: CC1 channel is configured as input, IC1 is mapped to TI1;</p> <p>10: CC1 channel is configured as input, IC1 is mapped to TI2;</p> <p>11: CC1 channel is configured as input, IC1 is mapped to TRC. This mode only works when the internal trigger input is selected</p> <p>(Selected by the TS bit in the TIMx_SMCR register).</p> <p>Note: CC1S is writable only when the channel is turned off (CC1E=0 in the TIMx_CCER register).</p>

Input Capture mode:

Bit	Name	R/W	Reset Value	Function
15: 12	IC2F	RW	0	Input capture 2 filter
11: 10	IC2PSC[1: 0]	RW	0	Capture/Compare 2 prescaler
9: 8	CC2S[1: 0]	RW	0	<p>Capture/Compare 2 selection</p> <p>These 2 bits define the channel direction (input/output) and the selection of the input pin:</p> <p>00: CC2 channel is configured as output;</p> <p>01: CC2 channel is configured as input, IC2 is mapped on TI2;</p> <p>10: CC2 channel is configured as input, IC2 is mapped on TI1;</p> <p>11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode only works when the internal trigger input is selected</p> <p>(Selected by the TS bit in the TIMx_SMCR register).</p> <p>Note: CC2S is writable only when the channel is closed (CC2E=0 in the TIMx_CCER register).</p>
7: 4	IC1F[3: 0]	RW	0000	<p>Input capture 1 filter</p> <p>These bits define the sampling frequency and digital filter length for TI1 input. The digital filter consists of an event counter,</p> <p>It generates an output transition after recording N events:</p> <p>0000: No filter, sampling at fDTS</p> <p>0001: fSAMPLING=fCK_INT, N=2</p>

Bit	Name	R/W	Reset Value	Function
				0010: fSAMPLING=fCK_INT, N=4 0011: fSAMPLING=fCK_INT, N=8 0100: fSAMPLING=fDTS/2, N=6 0101: fSAMPLING=fDTS/2, N=8 0110: fSAMPLING=fDTS/4, N=6 0111: fSAMPLING=fDTS/4, N=8 1000: fSAMPLING=fDTS/8, N=6 1001: fSAMPLING=fDTS/8, N=8 1010: fSAMPLING=fDTS/16, N=5 1011: fSAMPLING=fDTS/16, N=6 1100: fSAMPLING=fDTS/16, N=8 1101: fSAMPLING=fDTS/32, N=5 1110: fSAMPLING=fDTS/32, N=6 1111: fSAMPLING=fDTS/32, N=8
3: 2	IC1PSC[1: 0]	RW	00	Capture/Compare 1 Prescaler 00: No prescaler, every edge detected on the capture input triggers a capture; 01: Capture triggered every 2 events; 10: Capture triggered every 4 events; 11: Capture triggered every 8 events.
1: 0	CC1S[1: 0]	RW	00	CC1S[1:0]: Capture/Compare 1 Selection. These 2 bits define the channel direction (input/output) and the selection of the input pin: 00: CC1 channel is configured as output; 01: CC1 channel is configured as input, IC1 is mapped to TI1; 10: CC1 channel is configured as input, IC1 is mapped to TI2; 11: CC1 channel is configured as input, IC1 is mapped to TRC. This mode only works when the internal trigger input is selected (Selected by the TS bit in the TIMx_SMCR register). Note: CC1S is writable only when the channel is turned off (CC1E=0 in the TIMx_CCER register).

25.7.8. TIM15 Capture/Compare Enable Register (TIMx_CCER)

Address offset: 0x20

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res								CC2NP	Res	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
-								RW	-	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15: 8	Reserved	-	-	Reserved
7	CC2NP	RW	0	Capture/Compare 2 complementary output polarity. Refer to the description of CC1NP.
6	Reserved	-	-	Reserved
5	CC2P	RW	0	Capture/Compare 2 output polarity. Refer to the description of CC1P.
4	CC2E	RW	0	Capture/Compare 2 output enable. Refer to the description of CC1E.
3	CC1NP	RW	0	Capture/Compare 1 complementary output polarity 0: OC1N active high 1: OC1N active low Note: Once the LOCK level (LOCK bit in TIMx_BDTR register) is set to 3 or 2 and CC1S=00 (channel configured as output), this bit cannot be modified.
2	CC1NE	RW	0	Capture/Compare 1 complementary output enable 0: OC1N output is disabled 1: OC1N signal is output to the corresponding output pin When the CC1 channel is configured as an output, the OC1N output level is determined by the combined settings of MOE, OSSI, OSSR, OIS1, OIS1N, CC1E, and CC1NE bits, as shown in the table below.
1	CC1P	RW	0	Capture/Compare 1 output polarity CC1 channel configured as output: 0: OC1 active high 1: OC1 active low CC1 channel configured as input: CC1NP/CC1P bits select the polarity of TI1FP1 and TI2FP1 as trigger or capture signals. 00: Not inverted / rising edge: TIxFP1 is active on rising edge (capture, trigger in reset mode, external clock or trigger mode); TIxFP1 is not inverted (gated mode, encoder mode). 01: Inverted / falling edge: TIxFP1 is active on falling edge (capture, trigger in reset mode, external clock or trigger mode);

Bit	Name	R/W	Reset Value	Function
				<p>TIxFP1 is inverted (gated mode, encoder mode).</p> <p>10: Reserved, do not use this configuration.</p> <p>11: Not inverted / both edges</p> <p>TIxFP1 is active on both rising and falling edges (capture, trigger in reset mode, external clock or trigger mode);</p> <p>TIxFP1 is not inverted (gated mode). This configuration cannot be applied in encoder mode.</p> <p>Note:</p> <p>1. For complementary output channels, this bit is preloaded. If the CCPC bit in the TIMx_CR2 register is set, the actual valid CC1P bit will only load the preload value when a COM event occurs.</p> <p>2. Once the LOCK level (LOCK bits in TIMx_BDTR register) is set to 3 or 2, this bit cannot be modified</p>
0	CC1E	RW	0	<p>Capture/Compare 1 output enable</p> <p>0: Capture mode disabled/OC1 output disabled</p> <p>1: Capture mode enabled/OC1 signal output to the corresponding output pin</p> <p>When the CC1 channel is configured as output, the OC1 output level is determined by the combined effect of MOE, OSSI, OSSR, OIS1, OIS1N, CC1E, and CC1NE bits, as shown in the table below.</p> <p>Note:</p> <p>For complementary output channels, this bit is preloaded. If the CCPC bit in the TIMx_CR2 register is set, the actual effective bit of CC1E will only load the preload value when a COM event occurs.</p>

Table 25-1 Output control of complementary OCx and OCxN channels with interrupt capability

Control bits					Output state	
MOE	OSSI	OSSR	CCxE	CCxNE	OCx output state	OCxN output state
1	X	0	0	0	Output disabled (disconnected from timer), OCx=0, OCx_EN=0	Output disabled (disconnected from timer), OCxN=0, OCxN_EN=0
		0	0	1	Output disabled (disconnected from timer), OCx=0, OCx_EN=0	OCxREF + Polarity, OCxN=OCxREF XOR CCxNP, OCxN_EN=1

Control bits					Output state	
MOE	OSSI	OSSR	CCxE	CCxNE	OCx output state	OCxN output state
		0	1	0	OCxREF + Polarity OCx=OCREF XOR CCxP, OCx_EN=1	Output disabled (disconnected from timer), OCxN=0, OCxN_EN=0
		0	1	1	OCREF + Polarity + dead-time, OCx_EN=1	Complement of OCREF (not OCREF) + Polarity + dead-time, OCxN_EN=1
		1	0	0	Output disabled (disconnected from timer), OCx=CCxP, OCx_EN=0	Output disabled (disconnected from timer), OCxN=CCxNP, OCxN_EN=0
		1	0	1	Output disabled (disconnected from timer), OCx=CCxP, OCx_EN=1	OCxREF + Polarity OCxN=OCxREF xor CCxNP, OCxN_EN=1
		1	1	0	OCxREF + Polarity OCx=OCxREF xor CCxP, OCx_EN=1	Off state (output enabled and inactive level), OCxN=CCxNP, OCxN_EN=1
		1	1	1	OCREF + Polarity + dead-time OCx_EN=1	Complementary of OC- REF (not OCREF) + po- larity + dead-time OCN_EN=1
0	0	X	0	0	Output disabled (disconnected from timer)	
	0		0	1		
	0		1	0		
	0		1	1		
	1		0	0		
	1		0	1	Off state (output enabled and at inactive level) Asynchronous: OCx=CCxP, OCx_EN=1, OCxN=CCxNP, OCxN_EN=1 If the clock exists: After a dead time, assuming OISx and OISxN do not both correspond to the active level of OCx and OCxN, OCx=OISx and OCxN=OISxN.	
	1		1	0		
	1		1	1		

If neither output of a channel is used (CCxE = CCxNE = 0), then OISx, OISxN, CCxP, and CCxNP must all be cleared.

Note: The status of the external I/O pins connected to complementary OCx and OCxN channels depends on the state of OCx and OCxN channels, as well as GPIO and AFIO registers.

25.7.9. TIM15 counter (TIMx_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
15: 0	CNT[15:0]	RW	0	Counter value

25.7.10. TIM15 prescaler (TIMx_PSC)

Address offset: 0x28

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
15: 0	PSC[15:0]	RW	0	<p>Prescaler value</p> <p>The clock frequency of the counter (CK_CNT) equals $f_{CK_PSC} / (PSC[15:0] + 1)$.</p> <p>PSC contains the value to be loaded into the current pre-scaler register when an update event occurs; the update event includes the counter</p> <p>Cleared by the UG bit of TIM_EGR or cleared by the slave controller operating in reset mode.</p>

25.7.11. TIM15 auto-reload register (TIMx_ARR)

Address offset: 0x2c

Reset value: 0x0000 FFFF

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

Bit	Name	R/W	Reset Value	Function
15: 0	ARR[15:0]	RW	FFFF	<p>Auto-reload value</p> <p>ARR contains the value to be loaded into the actual auto-reload register.</p>

				When the auto-reload value is empty, the counter does not work.
--	--	--	--	---

25.7.12. TIM15 repetition counter register (TIMx_RCR)

Address offset: 0x30

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
15: 8	Reserved	-	-	Reserved
7: 0	REP[7:0]	RW	0	<p>Value of the period counter</p> <p>With the preload function enabled, these bits allow users to set the update rate of the compare registers.</p> <p>If the preload function is enabled, these bits allow users to set the update rate of the compare registers (i.e., periodically transferring from the preload register to the current register); if update interrupts are enabled, this will also affect the rate at which update interrupts are generated.</p> <p>Each time the down-counter REP_CNT reaches 0, an update event is generated, and the counter REP_CNT restarts counting from the REP value. Since REP_CNT only reloads the REP value when a period update event U_RC occurs, the new value written to the TIMx_RCR register will only take effect during the next period update event.</p> <p>This means that in PWM mode, (REP+1) corresponds to:</p> <ul style="list-style-type: none"> - In edge-aligned mode, the number of PWM cycles;

25.7.13. TIM15 Capture/Compare Register 1 (TIMx_CCR1)

Address offset: 0x34

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
15: 0	CCR1[15: 0]	RW/RO	0	<p>Capture/Compare 1 value</p> <p>If the CC1 channel is configured as output:</p>

				<p>CCR1 contains the value loaded into the current capture/compare 1 register (preload value).</p> <p>If the preload feature is not selected in the TIMx_CCMR1 register (OC1PE bit), it is always loaded into the current register.</p> <p>Otherwise, this preload value is loaded into the current capture/compare 1 register only when an update event occurs.</p> <p>The current capture/compare register contains the value to be compared with the counter TIMx_CNT and outputs a signal on the OC1 port.</p> <p>If the CC1 channel is configured as input:</p> <p>CCR1 contains the counter value transferred by the last input capture 1 event (IC1).</p>
--	--	--	--	--

25.7.14. TIM15 Capture/Compare Register 2 (TIMx_CCR2)

Address offset: 0x38

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
15: 0	CCR2[15: 0]	RW/RO	0	<p>Capture/compare 2 value</p> <p>If the CC2 channel is configured as output:</p> <p>CCR2 contains the value loaded into the current capture/compare 2 register (preload value).</p> <p>If the preload feature is not selected in the TIMx_CCMR1 register (OC2PE bit), it is always loaded into the current register.</p> <p>Otherwise, this preload value is loaded into the current capture/compare 2 register only when an update event occurs.</p> <p>The current capture/compare register contains the value to be compared with the counter TIMx_CNT, and outputs a signal on the OC port.</p> <p>If the CC2 channel is configured as input:</p> <p>CCR2 contains the counter value transferred from the last input capture 2 event (IC2).</p>

25.7.15. TIM15 brake and dead-time register (TIMx_BDTR)

Address offset: 0x44

Reset value: 0x0000 0000

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

Note: According to the lock settings, AOE, BKP, BKE, OSSI, OSSR, and DTG[7:0] bits can be write-protected. It is necessary to configure them during the first write operation to the TIMx_BDTR register.

Bit	Name	R/W	Reset Value	Function
15	MOE	RW	0	<p>Main output enable</p> <p>Once the brake input is active, this bit is asynchronously cleared by hardware. Can be cleared by software or automatically set according to the AOE bit value. It is only valid for output-configured channels.</p> <p>0: Disable OC and OCN output or force to idle state; 1: If the corresponding enable bit is set (CCxE, CCxNE bits in TIMx_CCER register), OC and OCN output.</p>
14	AOE	RW	0	<p>Automatic output enable</p> <p>0: MOE can only be set by software; 1: MOE can be set by software or automatically set at the next update event (if brake input is inactive).</p> <p>Note: Once the LOCK level (LOCK bit in TIMx_BDTR register) is set to 1, this bit cannot be modified.</p>
13	BKP	RW	0	<p>Brake input polarity</p> <p>0: Brake input active low; 1: Brake input active high.</p> <p>Note: Once the LOCK level (LOCK bit in TIMx_BDTR register) is set to 1, this bit cannot be modified.</p> <p>Any write operation to this bit takes effect after a delay of one APB clock cycle.</p>
12	BKE	RW	0	<p>Brake function enable</p> <p>0: Disable brake input; 1: Enable brake input.</p> <p>Note: Once the LOCK level (LOCK bit in TIMx_BDTR register) is set to 1, this bit cannot be modified.</p> <p>Any write operation to this bit takes effect after a delay of one APB clock cycle.</p>

Bit	Name	R/W	Reset Value	Function
11	OSSR	RW	0	<p>Off-state selection in run mode</p> <p>This bit is used when MOE=1 and the channel is in complementary output mode. The OSSR bit does not exist in timers without complementary outputs.</p> <p>Refer to the detailed description of OC/OCN enable (Capture/Compare Enable Register (TIMx_CCER)).</p> <p>0: When the timer is not active, OC/OCN output is disabled (OC/OCN enabled output signal = 0);</p> <p>1: When the timer is not working, once CCxE=1 or CCxNE=1, enable OC/OCN output and output invalid level.</p> <p>OC/OCN enabled output signal = 1.</p> <p>Note: Once the LOCK level (LOCK bit in the TIMx_BDTR register) is set to 2, this bit cannot be modified.</p>
10	OSSI	RW	0	<p>Off-state selection in idle mode</p> <p>This bit is used when MOE=0 and the channel is configured as output.</p> <p>Refer to the detailed description of OC/OCN enable (Capture/Compare Enable Register (TIMx_CCER)).</p> <p>0: When the timer is not active, OC/OCN output is disabled (OC/OCN enabled output signal = 0);</p> <p>1: When the timer is not working, once CCxE=1 or CCxNE=1, the OC/OCN first outputs its idle level.</p> <p>OC/OCN enabled output signal = 1.</p> <p>Note: Once the LOCK level (LOCK bit in the TIMx_BDTR register) is set to 2, this bit cannot be modified.</p>
9: 8	LOCK[1:0]	RW	00	<p>Lock setup</p> <p>This bit provides write protection to prevent software errors.</p> <p>00: Lock off, no write protection for the register;</p> <p>01: Lock level 1, the DTG/BKE/BKP/AOE bits in the TIMx_BDTR register and the OISx/OISxN bits in the TIMx_CR2 register cannot be written;</p> <p>10: Lock level 2, bits in lock level 1 cannot be written, nor can the CC polarity bits (once the corresponding channel is set as output via the CCxS bit, the CCxP/CCNxP bits in the TIMx_CCER register) or the OSSR/OSSI bits;</p> <p>11: Lock level 3, bits in lock level 2 cannot be written, nor can the CC control bits (once the corresponding channel</p>

Bit	Name	R/W	Reset Value	Function
				is set as output via the CCxS bit, the OCxM/OCxPE bits in the TIMx_CCMRx register); Note: After a system reset, the LOCK bit can only be written once. Once written to the TIMx_BDTR register, its contents are frozen until To reset.
7: 0	DTG[7:0]	RW	0000 0000	Dead-time generator setup These bits define the dead-time duration inserted between complementary outputs. Assuming DT represents its duration: DTG[7:5]=0xx => DT=DTG[7:0] × Tdtg, Tdtg = TDTS; DTG[7:5]=10x => DT=(64+DTG[5:0]) × Tdtg, Tdtg = 2 × TDTS; DTG[7:5]=110 => DT=(32+DTG[4:0]) × Tdtg, Tdtg = 8 × TDTS; DTG[7:5]=111 => DT=(32+DTG[4:0]) × Tdtg, Tdtg = 16 × TDTS; Example: If TDTS = 125ns (8MHz), the possible dead-time is: 0 to 15875ns, if the step time is 125ns; 16us to 31750ns, if the step time is 250ns; 32us to 63us, if the step time is 1us; 64us to 126us, if the step time is 2us; Note: Once the LOCK level (LOCK bits in TIMx_BDTR register) is set to 1, 2, or 3, these bits cannot be modified. Modified.

25.7.16. TIM15 DMA control register (TIMx_DCR)

Address offset: 0x48

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	DBL[4:0]					Res			DBA[4:0]				
-	-	-	RW	RW	RW	RW	RW	-	-	-	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15: 13	Reserved	-	-	Reserved
12: 8	DBL[4:0]	RW	0 0000	DMA continuous transfer length

Bit	Name	R/W	Reset Value	Function
				<p>These bits define the transfer length of DMA in continuous mode (when reading from or writing to the address of the TIMx_DMAR register , the timer performs one continuous transfer), i.e.: defines the number of bytes to be transferred:</p> <p>00000: 1 transfer 00001: 2 transfers 00010: 3 transfers 10001: 18 transfers</p> <p>Example: We consider such a transfer: DBL=7, DBA=TIM2_CR1</p> <p>- If DBL=7 and DBA=TIM2_CR1 represents the address of the data to be transferred, the transfer address is given by:</p> <p>(TIMx_CR1 address) + DBA + (DMA index), where DMA index = DBL</p> <p>Where (TIMx_CR1 address) + DBA + 7 gives the address from which data will be written or read, so the data transfer will occur across 7 registers starting from the address (TIMx_CR1 address) + DBA.</p> <p>Depending on the DMA data length setting, the following scenarios may occur:</p> <p>- If data is set to half-word (16-bit), the data will be transferred to all 7 registers.</p> <p>- If data is set to byte, the data 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, for timers, the user must specify the data width for DMA transfers.</p>
7: 5	Reserved	-	-	Reserved
4: 0	DBA[4:0]	RW	0 0000	<p>DBA[4:0]: DMA base address</p> <p>These bits define the base address for DMA in continuous mode (when reading or writing to the TIMx_DMAR register address), DBA is defined as the offset from the base address of the TIMx_CR1 register:</p> <p>00000: TIMx_CR1,</p>

Bit	Name	R/W	Reset Value	Function
				00001: TIMx_CR2, 00010: TIMx_SMCR,

25.7.17. DMA address for TIM15 continuous mode (TIMx_DMAR)

Address offset: 0x4C

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
15: 0	DMAB[31:0]	RW	0	<p>DMA Continuous Transfer Register</p> <p>Reading or writing to the TIMx_DMAR register results in access to the register at the following address:</p> <p>TIMx_CR1 address + (DBA + DMA pointer) x4, where:</p> <p>"TIMx_CR1 address" is the address of Control Register 1;</p> <p>"DBA" is the base address defined in the TIMx_DCR register;</p> <p>The "DMA pointer" is an offset automatically controlled by the DMA, which depends on the DBL defined in the TIMx_DCR register.</p>

Note: When using the DMA continuous transfer function, the value of the CNDTR register in the corresponding DMA channel must correspond to the DBL value in the TIMx_DCR register; otherwise, it will not function properly.

25.8. TIM16_17 Register Description

0x4001 4800 - 0x4001 4BFF TIM17

0x4001 4400 - 0x4001 47FF TIM16

0x4001 4000 - 0x4001 43FF TIM15

25.8.1. TIM16_17 Control Register 1 (TIMx_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	CKD[1:0]		ARPE	Res	Res	Res	OPM	URS	UDIS	CEN
-	-	-	-	-	-	RW		RW	-	-	-	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15: 10	Reserved	-	-	Reserved

Bit	Name	R/W	Reset Value	Function
9: 8	CKD[1:0]	RW	00	Clock division factor These two bits define the division ratio between the timer clock (CK_INT) frequency, dead-time, and the sampling clock used by the dead-time generator and digital filter (ETR, Tix). 00: tDTS = tCK_INT 01: tDTS = 2 x tCK_INT 10: tDTS = 4 x tCK_INT 11: Reserved, do not use this configuration.
7	ARPE	RW	0	Auto-reload preload enable bit 0: TIMx_ARR register is not buffered 1: TIMx_ARR register is buffered
6: 4	Reserved	-	-	Reserved
3	OPM	RW	0	One-pulse mode 0: The counter does not stop when an update event occurs 1: The counter stops at the next update event (clearing the CEN bit).
2	URS	RW	0	Update request source Software uses this bit to select the source of UEV event 0: If update interrupt or DMA request is enabled, any of the following events generates an update interrupt or DMA request: – Counter overflow/underflow – Setting the UG bit – Update generated by the slave mode controller 1: If update interrupt or DMA request is enabled, only counter overflow/underflow generates an update interrupt or DMA request. break or DMA request
1	UDIS	RW	0	Disable update The software uses this bit to enable/disable the generation of UEV events 0: Enable UEV. An update (UEV) event is generated by any of the following events: – Counter overflow/underflow – Setting the UG bit – Update generated by the slave mode controller The buffered registers are loaded with their preload values. 1: Disable UEV. No update event is generated, and the shadow registers (ARR, PSC, CCRx) retain their values.

Bit	Name	R/W	Reset Value	Function
				If the UG bit is set or a hardware reset is issued by the slave mode controller, the counter and prescaler are re-initialized.
0	CEN	RW	0	<p>Enable counter</p> <p>0: Disable counter</p> <p>1: Enable counter</p> <p>Note: After the CEN bit is set in the software, the external clock, gated mode, and encoder mode can operate. The trigger mode can automatically set the CEN bit through hardware.</p>

25.8.2. TIM16_17 Control Register 2 (TIMx_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res						OIS1N	OIS1	Res				CCDS	CCUS	Res	CCPC
-						RW	RW	-				Rw	RW	-	RW

Bit	Name	R/W	Reset Value	Function
15: 10	Reserved	-	-	Reserved
9	OIS1N	RW	0	<p>Output idle state 1 (OC1N output).</p> <p>0: When MOE=0, OC1N=0 after dead time</p> <p>1: When MOE=0, OC1N=1 after dead time</p> <p>Note: This bit cannot be modified after LOCK (TIMx_BDTR register) levels 1, 2, or 3 have been set.</p>
8	OIS1	RW	0	<p>Output idle state 1 (OC1 output).</p> <p>0: When MOE=0, if OC1N is implemented, OC1=0 after dead time</p> <p>1: When MOE=0, if OC1N is implemented, OC1=1 after dead time</p> <p>Note: This bit cannot be modified after LOCK (TIMx_BDTR register) levels 1, 2, or 3 have been set.</p>
7: 4	Reserved	-	-	Reserved
3	CCDS	RW	0	<p>Capture/Compare DMA selection</p> <p>0: When a CCx event occurs, send a DMA request for CCx.</p> <p>1: When an update event occurs, send a DMA request for CCx.</p>
2	CCUS	RW	0	Capture/Compare Control Update Selection

Bit	Name	R/W	Reset Value	Function
				<p>0: If the capture/compare control bits are preloaded (CCPC=1), they can only be updated by setting the COM bit.</p> <p>1: If the capture/compare control bits are preloaded (CCPC=1), they can be updated by setting the COM bit or on TRGI a rising edge updates them.</p> <p>Note: This bit only affects channels with complementary outputs.</p>
1	Reserved	-	-	Reserved
0	CCPC	RW	0	<p>Capture/Compare Preload Control Bit</p> <p>0: CCxE, CCxNE, and OCxM bits are not preloaded.</p> <p>1: CCxE, CCxNE, and OCxM bits are preloaded; after setting this bit, they are only updated when the COM bit is set.</p> <p>Note: This bit only affects channels with complementary outputs.</p>

25.8.3. TIM16_17 DMA/Interrupt Enable Register (TIMx_DIER)

Address offset: 0x0C

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res						CC1DE	UDE	BIE	Res	COMIE	Res	Res	Res	CC1IE	UIE
-						RW	RW	RW	-	RW	-	-	-	RW	RW

Bit	Name	R/W	Reset Value	Function
15: 10	Reserved	-	-	Reserved
9	CC1DE	RW	0	<p>CC1DE: Enable capture/compare 1 DMA request</p> <p>0: Disable capture/compare 1 DMA request</p> <p>1: Enable capture/compare 1 DMA request</p>
8	UDE	RW	0	<p>UDE: Enable update DMA request</p> <p>0: Disable update DMA request</p> <p>1: Enable update DMA request</p>
7	BIE	RW	0	<p>BIE: Enable brake interrupt</p> <p>0: Disable brake interrupt</p> <p>1: Enable brake interrupt</p>
6	Reserved	-	-	Reserved
5	COMIE	RW	0	<p>COMIE: COM interrupt enable</p> <p>0: COM interrupt disabled</p>

				1: COM interrupt enabled
4: 2	Reserved	-	-	Reserved
1	CC1IE	RW	0	CC1IE: Capture/Compare 1 interrupt enable 0: Capture/Compare 1 interrupt disabled 1: Capture/Compare 1 interrupt enabled
0	UIE	RW	0	UIE: Update interrupt enable 0: Update interrupt disabled 1: Update interrupt enabled

25.8.4. TIM16_17 Status Register (TIMx_SR)

Address offset: 0x010

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res						CC1OF	Res	BIF	Res	COMIF	Res			CC1F	UIF
-						RC_W0	-	RC_W0	-	RC_W0	-			RC_W0	RC_W0

Bit	Name	R/W	Reset Value	Function
15: 10	Reserved	-	-	Reserved
9	CC1OF	RC_W0	0	Capture/Compare 1 over-capture flag This flag can only be set to 1 by hardware when the corresponding channel is configured as input capture. Writing 0 can clear this bit. 0: No over-capture generated; 1: When CC1OF is set to 1, the counter value has been captured into the TIMx_CCR1 register.
8	Reserved	-	-	Reserved
7	BIF	RC_W0	0	Brake interrupt flag Once the brake input is valid, this bit is set to 1 by hardware. If the brake input is invalid, this bit can be cleared to 0 by software. 0: No brake event generated; 1: A valid level is detected on the brake input.
6	Reserved	-	-	Reserved
5	COMIF	RC_W0	0	COM interrupt flag This bit is set by hardware once a COM event occurs (when CCxE, CCxNE, OCxM have been updated). It is cleared by software. 0: No COM event generated; 1: COM interrupt pending
4: 2	Reserved	-	-	Reserved

Bit	Name	R/W	Reset Value	Function
1	CC1IF	RC_W0	0	<p>Capture/Compare 1 interrupt flag</p> <p>If channel CC1 is configured in output mode:</p> <p>This bit is set by hardware when the counter value matches the compare value, except in center-aligned mode (refer to the TIMx_CR1 register the CMS bit of the register). It is cleared by software.</p> <p>0: No match occurs;</p> <p>1: The value of TIMx_CNT matches the value of TIMx_CCR1.</p> <p>If channel CC1 is configured in input mode:</p> <p>This bit is set by hardware when a capture event occurs. It is cleared by software or by reading TIMx_CCR1.</p> <p>0: No input capture occurred;</p> <p>1: An input capture occurred and the counter value has been loaded into TIMx_CCR1 (a edge with the selected polarity was detected on IC1).</p>
0	UIF	RC_W0	0	<p>Update interrupt flag</p> <p>This bit is set by hardware when an update event occurs. It is cleared by software.</p> <p>0: No update event generated;</p> <p>1: An update event is pending. This bit is set by hardware when the register is updated:</p> <ul style="list-style-type: none"> – If UDIS=0 in TIMx_CR1 register, an update event is generated when REP_CNT=0 (when the counter overflows); – If UDIS=0 and URS=0 in TIMx_CR1 register, an update event is generated when UG=1 in TIMx_EGR register (software reinitializes CNT); – If UDIS=0 and URS=0 in TIMx_CR1 register, an update event is generated when CNT is reinitialized by a trigger event <p>event. (Refer to Slave Mode Control Register (TIMx_SMCR))</p>

25.8.5. TIM16_17 Event Generation Register (TIMx_EGR)

Address offset: 0x14

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	BG	Res	COMG	Res	Res	Res	CC1G	UG
-	-	-	--	-	-	-	-	W	-	W	-	-	-	W	W

Bit	Name	R/W	Reset Value	Function
15: 8	Res	-	-	Reserved
7	BG	W	0	<p>Generate brake event</p> <p>This bit is set to 1 by software to generate a brake event and is automatically cleared to 0 by hardware.</p> <p>0: No action;</p> <p>1: Generates a brake event. When MOE=0 and BIF=1, if the corresponding interrupt and DMA are enabled, the corresponding interrupt and DMA will be generated.</p>
6	Res	-	-	Reserved
5	COMG	W	0	<p>Capture/compare event generates a control update.</p> <p>This bit is set to 1 by software and automatically cleared to 0 by hardware.</p> <p>0: No action;</p> <p>1: When CCPC=1, allows updating the CCxE, CCxNE, and OCxM bits.</p> <p>Note: This bit is only valid for channels with complementary outputs.</p>
4: 2	Res	-	-	Reserved
1	CC1G	W	0	<p>Generate capture/compare 1 event</p> <p>This bit is set to 1 by software to generate a capture/compare event and is automatically cleared by hardware.</p> <p>0: No action;</p> <p>1: Generate a capture/compare event on channel CC1:</p> <p>If channel CC1 is configured as output:</p> <p>Set CC1IF=1, and if the corresponding interrupt and DMA are enabled, the corresponding interrupt and DMA are generated.</p> <p>If channel CC1 is configured as input:</p> <p>The current counter value is captured into the TIMx_CCR1 register, CC1IF is set to 1, and if the corresponding interrupt and DMA are enabled, the corresponding interrupt</p>

Bit	Name	R/W	Reset Value	Function
				and DMA are generated. If CC1IF is already set to 1, then set CC1OF=1.
0	UG	W	0	<p>Generate an update event. This bit is set by software and cleared automatically by hardware.</p> <p>0: No action; 1: Reinitialize the counter and generate an update event.</p> <p>Note: The prescaler counter is also cleared (but the prescaler</p> <p>The coefficient remains unchanged). In center-aligned mode or when DIR=0 (upcounting), the counter is cleared to 0; when DIR=1 (downcounting), the counter is loaded with the TIMx_ARR value.</p>

25.8.6. TIM16_17 Capture/Compare Mode Register 1 (TIMx_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res						Res		OC1M[2: 0]				OC1PE	OC1FE	CC1S[1: 0]	
Res							IC1F[3: 0]				IC1PSC[1: 0]				
-						-	RW	RW	RW	RW	RW	RW	RW	RW	RW

Output Compare Mode:

Bit	Name	R/W	Reset Value	Function
15: 7	Reserved	-	-	Reserved
6: 4	OC1M[2: 0]	RW	00	<p>Output compare 1 mode</p> <p>These 3 bits define the action of output reference signal OC1REF, which determines the values of OC1 and OC1N. OC1REF is active high, while the active levels of OC1 and OC1N depend on CC1P and CC1NP bits.</p> <p>000: Freeze. The comparison between output compare register TIMx_CCR1 and counter TIMx_CNT has no effect on OC1REF.</p> <p>;</p> <p>001: Set channel 1 to active level on match. Force OC1REF to high when TIMx_CNT matches TIMx_CCR1.</p> <p>010: Set channel 1 to inactive level on match. When the value of counter TIMx_CNT matches the capture/compare register</p> <p>Force OC1REF to low when TIMx_CNT matches TIMx_CCR1.</p>

Bit	Name	R/W	Reset Value	Function
				<p>011: Toggle. Toggle OC1REF level when TIMx_CCR1 = TIMx_CNT.</p> <p>100: Force to inactive level. Force OC1REF to low.</p> <p>101: Force to active level. Force OC1REF to high.</p> <p>110: PWM mode 1 - During up-counting, channel 1 is at active level when TIMx_CNT<TIMx_CCR1; otherwise, it is at inactive level.</p> <p>111: PWM mode 2 - During up-counting, channel 1 is at inactive level when TIMx_CNT<TIMx_CCR1; otherwise, it is at active level.</p> <p>Note 1: Once the LOCK level is set to 3 (LOCK bit in the TIMx_BDTR register) and CC1S=00 (the channel is configured as an output), this bit cannot be modified.</p> <p>Note 2: In PWM mode 1 or PWM mode 2, the OC1REF level only changes when the compare result changes or when switching from freeze mode to PWM mode in output compare mode.</p>
3	OC1PE	RW	0	<p>Output Compare 1 Preload Enable</p> <p>0: Disable the preload function of the TIMx_CCR1 register. The TIMx_CCR1 register can be written at any time, and the new value takes effect immediately.</p> <p>1: Enable the preload function of the TIMx_CCR1 register. Read and write operations only affect the preload register. The preload value of TIMx_CCR1 is loaded into the current register when an update event occurs.</p> <p>Note 1: Once the LOCK level is set to 3 (LOCK bit in the TIMx_BDTR register) and CC1S=00 (the channel is configured as an output), this bit cannot be modified.</p> <p>Note 2: In single-pulse mode only, PWM mode can be used without acknowledging the preload register; otherwise, its behavior is undefined.</p>
2	OC1FE	RW	0	<p>Output Compare 1 Fast Enable</p> <p>This bit is used to speed up the CC output's response to trigger input events.</p> <p>0: CC1 operates normally based on the counter and CCR1 value, even if the trigger is enabled. When there is an active edge at the trigger input, the minimum delay to activate the CC1 output is 5 clock cycles.</p>

Bit	Name	R/W	Reset Value	Function
				<p>1: The active edge input to the trigger acts as if a compare match has occurred. Therefore, OC is set as the comparison level and</p> <p>Independent of the comparison result. The delay between the active edge of the sampling trigger and the CC1 output is reduced to 3 clock cycles.</p> <p>OCFE only takes effect when the channel is configured in PWM1 or PWM2 mode.</p>
1: 0	CC1S[1: 0]	RW	00	<p>Capture/Compare 1 selection.</p> <p>These 2 bits define the channel direction (input/output) and the selection of the input pin:</p> <p>00: CC1 channel is configured as output;</p> <p>01: CC1 channel is configured as input, IC1 is mapped to TI1;</p> <p>10: CC1 channel is configured as input, IC1 is mapped to TI2;</p> <p>11: CC1 channel is configured as input, IC1 is mapped to TRC. This mode only works when the internal trigger input is selected</p> <p>(Selected by the TS bit in the TIMx_SMCR register).</p> <p>Note: CC1S is writable only when the channel is turned off (CC1E=0 in the TIMx_CCER register).</p>

Input capture mode:

Bit	Name	R/W	Reset Value	Function
15: 8	Reserved	-	-	Reserved
7: 4	IC1F[3: 0]	RW	0000	<p>Input capture 1 filter</p> <p>These bits define the sampling frequency and digital filter length for TI1 input. The digital filter consists of an event counter,</p> <p>It generates an output transition after recording N events:</p> <p>0000: No filter, sampling at fDTS</p> <p>0001: fSAMPLING=fCK_INT, N=2</p> <p>0010: fSAMPLING=fCK_INT, N=4</p> <p>0011: fSAMPLING=fCK_INT, N=8</p> <p>0100: fSAMPLING=fDTS/2, N=6</p> <p>0101: fSAMPLING=fDTS/2, N=8</p> <p>0110: fSAMPLING=fDTS/4, N=6</p> <p>0111: fSAMPLING=fDTS/4, N=8</p>

Bit	Name	R/W	Reset Value	Function
				1000: fSAMPLING=fDTS/8, N=6 1001: fSAMPLING=fDTS/8, N=8 1010: fSAMPLING=fDTS/16, N=5 1011: fSAMPLING=fDTS/16, N=6 1100: fSAMPLING=fDTS/16, N=8 1101: fSAMPLING=fDTS/32, N=5 1110: fSAMPLING=fDTS/32, N=6 1111: fSAMPLING=fDTS/32, N=8
3: 2	IC1PSC[1: 0]	RW	00	Capture/Compare 1 Prescaler 00: No prescaler, every edge detected on the capture input triggers a capture; 01: Capture triggered every 2 events; 10: Capture triggered every 4 events; 11: Capture triggered every 8 events.
1: 0	CC1S[1: 0]	RW	00	CC1S[1:0]: Capture/Compare 1 Selection. These 2 bits define the channel direction (input/output) and the selection of the input pin: 00: CC1 channel is configured as output; 01: CC1 channel is configured as input, IC1 is mapped to TI1; 10: CC1 channel is configured as input, IC1 is mapped to TI2; 11: CC1 channel is configured as input, IC1 is mapped to TRC. This mode only works when the internal trigger input is selected (Selected by the TS bit in the TIMx_SMCR register).

25.8.7. TIM16_17 Capture/Compare Enable Register (TIMx_CCER)

Address offset: 0x20

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	CC1NP	CC1NE	CC1P	CC1E
-	-	-	-	-	-	-	-	-	-	-	-	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15: 4	Reserved	-	-	Reserved
3	CC1NP	RW	0	Capture/Compare 1 complementary output polarity 0: OC1N active high 1: OC1N active low

Bit	Name	R/W	Reset Value	Function
				Note: Once the LOCK level (LCK bit in the TIMx_BDTR register) is set to 3 or 2 and CC1S=00 (channel configured as output), this bit cannot be modified.
2	CC1NE	RW	0	<p>Capture/Compare 1 complementary output enable</p> <p>0: OC1N output is disabled</p> <p>1: OC1N signal is output to the corresponding output pin</p> <p>When the CC1 channel is configured as an output, the OC1N output level is determined by the combined settings of MOE, OSSI, OSSR, OIS1, OIS1N, CC1E, and CC1NE bits, as shown in the table below.</p>
1	CC1P	RW	0	<p>Capture/Compare 1 output polarity</p> <p>CC1 channel configured as output:</p> <p>0: OC1 active high</p> <p>1: OC1 active low</p> <p>CC1 channel configured as input:</p> <p>CC1NP/CC1P bits select the polarity of TI1FP1 and TI2FP1 as trigger or capture signals.</p> <p>00: Not inverted / rising edge:</p> <p>TIxFP1 is active on rising edge (capture, trigger in reset mode, external clock or trigger mode);</p> <p>TIxFP1 is not inverted (gated mode, encoder mode).</p> <p>01: Inverted / falling edge:</p> <p>TIxFP1 is active on falling edge (capture, trigger in reset mode, external clock or trigger mode);</p> <p>TIxFP1 is inverted (gated mode, encoder mode).</p> <p>10: Reserved, do not use this configuration.</p> <p>11: Not inverted / both edges</p> <p>TIxFP1 is active on both rising and falling edges (capture, trigger in reset mode, external clock or trigger mode);</p> <p>TIxFP1 is not inverted (gated mode). This configuration cannot be applied in encoder mode.</p> <p>Note:</p> <p>Once the LOCK level (LOCK bit in TIMx_BDTR register) is set to 3 or 2, this bit cannot be modified.</p>
0	CC1E	RW	0	<p>Capture/Compare 1 output enable</p> <p>0: Capture mode disabled/OC1 output disabled</p> <p>1: Capture mode enabled/OC1 signal output to the corresponding output pin</p>

Bit	Name	R/W	Reset Value	Function
				<p>When the CC1 channel is configured as output, the OC1 output level is determined by the combined effect of MOE, OSSI, OSSR, OIS1, OIS1N, CC1E, and CC1NE bits, as shown in the table below.</p> <p>Note:</p> <p>For complementary output channels, this bit is preloaded. If the CCPC bit in the TIMx_CR2 register is set, the actual effective bit of CC1E will only load the preload value when a COM event occurs.</p>

Table 25-2 Output control of complementary OCx and OCxN channels with interrupt capability

Control bits					Output state	
MOE	OSSI	OSSR	CCxE	CCxNE	OCx output state	OCxN output state
1	X	0	0	0	Output disabled (disconnected from timer), OCx=0, OCx_EN=0	Output disabled (disconnected from timer), OCxN=0, OCxN_EN=0
		0	0	1	Output disabled (disconnected from timer), OCx=0, OCx_EN=0	OCxREF + Polarity, OCxN=OCxREF XOR CCxNP, OCxN_EN=1
		0	1	0	OCxREF + Polarity OCx=OCREF XOR CCxP, OCx_EN=1	Output disabled (disconnected from timer), OCxN=0, OCxN_EN=0
		0	1	1	OCREF + Polarity + dead-time, OCx_EN=1	Complement of OCREF (not OCREF) + Polarity + dead-time, OCxN_EN=1
		1	0	0	Output disabled (disconnected from timer), OCx=CCxP, OCx_EN=0	Output disabled (disconnected from timer), OCxN=CCxNP, OCxN_EN=0
		1	0	1	Output disabled (disconnected from timer), OCx=CCxP, OCx_EN=1	OCxREF + Polarity OCxN=OCxREF xor CCxNP, OCxN_EN=1
		1	1	0	OCxREF + Polarity OCx=OCxREF xor CCxP, OCx_EN=1	Off state (output enabled and inactive level), OCxN=CCxNP, OCxN_EN=1

Control bits					Output state	
MOE	OSSI	OSSR	CCxE	CCxNE	OCx output state	OCxN output state
		1	1	1	OCREF + Polarity + dead-time OCx_EN=1	Complementary of OC-REF (not OCREF) + polarity + dead-time OCN_EN=1
0	0	X	0	0	Output disabled (disconnected from timer)	
	0		0	1		
	0		1	0		
	0		1	1		
	1		0	0		
	1		0	1	Off state (output enabled and at inactive level) Asynchronous: OCx=CCxP, OCx_EN=1, OCxN=CCxNP, OCxN_EN=1 If the clock exists: After a dead time, assuming OISx and OISxN do not both correspond to the active level of OCx and OCxN, OCx=OISx and OCxN=OISxN.	
	1		1	0		
	1		1	1		

If neither output of a channel is used (CCxE = CCxNE = 0), then OISx, OISxN, CCxP, and CCxNP must all be cleared.

Note: The status of the external I/O pins connected to complementary OCx and OCxN channels depends on the state of OCx and OCxN channels, as well as GPIO and AFIO registers.

25.8.8. TIM16_17 counter (TIMx_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
15: 0	CNT[15:0]	RW	0	Counter value

25.8.9. TIM16_17 prescaler (TIMx_PSC)

Address offset: 0x28

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
-----	------	-----	-------------	----------

15: 0	PSC[15:0]	RW	0	<p>Prescaler value</p> <p>The clock frequency of the counter (CK_CNT) equals $f_{CK_PSC} / (PSC[15:0] + 1)$.</p> <p>PSC contains the value to be loaded into the current pre-scaler register when an update event occurs; the update event includes the counter</p> <p>Cleared by the UG bit of TIM_EGR or cleared by the slave controller operating in reset mode.</p>
-------	-----------	----	---	---

25.8.10. TIM16_17 auto-reload register (TIMx_ARR)

Address offset: 0x2c

Reset value: 0x0000 FFFF

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

Bit	Name	R/W	Reset Value	Function
15: 0	ARR[15:0]	RW	FFFF	<p>Auto-reload value</p> <p>ARR contains the value to be loaded into the actual auto-reload register.</p> <p>When the auto-reload value is empty, the counter does not work.</p>

25.8.11. TIM16_17 repetition counter register (TIMx_RCR)

Address offset: 0x30

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
15: 8	Reserved	-	-	Reserved
7: 0	REP[7:0]	RW	0	<p>Value of the period counter</p> <p>With the preload function enabled, these bits allow users to set the update rate of the compare registers.</p> <p>If the preload function is enabled, these bits allow users to set the update rate of the compare registers (i.e., periodically transferring from the preload register to the current</p>

				<p>register); if update interrupts are enabled, this will also affect the rate at which update interrupts are generated.</p> <p>Each time the down-counter REP_CNT reaches 0, an update event is generated, and the counter REP_CNT restarts counting from the REP value. Since REP_CNT only reloads the REP value when a period update event U_RC occurs, the new value written to the TIMx_RCR register will only take effect during the next period update event.</p> <p>This means that in PWM mode, (REP+1) corresponds to:</p> <ul style="list-style-type: none"> - In edge-aligned mode, the number of PWM cycles;
--	--	--	--	--

25.8.12. TIM16_17 Capture/Compare Register 1 (TIMx_CCR1)

Address offset: 0x34

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
15: 0	CCR1[15: 0]	RW/RO	0	<p>Capture/Compare 1 value</p> <p>If the CC1 channel is configured as output:</p> <p>CCR1 contains the value loaded into the current capture/compare 1 register (preload value).</p> <p>If the preload feature is not selected in the TIMx_CCMR1 register (OC1PE bit), it is always loaded into the current register.</p> <p>Otherwise, this preload value is loaded into the current capture/compare 1 register only when an update event occurs.</p> <p>The current capture/compare register contains the value to be compared with the counter TIMx_CNT and outputs a signal on the OC1 port.</p> <p>If the CC1 channel is configured as input:</p> <p>CCR1 contains the counter value transferred by the last input capture 1 event (IC1).</p>

25.8.13. TIM16_17 Brake and Dead-Time Register (TIMx_BDTR)

Address offset: 0x44

Reset value: 0x0000 0000

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

MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Note: According to the lock settings, AOE, BKP, BKE, OSSI, OSSR, and DTG[7:0] bits can be write-protected. It is necessary to configure them during the first write operation to the TIMx_BDTR register.

Bit	Name	R/W	Reset Value	Function
15	MOE	RW	0	<p>Main output enable</p> <p>Once the brake input is active, this bit is asynchronously cleared by hardware. Can be cleared by software or automatically set according to the AOE bit value. It is only valid for output-configured channels.</p> <p>0: Disable OC and OCN output or force to idle state; 1: If the corresponding enable bit is set (CCxE, CCxNE bits in TIMx_CCER register), OC and OCN output.</p>
14	AOE	RW	0	<p>Automatic output enable</p> <p>0: MOE can only be set by software; 1: MOE can be set by software or automatically set at the next update event (if brake input is inactive).</p> <p>Note: Once the LOCK level (LOCK bit in TIMx_BDTR register) is set to 1, this bit cannot be modified.</p>
13	BKP	RW	0	<p>Brake input polarity</p> <p>0: Brake input active low; 1: Brake input active high.</p> <p>Note: Once the LOCK level (LOCK bit in TIMx_BDTR register) is set to 1, this bit cannot be modified.</p> <p>Any write operation to this bit takes effect after a delay of one APB clock cycle.</p>
12	BKE	RW	0	<p>Brake function enable</p> <p>0: Disable brake input; 1: Enable brake input.</p> <p>Note: Once the LOCK level (LOCK bit in TIMx_BDTR register) is set to 1, this bit cannot be modified.</p> <p>Any write operation to this bit takes effect after a delay of one APB clock cycle.</p>
11	OSSR	RW	0	<p>Off-state selection in run mode</p> <p>This bit is used when MOE=1 and the channel is in complementary output mode. The OSSR bit does not exist in timers without complementary outputs.</p> <p>Refer to the detailed description of OC/OCN enable (Capture/Compare Enable Register (TIMx_CCER)).</p>

Bit	Name	R/W	Reset Value	Function
				<p>0: When the timer is not active, OC/OCN output is disabled (OC/OCN enabled output signal = 0);</p> <p>1: When the timer is not working, once CCxE=1 or CCxNE=1, enable OC/OCN output and output invalid level.</p> <p>OC/OCN enabled output signal = 1.</p> <p>Note: Once the LOCK level (LOCK bit in the TIMx_BDTR register) is set to 2, this bit cannot be modified.</p>
10	OSSI	RW	0	<p>Off-state selection in idle mode</p> <p>This bit is used when MOE=0 and the channel is configured as output.</p> <p>Refer to the detailed description of OC/OCN enable (Capture/Compare Enable Register (TIMx_CCER)).</p> <p>0: When the timer is not active, OC/OCN output is disabled (OC/OCN enabled output signal = 0);</p> <p>1: When the timer is not active, once CCxE=1 or CCxNE=1, OC/OCN first outputs its idle level.</p> <p>OC/OCN enabled output signal = 1.</p> <p>Note: Once the LOCK level (LOCK bit in the TIMx_BDTR register) is set to 2, this bit cannot be modified.</p>
9: 8	LOCK[1:0]	RW	00	<p>Lock setup</p> <p>This bit provides write protection to prevent software errors.</p> <p>00: Lock off, no write protection for the register;</p> <p>01: Lock level 1, the DTG/BKE/BKP/AOE bits in the TIMx_BDTR register and the OISx/OISxN bits in the TIMx_CR2 register cannot be written;</p> <p>10: Lock level 2, bits in lock level 1 cannot be written, nor can the CC polarity bits (once the corresponding channel is set as output via the CCxS bit, the CCxP/CCNxP bits in the TIMx_CCER register) or the OSSR/OSSI bits;</p> <p>11: Lock level 3, bits in lock level 2 cannot be written, nor can the CC control bits (once the corresponding channel is set as output via the CCxS bit, the OCxM/OCxPE bits in the TIMx_CCMRx register);</p> <p>Note: After a system reset, the LOCK bit can only be written once. Once written to the TIMx_BDTR register, its contents are frozen until</p> <p>To reset.</p>

Bit	Name	R/W	Reset Value	Function
7: 0	DTG[7:0]	RW	0000 0000	<p>Dead-time generator setup</p> <p>These bits define the dead-time duration inserted between complementary outputs. Assuming DT represents its duration:</p> <p>DTG[7:5]=0xx => DT=DTG[7:0] × Tdtg, Tdtg = TDTs;</p> <p>DTG[7:5]=10x => DT=(64+DTG[5:0]) × Tdtg, Tdtg = 2 × TDTs;</p> <p>DTG[7:5]=110 => DT=(32+DTG[4:0]) × Tdtg, Tdtg = 8 × TDTs;</p> <p>DTG[7:5]=111 => DT=(32+DTG[4:0]) × Tdtg, Tdtg = 16 × TDTs;</p> <p>Example: If TDTs = 125ns (8MHz), the possible dead-time is:</p> <p>0 to 15875ns, if the step time is 125ns;</p> <p>16us to 31750ns, if the step time is 250ns;</p> <p>32us to 63us, if the step time is 1us;</p> <p>64us to 126us, if the step time is 2us;</p> <p>Note: Once the LOCK level (LOCK bits in TIMx_BDTR register) is set to 1, 2, or 3, these bits cannot be modified.</p> <p>Modified.</p>

25.8.14. TIM16_17 DMA control register (TIMx_DCR)

Address offset: 0x48

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	DBL[4:0]					Res			DBA[4:0]				
-	-	-	RW	RW	RW	RW	RW	-	-	-	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15: 13	Reserved	-	-	Reserved
12: 8	DBL[4:0]	RW	0 0000	<p>DMA continuous transfer length</p> <p>These bits define the transfer length of DMA in continuous mode (when reading from or writing to the address of the TIMx_DMAR register , the timer performs one continuous transfer), i.e.: defines the number of bytes to be transferred:</p> <p>00000: 1 transfer</p> <p>00001: 2 transfers</p> <p>00010: 3 transfers</p>

Bit	Name	R/W	Reset Value	Function
				<p>.....</p> <p>.....</p> <p>10001: 18 transfers</p> <p>Example: We consider such a transfer: DBL=7, DBA=TIM2_CR1</p> <p>- If DBL=7 and DBA=TIM2_CR1 represents the address of the data to be transferred, the transfer address is given by:</p> <p>(TIMx_CR1 address) + DBA + (DMA index), where DMA index = DBL</p> <p>Where (TIMx_CR1 address) + DBA + 7 gives the address from which data will be written or read, so the data transfer will occur across 7 registers starting from the address (TIMx_CR1 address) + DBA.</p> <p>Depending on the DMA data length setting, the following scenarios may occur:</p> <p>- If data is set to half-word (16-bit), the data will be transferred to all 7 registers.</p> <p>- If data is set to byte, the data 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.</p> <p>Therefore, for timers, the user must specify the data width for DMA transfers.</p>
7: 5	Reserved	-	-	Reserved
4: 0	DBA[4:0]	RW	0 0000	<p>DBA[4:0]: DMA base address</p> <p>These bits define the base address for DMA in continuous mode (when reading or writing to the TIMx_DMAR register address</p> <p>), DBA is defined as the offset from the base address of the TIMx_CR1 register:</p> <p>00000: TIMx_CR1, 00001: TIMx_CR2, 00010: TIMx_SMCR,</p>

25.8.15. DMA address for TIM16_17 continuous mode (TIMx_DMAR)

Address offset: 0x4C

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
15: 0	DMAB[31:0]	RW	0	<p>DMA Continuous Transfer Register</p> <p>Reading or writing to the TIMx_DMAR register results in access to the register at the following address:</p> <p>TIMx_CR1 address + (DBA + DMA pointer) x4, where:</p> <p>"TIMx_CR1 address" is the address of Control Register 1;</p> <p>"DBA" is the base address defined in the TIMx_DCR register;</p> <p>The "DMA pointer" is an offset automatically controlled by the DMA, which depends on the DBL defined in the TIMx_DCR register.</p>

Note: When using the DMA continuous transfer function, the value of the CNDTR register in the corresponding DMA channel must correspond to the DBL value in the TIMx_DCR register; otherwise, it will not function properly.

26. Low-power Timer (LPTIM)

26.1. Introduction

The LPTIM is a 16-bit timer. The LPTIM's ability to wake the system from low-power modes makes it suitable for implementing low-power applications.

The LPTIM can count using high-frequency clocks (PCLK) and low-frequency clocks (LSI/LSE), providing the required functionality and performance while minimizing power consumption.

26.2. LPTIM Key Features

- 16-bit Up Counter
- 3-bit Prescaler with 8 Possible Division Factors (1, 2, 4, 8, 16, 32, 64, 128)
- Optional Clock
 - Internal Clock Sources: LSE, LSI, or APB Clock (PCLK)
- 16-bit ARR Reload Register
- Continuous/Single-shot Mode

26.3. Low-power Timer (LPTIM) Functional Description

26.3.1. LPTIM Block Diagram

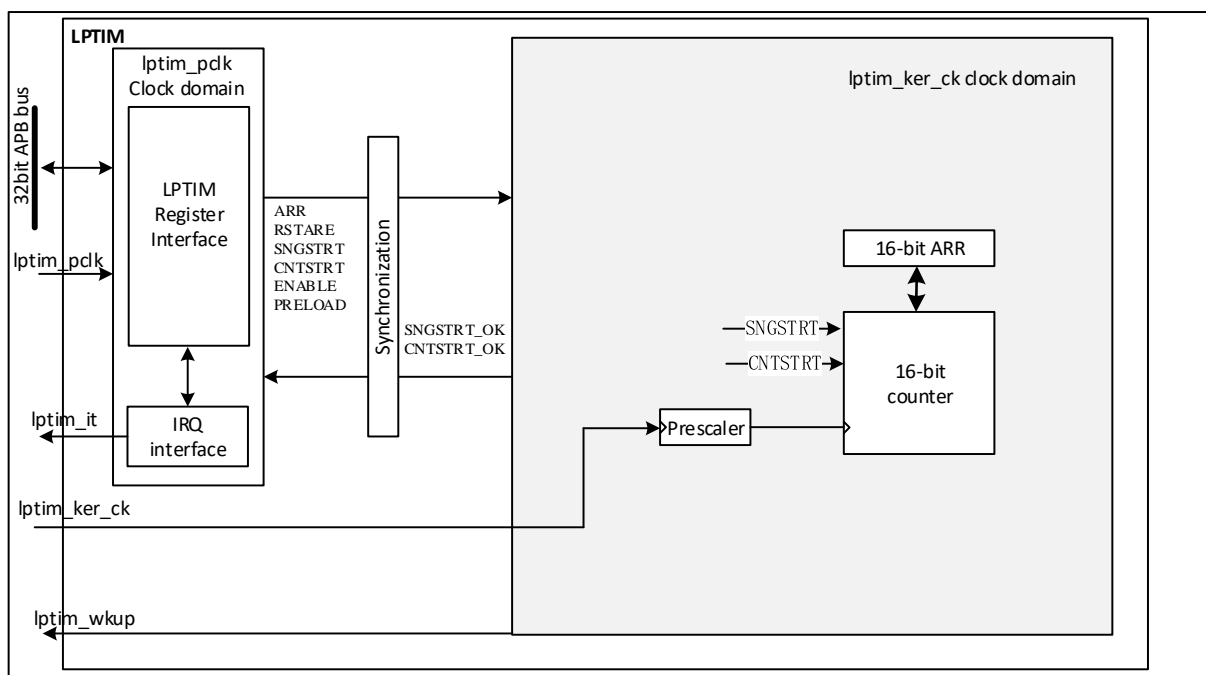


Figure 26-1 Low-power Timer Block Diagram

26.3.2. LPTIM Pins and Internal Signals

Name	Signal Type	Description
lptim_pclk	Input	LPTIM APB Clock

lptim_ker_ck	Input	LPTIM Internal Clock Source
lptim_it	Output	LPTIM Global Interrupt
lptim_wkup	Output	LPTIM Wake-up Event

26.3.3. LPTIM Reset and Clock

The LPTIM can be clocked by multiple clock sources.

It can be clocked using an internal clock signal via the RCC module (this clock signal can be selected from PCLK, LSI, or LSE sources).

26.3.4. Prescaler

The LPTIM 16-bit counter is driven by a configurable power-of-two prescaler. The prescaler division ratio is controlled by PRESC[2:0].

The following table lists all scenarios:

Table 26-1 Prescaler Factor

Programming	Dividing factor
000	/1
001	/2
010	/4
011	/8
100	/16
101	/32
110	/64
111	/128

26.3.5. Operating mode

LPTIM has the following two operating modes.

- **Single mode: The timer starts from a trigger event (writing to the SNGSTRT register) and stops when the ARR value is reached.**

To enable single counting, the SNGSTRT bit must be set to 1.

A new trigger event will restart the timer. After the counter starts, any trigger event before reaching ARR will be ignored.

- **Continuous mode: The timer runs freely, starting from a trigger event (writing to the CNTSTRT register) and continues until the timer is disabled.**

To enable continuous counting, the LPTIM_CR.CNTSTRT bit must be set to 1.

Setting LPTIM_CR.CNTSTRT will start the counter for continuous counting.

It is possible to switch from single-shot mode to continuous mode in real-time:

- If the continuous mode was previously selected, setting LPTIM_CR.SNGSTRT will switch the LPTIM to single-shot mode. The counter (if active) will stop immediately after reaching ARR.

- If the single-shot mode was previously selected, setting LPTIM_CR.CNTSTRT will switch the LPTIM to continuous mode. The counter (if active) will restart as soon as it reaches ARR.

26.3.6. Register update

The PRELOAD bit controls how the LPTIM_ARR register is updated:

- When the PRELOAD bit is reset to "0": The LPTIM_ARR register is updated immediately after any write access.
- When the PRELOAD bit is set to "1": If the timer is already started, LPTIM_ARR will be updated synchronously at the next LPTIM update event.

The LPTIM APB interface and LPTIM Kernel logic use different clocks, so there is a certain delay between APB writes and when the written values are applied to the counter comparator. During this delay period, any additional write operations to these registers must be avoided.

The ARROK flag in the LPTIM_ISR register indicates whether the write operation to the LPTIM_ARR register is completed.

After a write operation to the LPTIM_ARR register, a new write operation to the same register can only be performed once the previous write operation is completed. Any consecutive writes before the ARROK flag is set will result in unpredictable outcomes.

26.3.7. Counter mode

LPTIM only supports internal clock counting mode.

The ENABLE bit in the LPTIM_CR register is used to enable/disable the LPTIM core logic. After setting the ENABLE bit, the hardware delays 3 counter clocks before enabling the LPTIM.

The LPTIM_CFGR and LPTIM_IER registers can only be modified when the LPTIM is disabled.

26.3.8. Counter reset

To reset the content of the LPTIM_CNT register, a reset mechanism is provided:

Synchronous reset mechanism:

The synchronous reset is controlled by the LPTIM_CR.CONURST bit. After setting the COUNTRST bit, the reset signal is sent to the LPTIM Kernel clock domain. Therefore, note that several clock cycles have passed in the LPTIM Kernel clock domain logic before the reset takes effect. This causes the LPTIM counter to count several extra numbers from reset trigger to reset taking effect.

Since COUNTRST is in the APB clock domain while the LPTIM counter is in the LPTIM Kernel clock domain, when writing 1 to the COUNTRST bit, a 3-clock-cycle delay of the Kernel clock is required to synchronize the reset signal from the APB clock domain.

Asynchronous reset mechanism:

Asynchronous reset is controlled by the RSTARE bit in the LPTIM_CR register. When this bit is set to 1, any read access to the LPTIM_CNT register will reset its content to zero. The asynchronous reset should be triggered during a time period when the LPTIM kernel clock is not provided.

Note that to achieve reliable reading of the LPTIM_CNT register content, two consecutive read accesses must be performed and compared. When the values from two read accesses are equal, the read access can be considered reliable. However, when asynchronous reset is enabled, it is not possible to read the LPTIM_CNT register twice.

26.3.9. Debug mode

When the chip enters debug mode, the LPTIM counter will either continue normal operation or stop working based on the DBG_LPTIM_STOP configuration bit in the DBG module.

26.4. LPTIM low-power modes

Table 26-26 Differences in LPTIM low-power modes

Mode	Description
Sleep	Functionality remains unaffected. LPTIM interrupt (when enabled) exits sleep mode.
Stop	Functionality remains unaffected when LSE/LSI clock is present. LPTIM interrupt (when enabled) exits stop mode.

26.5. LPTIM interrupt

If the following events are enabled in the LPTIM_IER register, they will generate interrupt/wakeup events:

- Auto-reload matching

Note: If the corresponding bit in the LPTIM_IER register (interrupt enable register) is set to 1 after the corresponding flag in the LPTIM_ISR register (status register) is set to 1, no interrupt is generated.

Table 26-26 LPTIM interrupt events

Interrupt event	Description
Auto-reload match	The interrupt flag is set when the content of the counter register (LPTIM_CNT) matches the content of the auto-reload register (LPTIM_ARR)
Auto-reload register update OK	The interrupt flag is set when the write operation to the LPTIM_ARR register is completed

26.6. LPTIM registers

26.6.1. LPTIM interrupt and status register (LPTIM_ISR)

Address offset: 0x000

Reset value: 0x0000 0000

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

-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	ARROK	Res	Res	ARRM	Res
-	-	-	-	-	-	-	-	-	-	-	R	-	-	R	-

Bit	Name	R/W	Reset Value	Function
31: 5	Reserved	-	-	Reserved
4	ARROK	R	0	Auto-reload register update OK. ARROK is set by hardware to notify the application that the APB bus write operation to LPTIM_ARR has been successfully completed. Writing 1 to LPTIM_ICR.ARROKCF clears the ARROK flag.
3: 2	Reserved	-	-	Reserved
1	ARRM	R	0	Auto-reload match ARRM is set by hardware to notify the application that the LPTIM_CNT register value matches the value of the LPTIM_ARR register. Writing 1 to the ARRMCF bit of the LPTIM_ICR register clears the ARRM flag
0	Reserved	-	-	Reserved

26.6.2. LPTIM Interrupt Clear Register (LPTIM_ICR)

Address offset: 0x004

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 5	Reserved	-	-	Reserved
4	ARROKCF	W	0	Auto-reload register update OK clear flag Writing 1 to this bit clears the ARROK flag in the LPTIM_ISR register
3: 2	Reserved	-	-	Reserved
1	ARRMCF	W	0	Auto-reload match clear flag Writing 1 to this bit clears the ARRM flag in the LPTIM_ISR register

0	Reserved	-	-	Reserved
---	----------	---	---	----------

26.6.3. LPTIM Interrupt Enable Register (LPTIM_IER)

Address offset: 0x008

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 5	Reserved	-	-	Reserved
4	ARROKIE	RW	0	Auto-reload register update OK interrupt enabled. 0: ARROK interrupt disabled 1: ARROK interrupt enabled
3: 2	Reserved	-	-	Reserved
1	ARRMIE	RW	0	Auto-reload match interrupt enabled 0: ARRM interrupt disabled 1: ARRM interrupt enabled
0	Reserved	-	-	Reserved

26.6.4. LPTIM configuration register (LPTIM_CFGR)

Address offset: 0x00C

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 23	Reserved	-	-	Reserved
22	PRELOAD	RW	0	Register update mode The preload bit controls the update mode of the LPTIM_ARR register

				0: Registers are updated after each APB bus write access 1: Registers are updated at the end of the current LPTIM cycle
21: 12	Reserved	-	-	Reserved
11: 9	PRESC[2:0]	RW	0	Clock prescaler PRESC bits configure the prescaler division factor. It can be one of the following division factors: 000: /1 001: /2 010: /4 011: /8 100: /16 101: /32 110: /64 111: /128
8: 0	Reserved	-	-	Reserved

26.6.5. LPTIM control register (LPTIM_CR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Re s	Re s	Re s	Re s	Re s	Re s	Re s	Re s	Re s	Re s	Re s	R STAR E	COUN T RST	CNT STR T	SNGSTR T	EN ABL E
-	-	-	-	-	-	-	-	-	-	-	RW	RS	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 5	Reserved	-	-	Reserved
4	RSTARE	RW	0	Reset after read enable This bit is set to "1" and cleared to "0" by software. When RSTARE is set to "1", any read access to the LPTIM_CNT register will asynchronously reset the LPTIM_CNT register contents.
3	COUNTRST	RS	0	Counter reset. This bit is set to "1" by software and cleared to "0" by hardware. When set to "1", this bit triggers a synchronous

Bit	Name	R/W	Reset Value	Function
				reset of the LPTIM_CNT counter register. Due to the synchronous nature of this reset, it only needs to be released after a synchronization delay of 3 LPTIM core clock cycles (the LPTIM core clock may differ from the APB clock). Note: Software must never set COUNTRST to "1" before it has been cleared to "0" by hardware. Therefore, software should check whether the COUNTRST bit has been cleared to "0" before attempting to set it to "1".
2	CNTSTRT	RW	0	Timer starts in continuous mode. This bit is set by software, setting this bit will start LPTIM in continuous mode. If this bit is set to 1 during single-shot mode counting, the timer will not stop at the next pulse mode count when LPTIM_ARR and LPTIM_CNT registers match. LPTIM counter keeps counting in continuous mode. Note: This bit can only be set to 1 when LPTIM is enabled. It will be automatically reset by hardware.
1	SNGSTRT	RW	0	LPTIM starts in single-shot mode. This bit is set by software and cleared by hardware. Setting this bit will start LPTIM in single-pulse mode. Note: This bit can only be set to 1 when LPTIM is enabled. It will be automatically reset by hardware.
0	ENABLE	RW	0	LPTIM enable bit, set and cleared by software 0: LPTIM disabled 1: LPTIM enabled

26.6.6. LPTIM auto-reload register (LPTIM_ARR)

Address offset: 0x018

Reset value: 0x0000 0001

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

Bit	Name	R/W	Reset Value	Function
31: 16	Reserved	-	-	Reserved
15: 0	ARR	RW	0x0001	Auto-reload value

				ARR is the auto-reload value of LPTIM This register can only be updated when LPTIM is enabled
--	--	--	--	--

26.6.7. LPTIM counter register (LPTIM_CNT)

Address offset: 0x01C

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 16	Reserved	-	-	Reserved
15: 0	CNT	R	0	Counter value When LPTIM operates with an asynchronous clock, reading the LPTIM_CNT register may return unreliable values. Therefore, in this case, it is necessary to perform two consecutive read accesses and verify whether the two returned values are identical. When two consecutive read accesses return the same value, the read access can be considered reliable.

27. Independent watchdog (IWDG)

27.1. Introduction

The chip integrates an independent watchdog timer (IWDG), which features high security, precise timing, and flexible usage. IWDG detects and resolves malfunctions caused by software failures and triggers a system reset when the counter reaches the specified timeout value.

IWDG is clocked by LSI, allowing it to remain operational even when the system clock is turned off.

IWDG is best suited for applications where the watchdog needs to operate completely independently outside the main program and where timing precision requirements are low.

27.2. IWDG key features

- Free-running downcounter
- The clock is provided by an independent RC oscillator (can operate in Stop mode)
- After enabling IWDG in hardware or software mode, the RCC module automatically enables LSI as the IWDG clock.
- Once the watchdog is activated, a reset is generated when the counter counts down to 0x000.

27.3. IWDG functional description

27.3.1. IWDG block diagram

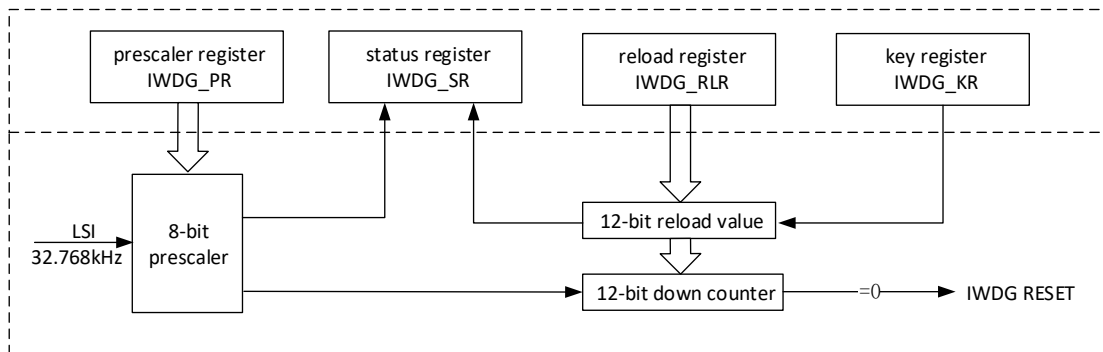


Figure 27-1 IWDG block diagram

Note: The watchdog function operates in the VDDD power domain, meaning it remains functional during Stop and Standby modes.

Writing 0xCCCC into the key register (IWDG_KR) enables the independent watchdog; at this point, the counter starts decrementing from its reset value 0xFFFF. When the counter reaches the end value 0x000, a reset signal (IWDG_RESET) is generated.

Whenever the value 0xAAAA is written into the key register IWDG_KR, the value in IWDG_RLR is reloaded into the counter to prevent an IWDG reset.

Table 27-1 Watchdog timeout period with 32.768 kHz input clock (LSI)

Prescaler factor	PR[2:0] bits	Minimum time (ms) RL[11:0]=0x000	Maximum time (ms) RL[11:0]=0xFFFF
/4	0	0.125	511.875
/8	1	0.25	1023.75
/16	2	0.5	2047.5
/32	3	1	4095
/64	4	2	8190
/128	5	4	16380
/256	(6 or 7)	8	32760

Note: These times are based on a 32.768 kHz clock. Additionally, even if the RC oscillator frequency is precise, the exact timing still depends on the phase difference between the APB interface clock and the RC oscillator clock, so there will always be one full RC cycle of uncertainty. Calibrating the LSI can achieve a relatively accurate watchdog timeout period.

27.3.2. Hardware watchdog

If the option bytes loaded at power-on are configured to enable the hardware watchdog, the IWDG is automatically enabled at power-on. If the KEY register of IWDG_KR is not rewritten by software before the counter reaches 0, a reset signal is generated.

27.3.3. Register protection

Write access to the IWDG prescaler and IWDG reload registers is protected. To modify them, the user must first write 0x00005555 to the KEY register of IWDG_KR. Writing other values to these registers will disrupt the timing sequence. For example, writing 0x0000AAAA to load will cause the registers to be protected again.

If the values of the prescaler register or reload register are being updated, the status register will reflect this.

27.3.4. Debug mode and STOP mode

This feature exists only when the system supports debug mode.

When the CPU enters debug mode, whether the IWDG continues counting or freezes the timer depends on the configuration of DBG_IWDG_STOP in the DBGMCU module.

When the CPU enters STOP low-power mode, the IWDG_STOP bit in the option bytes (Option byte) in Flash controls whether the IWDG continues normal counting or freezes the timer.

The configuration of IWDG_STOP in option bytes is as follows:

Configure IWDG timer operation status in STOP mode:

0: Timer frozen

1: Normal operation

27.4. IWDG registers

These peripheral registers can be operated in half-word (16-bit) or word (32-bit) mode.

27.4.1. Key Register (IWDG_KR)

Address offset: 0x00

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 16	Reserved	-	-	Reserved
15: 0	KEY[15:0]	W	0x00	<p>Key value.</p> <p>The software must write 0xAAAA to this register at regular intervals. Otherwise, when the counter counts down to 0, the watchdog will generate a reset.</p> <p>0x5555: Indicates permission to access the IWDG_PR and IWDG_RLR registers;</p> <p>0xCCCC: Indicates starting the IWDG (this command word is not restricted if the hardware watchdog is selected).</p>

27.4.2. 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
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	PR[2:0]		
-	-	-	-	-	-	-	-	-	-	-	-	-	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 3	Reserved	-	-	Reserved
2: 0	PR[2:0]	RW	0	Prescaler value.

				<p>Configure this register to select the prescaler value for the counter clock.</p> <p>To modify this register, the PVU bit in the IWDG_SR register must be 0.</p> <p>000: 4 division; 001: 8 division; 010: 16 division; 011: 32 division; 100: 64 division; 101: 128 division; 110: 256 division; 111: 256 division;</p>
--	--	--	--	---

27.4.3. Reload register (IWDG_RLR)

Address offset: 0x08

Reset value: 0x0000 0FFF

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

Bit	Name	R/W	Reset Value	Function
31: 12	Res	-	-	Reserved
11: 0	RL[11:0]	RW	0xFFFF	<p>IWDG counter reload value.</p> <p>When 0xAAAA is written to the IWDG_KR register, the RL value is transferred to the counter. The counter then begins decrementing from this value. The watchdog timeout period can be calculated using this RL value and the clock prescaler value.</p> <p>The register can only be modified when IWDG_SR.RVU=0.</p> <p>Additionally, please note that after writing the reload value to the RLR register, you must wait for three LSI clock cycles before reading it.</p>

27.4.4. Status register (IWDG_SR)

Address offset: 0x0C

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 2	Reserved	-	-	Reserved
1	RVU	R	0	<p>Watchdog counter reload value update.</p> <p>The RVU can only function properly after the IWDG is started by both software and hardware.</p> <p>This bit is set by hardware to indicate that the reload value is being updated. This bit is cleared by hardware when the reload value update is completed.</p>
0	PVU	R	0	<p>Watchdog prescaler value update.</p> <p>The IWDG must be started by both software and hardware for PVU to function properly.</p> <p>This bit is set to 1 by hardware, indicating that the prescaler value is being updated. This bit is cleared by hardware when the prescaler value update is complete.</p>

Note: Before updating IWDG_PR.PR and IWDG_RLR.RL, wait for IWDG_SR.PVU and IWDG_SR.RVU to become 0, respectively. However, after updating IWDG_PR.PR and IWDG_RLR.RL, it is not necessary to wait for IWDG_SR.PVU and IWDG_SR.RVU to become 0 before proceeding with the following code.

28. Window Watchdog (WWDG)

28.1. Introduction

The Window Watchdog (WWDG) is typically used to monitor software faults caused by external disturbances or unforeseen logic conditions that cause the application to deviate from its normal operation sequence. Unless the program refreshes the downcounter value before the T6 bit becomes 0, the watchdog circuit will generate an MCU reset when the preset time period is reached. If the 7-bit downcounter value in the control register is refreshed before the downcounter reaches the window register value, an MCU reset will also occur. This means the counter must be refreshed within the specified time window.

The WWDG clock is derived from the APB clock after prescaling, and it detects abnormal late or early operations of the application through a configurable time window.

The WWDG is most suitable for applications that require the watchdog to function within a precise timing window.

28.2. WWDG Key Features

- Programmable free-running downcounter.

Reset Conditions

- Reset when the downcounter value is less than 0x40 (if the watchdog is activated).
- Reset when reloading the downcounter outside the window (if the watchdog is activated).
- Early Wakeup Interrupt (EWI): Triggered when the downcounter equals 0x40 (if enabled and the watchdog is activated).

28.3. WWDG Functional Description

If the watchdog is activated (WDGA bit in the WWDG_CR register is set to 1), a reset will be triggered when the 7-bit downcounter (T[6:0] bits) decrements from 0x40 to 0x3F (T6 is cleared). If the software reloads the counter when its value is greater than the value stored in the window register, a reset will be generated.

The application must periodically write to the WWDG_CR register during normal operation to prevent an MCU reset. This operation can only be performed when the counter value is below the window register value and above 0x3F. The value to be stored in the WWDG_CR register must be between 0xFF and 0xC0.

28.3.1. WWDG block diagram

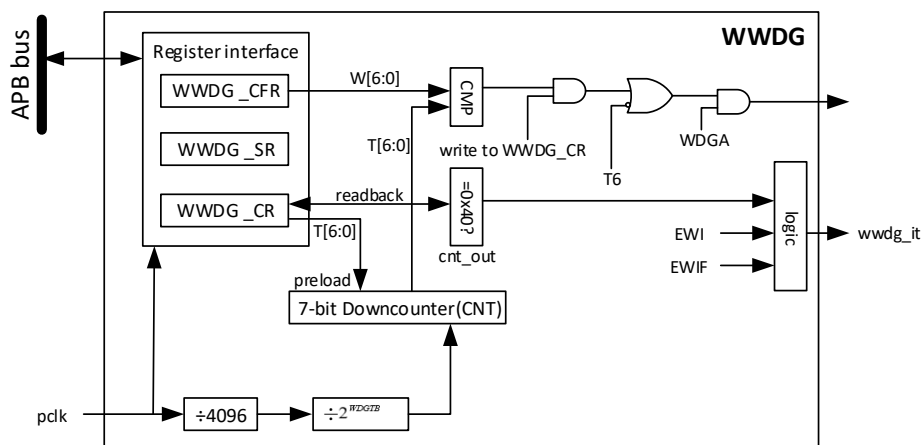


Figure 28-1 Window watchdog block diagram

28.3.2. Activate the watchdog

After a system reset, the watchdog is always disabled. It can be enabled by setting the WDGA bit in the WWDG_CR register or through the option byte. Once enabled, it cannot be disabled again unless a reset is performed.

The WWDG_SW bit in the option byte can also activate the watchdog, with the following values:

- 0: Hardware watchdog
- 1: Software watchdog

The watchdog is activated if either hardware or software activation is enabled.

28.3.3. Control the downcounter

The downcounter operates in free-running mode, meaning it continues to count down even if the watchdog is disabled. When the watchdog is enabled, the T6 bit must be set to prevent an immediate reset.

The T[5:0] bits contain the number of ticks before the watchdog generates a reset; the delay time before reset varies between a minimum and maximum value because the prescaler value is unknown when writing to the WWDG_CR register. The window register in the configuration register (WWDG_CFR) has an upper limit value. To avoid a reset, the downcounter must be reloaded when its value is less than the window register value and greater than 0x3F. Figure 28-2 illustrates the operation of the window register.

Another method to reload the counter is by using the Early Wakeup Interrupt (EWI). Enable this interrupt by setting the WEI bit in the WWDG_CFR register. This interrupt is generated when the downcounter reaches 0x40, and the corresponding interrupt service routine (ISR) can be used to reload the counter to prevent WWDG reset. Writing '0' to the WWDG_SR register clears this interrupt.

Note: A software reset can be generated using the T6 bit (set WDGA bit to '1' and T6 bit to '0').

28.3.4. Advanced watchdog interrupt function

If specific safety operations or data logging must be performed before the actual reset occurs, the Early Wakeup Interrupt (EWI) can be used. Enable the EWI interrupt by setting the EWI bit in the WWDG_CFR register. When the value of the down-counter is 0x40, an EWI interrupt is generated. Before resetting the device, the corresponding interrupt service routine (ISR) can be used to trigger specific operations (such as communication or data logging).

In some applications, the EWI interrupt can be used to manage software system checks and/or system recovery/functional degradation without generating a WWDG reset. In this case, the corresponding interrupt service routine (ISR) can be used to reload the WWDG counter to avoid a WWDG reset before triggering the required operation.

Clear the EWI interrupt by writing 0 to the EWIF bit in the WWDG_SR register.

Note: When the EWI interrupt cannot be used due to a system lock in a higher priority task, a WWDG reset will eventually occur

28.3.5. How to program the watchdog timeout

When writing to the WWDG_CR register, always write 1 to the T6 bit to avoid generating an immediate reset.

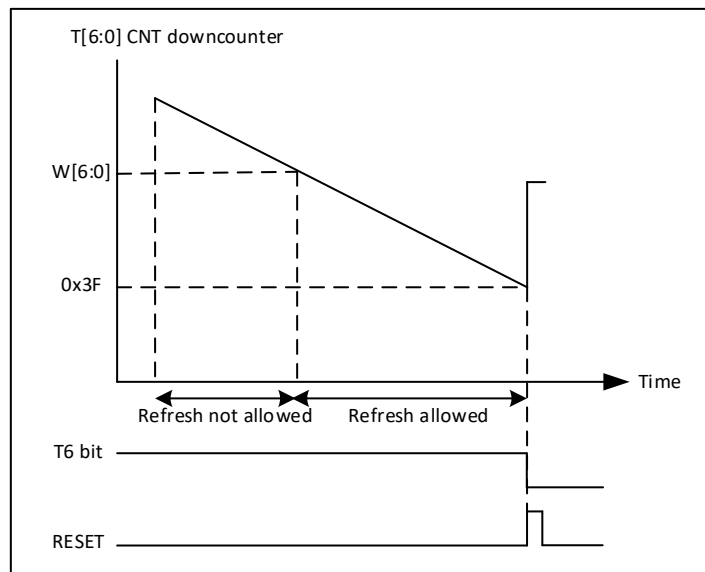


Figure 28-2 Window watchdog timing diagram

The formula to calculate the WWDG timeout value is as follows:

$$t_{\text{WWDG}} = t_{\text{PCLK}} \times 4096 \times 2^{\text{WWDGTB}[1:0]} \times (\text{T}[5:0] + 1) \quad (\text{ms})$$

Where:

1. t_{WWDG} : WWDG timeout
2. t_{PCLK} : APB clock cycle, measured in ms
3. 4096: Corresponds to the value of the internal divider

28.3.6. Debug mode

When the microcontroller enters debug mode (Cortex-M0+ core halted), the WWDG counter can continue to operate or stop depending on the status of the DBG_WWDG_STOP configuration bit in the DBGMCU module. Refer to the Debug mode chapter for details.

28.4. WWDG registers

28.4.1. Control register (WWDG_CR)

Address offset: 0x00

Reset value: 0x0000 007F

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

Bit	Name	R/W	Reset Value	Function
31: 8	Reserved	-	-	Reserved
7	WDGA	RS	0	WDGA: Activation bit. This bit is set to '1' by software but can only be cleared to '0' by hardware after a reset. When WDGA=1, the watchdog can generate a reset. 0: Disabled; 1: Enable;
6: 0	T[6:0]	RW	7F	7-bit counter (MSB to LSB). This register is used to store the watchdog counter value. Decrements by 1 every (4096x2WDGTB) PCLK cycles. When the counter value changes from 40h to 3Fh (T[6] becomes 0), a watchdog reset is generated.

28.4.2. Configuration register (WWDG_CFR)

Address offset: 0x04

Reset value: 0x0000 007F

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

Res	Res	Res	Res	Res	Res	EWI	WDGTB[1:0]		T[6:0]						
-	-	-	-	-	-	RS	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 10	Reserved	-	-	Reserved
9	EWI	RS	0	Early wake-up interrupt. When this bit is set to 1, an interrupt is generated when the counter value reaches 40h. This bit can only be cleared by hardware after reset.
8: 7	WDGTB[1:0]	RW	0	Time base (Timer base). The time base of the prescaler is set as follows: 00: CK counter clock (PCLK divided by 4096) divided by 1 01: CK counter clock (PCLK divided by 4096) divided by 2 10: CK counter clock (PCLK divided by 4096) divided by 4 11: CK counter clock (PCLK divided by 4096) divided by 8
6: 0	W[6:0]	RW	7F	7-bit window value. This register contains the window value used for comparison with the down-counter.

28.4.3. Status Register (WWDG_SR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	EWIF
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	RC_W0

Bit	Name	R/W	Reset Value	Function
31: 1	Reserved	-	-	Reserved
0	EWIF	RC_W0	0	Early wakeup interrupt flag. When the counter value reaches 40h, this bit is set to 1 by hardware. Writing 0 clears it; writing 1 has no effect. This bit is also set to 1 when the interrupt is not enabled.

29. Real-Time Clock (RTC)

29.1. Introduction

The Real-Time Clock (RTC) is an independent timer. The RTC module has a set of continuously counting counters that, with appropriate software configuration, can provide clock and calendar functions. Modifying the counter value can reset the system's current time and date.

29.2. RTC main features

- Programmable prescaler coefficient: up to 220 division factor
- 32-bit programmable counter for measuring longer time periods
- Two separate clocks: PCLK for the APB interface and RTC clock (RTC clock frequency must be less than one-fourth of the PCLK clock frequency)
 - The following three RTC clock sources can be selected: — HSE clock divided by 128
 - LSI oscillator clock
 - LSE oscillator clock
- Two independent reset types:
 - The APB interface is reset by the system;
 - The RTC core (prescaler, alarm, counter, and divider) can only be reset by the backup domain.
- Three dedicated maskable interrupts: — Alarm interrupt, used to generate a software-programmable alarm interrupt — Second interrupt, used to generate a programmable periodic interrupt signal (up to 1 second maximum) — Overflow interrupt, indicating the internal programmable counter overflow and rollover to 0 state

29.3. RTC functional description

29.3.1. Overview

The RTC consists of two main components (see block diagram below). The first part (APB interface) is used to connect to the APB bus. This unit also contains a set of 16-bit registers (RTC_CRL and RTC_CRH, which are actually registers scattered across two addresses). In this chapter, the descriptions of all such register groups are suffixed with x (e.g., RTC_CRx), which can be read and written via the APB bus. The APB interface is driven by the APB bus clock and is used to connect to the APB bus.

The other part (RTC core) consists of a set of programmable counters, divided into two main modules. The first module is the RTC prescaler module, which can be programmed to generate the RTC time base TR_CLK with a maximum duration of 1 second. The RTC prescaler module includes a 20-bit programmable divider (RTC prescaler). If the corresponding enable bit is set in the RTC_CRH register, the RTC generates an interrupt (second interrupt) every TR_CLK cycle.

The second module is a 32-bit programmable counter that can be initialized to the current system time. The system time increments with the TR_CLK period and is compared with the programmable time stored in the RTC_ALRx register. If the corresponding enable bit is set in the RTC_CRH control register, an alarm interrupt is generated upon a match.

All interrupts can serve as wake-up signals for Stop mode. As shown below, in Stop mode, the corresponding RTC interrupt as a wake-up source requires pre-configuration of the corresponding interrupt enable bit and the EXTI line19 register.

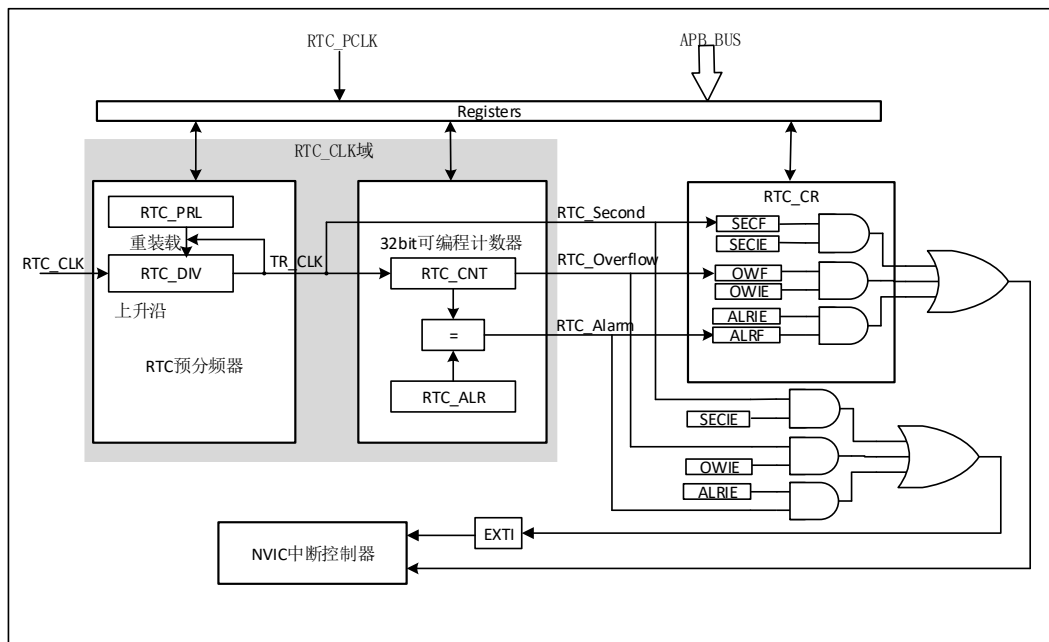


Figure 29-1 RTC Block Diagram

The DBP bit in the PWR_CR1 register is used to control the write protection disable for RTC registers. By default, DBP=0, which prevents write access to RTC registers. Only after software sets DBP can write operations to RTC registers be performed.

29.3.2. Reset RTC registers

The RTC_PRLx, RTC_ALRx, RTC_CNTx, and RTC_DIVx registers can only be reset by the RTC module soft reset or power reset signal. Other system registers (RTC_CRx) are asynchronously reset by system reset or power reset.

Note that when a reset source cannot reset the RTC module, it not only fails to reset the RTC module but also cannot reset the clock source enable signal sent to the RTC module or the clock source selection control sent to the RTC module.

29.3.3. Read RTC registers

The RTC core is completely independent of the RTC APB interface. Software accesses the RTC's prescaler value, counter value, and alarm value through the APB interface. However, the relevant readable registers are only updated when the signal synchronized from the RTC clock to the RTC APB clock is valid on the rising edge. The same applies to RTC flags.

This means that if the APB interface was previously disabled and a read operation is performed immediately after re-enabling the APB, the RTC register values read from the APB may be corrupted (typically reading as 0) until the first internal register update occurs. This situation can occur under the following circumstances:

- 1) A system reset or power reset has occurred.
- 2) The system has just woken up from Stop mode (in this case, the count value simply does not update because the CPU is inactive, the system clock is stopped, but the RTC continues counting normally, and the count value is not synchronized to the VDDD region).

In all the above cases, the RTC core remains operational even when the APB interface is disabled (during reset or when there is no clock).

Therefore, if the RTC's APB interface was previously disabled when reading RTC registers, the software must first wait for the RSF bit (Register Synchronization Flag) in the RTC_CRL register to be set to '1' by hardware.

Note:

- Registers readable by the CPU include RTC_CRx, RTC_CNTx, RTC_DIVx, and RTC_ALRx.
- The RTC_CRx register belongs to the RTC_PCLK domain, and the CPU can always read a stable value.
- RTC_CNTx and RTC_DIVx originate from the RTC_CLK domain. After the RTC starts operating, the RTC_DIVx register updates on every rising edge of RTC_CLK; RTC_CNTx and flags originating from the RTC_CLK clock domain also use the same update signal as the RTC_DIVx register, although the value of RTC_CNTx does not change with every update.
- RSF is implemented in the RTC_PCLK domain and is set when the pulse signal synchronized from RTC_CLK to RTC_PCLK is valid.
- RSF only controls the timing of reading RTC_CNTx and RTC_DIVx (hardware does not control this).

29.3.4. Configuring RTC registers

To write to the RTC_PRLx, RTC_CNTx, RTC_ALRx, or BKP_RTCCR registers, you must enter configuration mode by setting the CNF bit in the RTC_CRL register.

Additionally, any write operation to an RTC register must be performed after the previous write operation has completed. You can determine whether RTC registers are being updated by querying the RTOFF status bit in the RTC_CRL register. RTC registers can only be written when the RTOFF status bit is '1'.

Configuration process:

- 1) Poll the RTOFF bit until RTOFF becomes '1';
- 2) Set CNF to 1 to enter configuration mode;
- 3) Write to one or more RTC registers;
- 4) Clear the CNF flag to exit configuration mode;
- 5) Poll RTOFF until the RTOFF bit becomes '1' to confirm the write operation is complete.

Note: Write operations can only be performed when the CNF flag is cleared, and this process requires at least 3 RTC_CLK cycles. (After clearing the CNF flag, configuration cannot be restarted within 3 RTC_CLK cycles, otherwise a configuration error will occur (controlled by RTOFF=0 at this time))

Note:

- During this process, RTOFF=1 when the CPU writes to the register;
- The CPU write cycle is from CNF=1 to CNF=0, with other registers configured between these two operations;
- First write CNF to 1 and then clear CNF to 0. This operation clears RTOFF; writing only CNF=0 or writing CNF=1 without clearing will not clear RTOFF;
- First write CNF to 1 and then clear CNF to 0. This operation writes the cache register value to the RTC_CLK domain register;
- RTOFF is implemented in the RTC_PCLK domain;

29.3.5. Setting of RTC Flags

In each RTC_CLK clock cycle, before changing the RTC counter, the hardware sets the RTC second flag (SECF). During the last RTC clock cycle before the counter reaches 0x0000, the RTC overflow flag (OWF) is set.

During the RTC clock cycle before the counter value reaches the alarm register value plus 1 (ALR+1), the RTC_Alarm and RTC alarm flag (ALRF) are set. Writing to the RTC alarm must be synchronized with the RTC second flag using one of the following procedures:

- (1) Use the RTC alarm interrupt and modify the RTC alarm and/or RTC counter in the interrupt handler.
- (2) Wait for the SECF bit in the RTC control register to be set before changing the RTC alarm and/or RTC counter.

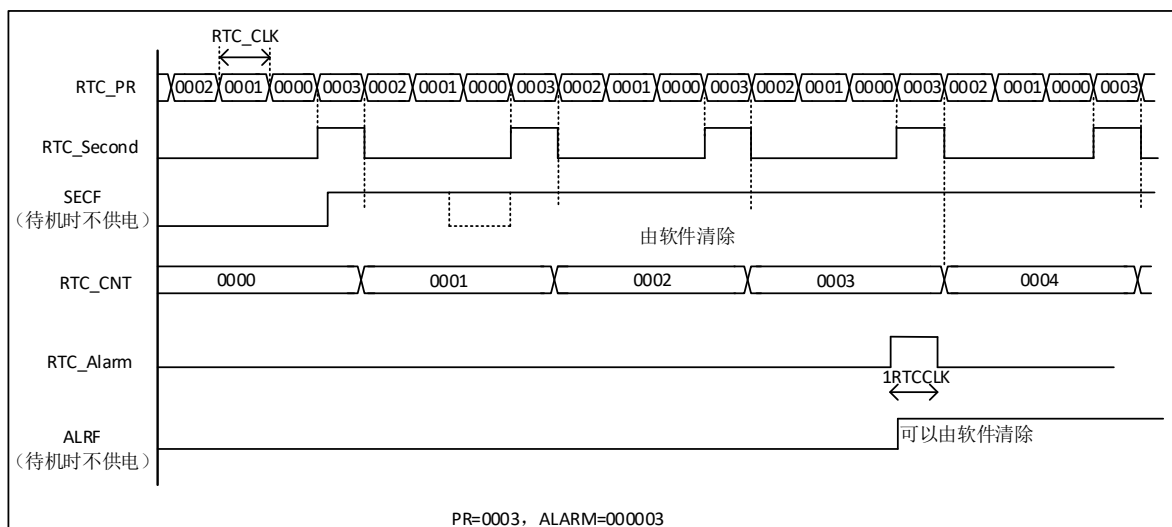


Figure 29-2 Example of RTC Second and Alarm Waveform, PR=0003, ALR=00003

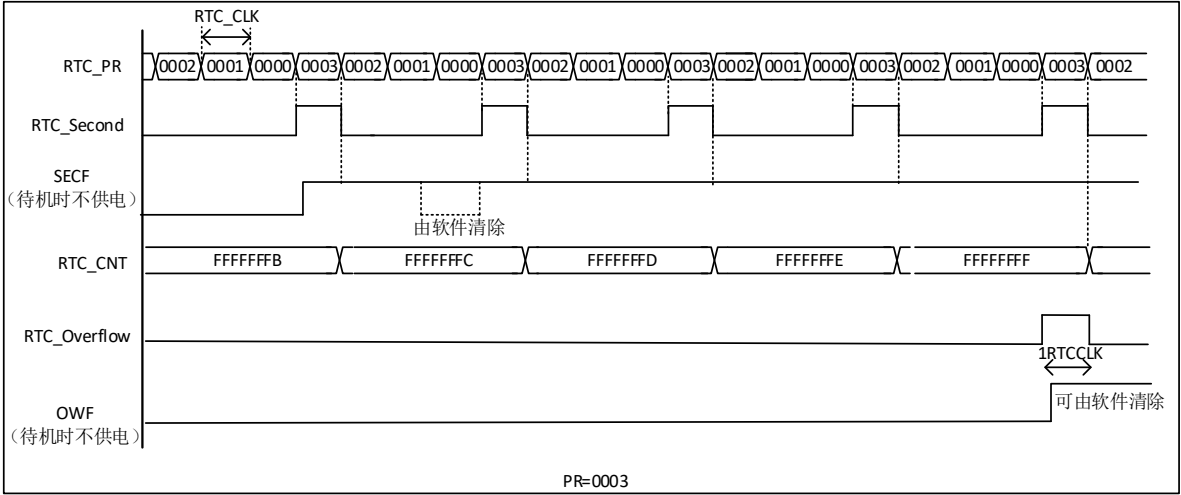


Figure 29-3 Example of RTC Overflow Waveform, PR=0003

29.3.6. RTC Calibration

For measurement purposes, the 64-division of the RTC clock can be output to the IO pin (PF5). This function is achieved by setting the CCO bit (BKP_RTCCR register).

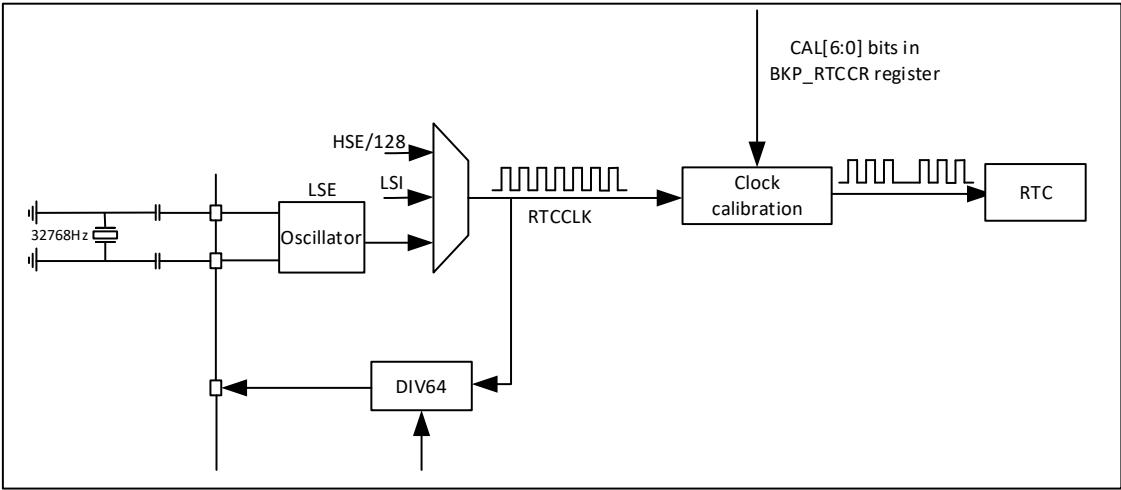


Figure 29-4 RTC Calibration Diagram

29.4. RTC Register

29.4.1. RTC Control Register (RTC_CRH)

Address offset: 0x00

Reset value: 0x0000

This register is reset by system reset.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	OWIE	ALR IE	SEC IE
-	-	-	-	-	-	-	-	-	-	-	-	-	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 3	Reserved	-	-	Reserved
2	OWIE	RW	0	Overflow interrupt enable bit 0: Overflow interrupt disabled 1: Overflow interrupt enabled
1	ALRIE	RW	0	Alarm interrupt enable bit 0: Disable alarm interrupt 1: Enable alarm interrupt
0	SECIE	RW	0	Second interrupt enable bit 0: Disable second interrupt 1: Enable second interrupt

These bits are used to mask interrupt requests. Note: After reset, all interrupts are disabled, so after initialization, it is possible to write to RTC registers to ensure no pending interrupt requests exist. However, the RTC_CRH register cannot be written when the peripheral is completing a previous write operation (RTOFF=0).

This control register governs the functionality of the RTC. Certain bits can only be written using a specific configuration sequence.

29.4.2. RTC Control Register (RTC_CRL)

Address offset: 0x04

Reset value: 0x0020

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res										RTOFF	CNF	RSF	OWF	ALRF	SECF
-										R	RW	RC_W0	RC_W0	RC_W0	RC_W0

Bit	Name	R/W	Reset Value	Function
31: 6	Reserved	-	-	Reserved
5	RTOFF	R	1	RTC operation OFF (RTC operation OFF), this bit is read-only.

				<p>The RTC module uses this bit to indicate the status of the last operation performed on its registers (indicating whether the operation is complete).</p> <p>If this bit is '0', it indicates that no write operation can be performed on any RTC register.</p> <p>0: The last write operation to the RTC registers is still in progress</p> <p>1: The last write operation to the RTC registers has been completed</p>
4	CNF	RW	0	<p>Configuration flag (Configuration flag) This bit must be set to '1' by software to enter configuration mode, allowing new values to be written to the RTC_CNTx, RTC_ALRx, or RTC_PRLx registers.</p> <p>A write operation will only be executed when this bit is set to '1' and then cleared to '0' again by software.</p> <p>0: Exit configuration mode (start updating RTC registers)</p> <p>1: Enter configuration mode</p>
3	RSF	RC_W0	0	<p>Registers synchronized flag (Registers synchronized flag)</p> <p>When the RTC_CNTx register and RTC_DIVx register are updated, hardware sets this bit to '1', and software clears it to '0'.</p> <p>After an APB reset or when the APB clock is stopped, this bit must be cleared to '0' by software.</p> <p>Before performing any read operation, the user program must wait for this bit to be set to '1' by hardware to ensure that RTC_CNTx, RTC_ALRx, or RTC_PRLx has been synchronized.</p> <p>0: Register has not been synchronized</p> <p>1: Register has been synchronized</p>
2	OWF	RC_W0	0	<p>Overflow flag When the 32-bit programmable counter overflows, this bit is set to '1' by hardware. If OWIE=1 in the RTC_CRH register, an interrupt is generated.</p> <p>This bit can only be cleared to '0' by software; writing '1' has no effect. 0: No overflow; 1: 32-bit programmable counter overflow</p>

1	ALRF	RC_W0	0	Alarm flag When the 32-bit programmable counter reaches the preset value set in the RTC_ALRx register, this bit is set to '1' by hardware. If ALRIE=1 in the RTC_CRH register, an interrupt is generated. This bit can only be cleared to '0' by software; writing '1' has no effect. 0: No alarm 1: Alarm exists
0	SECF	RC_W0	0	Second flag When the 32-bit programmable prescaler overflows, this bit is set to '1' by hardware, and the RTC counter increments by 1. Therefore, this flag provides a periodic signal (usually 1 second) for the resolution-programmable RTC counter. If SECIE=1 in the RTC_CRH register, an interrupt is generated. This bit can only be cleared by software; writing '1' has no effect. 0: Second flag condition not met 1: Second flag condition met

The functionality of the RTC is controlled by this control register. The RTC_CRx registers cannot be written when the peripheral is still completing the previous write operation (RTOFF=0).

Note:

- Any flag will remain pending until reset by software, indicating that the requested interrupt has been acknowledged.
- All interrupts are disabled upon reset, with no pending interrupt requests, allowing write operations to RTC registers (referring to writes from the buffer register to the RTC_CLK domain registers).
- When the APB clock is not running, the OWF, ALRF, SECF, and RSF bits are not updated (cannot be synchronized).
- The OWF, ALRF, SECF, and RSF bits can only be set by hardware and cleared by software.

29.4.3. RTC Prescaler Load Register (RTC_PRLH)

The PRL register holds the periodic count value of the RTC prescaler. This register is write-protected by the RTOFF bit in the RTC_CRL register. The CPU can only perform write operations (to the buffer register) when RTOFF=1.

Address offset: 0x08

Reset value: 0x0000

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

Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	PRL[19:16]			
-	-	-	-	-	-	-	-	-	-	-	-	W	W	W	W

Bit	Name	R/W	Reset Value	Function
31: 4	Reserved	-	-	Reserved
3: 0	PRL[19:16]	W	0	RTC Prescaler Reload Value High (RTC pre-scaler reload value high) According to the following formula, these bits define the clock frequency of the counter: $f_{TR_CLK} = f_{RTC_CLK} / (PRL[19:0] + 1)$ Note: Using a value of 0 is not recommended, as it may prevent correct generation of RTC interrupts and flags.

29.4.4. RTC Prescaler Divider Register (RTC_PRL)

Address offset: 0x0C

Reset value: 0x8000

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

Bit	Name	R/W	Reset Value	Function
31: 16	Reserved	-	-	Reserved
15: 0	PRL[15:0]	W	0x8000	RTC Prescaler Reload Value High (RTC pre-scaler reload value high) According to the following formula, these bits define the clock frequency of the counter: $f_{TR_CLK} = f_{RTC_CLK} / (PRL[19:0] + 1)$ Note: Using a value of 0 is not recommended, as it may prevent correct generation of RTC interrupts and flags.

29.4.5. RTC Prescaler Divider Factor Register High (RTC_DIVH)

During each TR_CLK cycle, the value of the RTC_PRLx register is reloaded into the RTC prescaler counter. Users can read the RTC_DIVx register to obtain the current value of the prescaler counter without stopping its operation, thereby achieving precise time measurement.

This register is read-only. When any change occurs in the value of RTC_PRLx or RTC_CNTx registers, the value of this register will be reloaded by hardware.

Address offset: 0x10

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 4	Reserved	-	-	Reserved
3: 0	RTC_DIV[19:16]	R	0	RTC clock divider high.

29.4.6. RTC prescaler divider factor register low (RTC_DIVL)

Address offset: 0x14

Reset value: 0x0000 8000

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

Bit	Name	R/W	Reset Value	Function
31: 16	Reserved	-	-	Reserved
15: 0	DIV[15:0]	R	0x8000	RTC Clock Divider Low

29.4.7. RTC Counter Register High (RTC_CNTH)

The RTC module has a 32-bit programmable counter. This register is accessed through two 16-bit registers, and counting is based on the TR_CLK time reference generated by the prescaler.

The RTC_CNTx register holds the count value of the counter. The register is write-protected and can only be written when RTOFF=1. Writing to the high 16-bit RTC_CNTH or low 16-bit RTC_CNTL register directly loads the value into the corresponding programmable counter and reloads the RTC prescaler.

When a read operation occurs, the current value of the counter (system time) is returned.

Address offset: 0x18

Reset value: 0x0000

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

Bit	Name	R/W	Reset Value	Function
31: 16	Reserved	-	-	Reserved
15: 0	RTC_CNT[31:16]	RW	0x0000	Upper 16 bits of the RTC counter When reading the RTC_CNTH register, it returns the upper 16 bits of the current value of the RTC counter register. Write operations to this register are only allowed in configuration mode.

29.4.8. RTC counter register low bits (RTC_CNTL)

Address offset: 0x1C

Reset value: 0x0000

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

Bit	Name	R/W	Reset Value	Function
31: 16	Reserved	-	-	Reserved
15: 0	RTC_CNT[15:0]	RW	0x0000	RTC counter lower 16 bits When reading the RTC_CNTL register, the lower 16 bits of the current value of the RTC counter register are returned. Write operations to this register are only allowed in configuration mode.

29.4.9. RTC Alarm Register High (RTC_ALRH)

When the programmable counter (count) reaches the 32-bit value stored in the RTC_ALRx register, an alarm interrupt request is generated. This register is write-protected by the RTOFF bit. Write access is only permitted when RTOFF=1.

Address offset: 0x20

Reset value: 0xFFFF

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

Bit	Name	R/W	Reset Value	Function
31: 16	Reserved	-	-	Reserved
15: 0	ALR[31:16]	RW	0xFFFF	RTC alarm register high 16 bits Software can write the high 16 bits of the alarm time. Writing to this register must enter configuration mode.

29.4.10. RTC alarm register high (RTC_ALRL)

Address offset: 0x24

Reset value: 0x0000 FFFF

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

Bit	Name	R/W	Reset Value	Function
31: 16	Reserved	-	-	Reserved
15: 0	ALR[15:0]	RW	0xFFFF	RTC alarm register lower 16 bits Software can write the lower 16 bits of the alarm time. Writing to this register must enter configuration mode.

29.4.11. RTC clock calibration and output configuration register

(BKP_RTCCR)

Address offset: 0x2C

Reset value: 0x0000 0000 (can only be reset by por and bdcr soft reset)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	ASOS	ASOE	CCO	CAL[6:0]						
-	-	-	-	-	-	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 10	Reserved	-	-	Reserved
9	ASOS	RW	0	Second/alarm pulse output selection bit When ASOE bit is set, ASOS bit can be used to select whether the pin outputs RTC second pulse or alarm signal 0: RTC alarm pulse signal 1: RTC second pulse signal
8	ASOE	RW	0	Second/alarm pulse output enable bit When this bit is set, the ASOS bit determines whether the RTC_OUT pin outputs RTC second pulse or alarm pulse signal.
7	CCO	RW	0	Calibration clock output bit When ASOE is not set, CCO can be set to output 1/64 division of RTC clock 0: No effect 1: Output 1/64 division of RTC clock on pin
6: 0	CAL[6:0]	RW	0	Calibration value This value shows the number of negligible clock pulses per 220 clocks, allowing RTC calibration to slow the clock in steps of 1000000/220 PPM. RTC_CLK can be slowed down from 0 to 121 PPM.

30. I2C interface

30.1. Introduction

The I2C (Inter-Integrated Circuit) bus interface connects microcontrollers to the serial I2C bus. It provides multi-master capability, controlling all I2C bus-specific sequences, protocols, arbitration, and timing. It supports both Standard (Sm) and Fast (Fm) modes.

DMA can be used to offload the CPU according to specific device requirements.

30.2. I2C key features

- Slave and master modes
- Multi-master capability
- Supports different communication speeds
 - Standard mode (Sm): up to 100 kHz
 - Fast mode (Fm): up to 400 kHz
- As master
 - Clock generation
 - Start and stop generation
- As slave
 - Programmable I2C address detection
 - Dual address capability supporting 2 slave addresses
 - Stop condition detection
- 7-bit/10-bit addressing mode
- Supports General call functionality
- Status flag
 - Transmit/Receive mode flag
 - Byte transfer complete flag
 - I2C bus busy flag
- Error flags
 - Master arbitration loss
 - ACK failure after address/data transmission
 - Start and stop errors
 - Overrun/Underrun (clock stretching disabled)
- Optional clock stretching function
- Single-byte buffer with DMA capability
- Software Reset
- Analog noise filtering function
- Configurable PEC (Packet Error Checking) generation and verification
 - PEC value can be sent in the last byte in Tx mode

- Receive last byte for PEC error checking
- SMBus compatible
 - 25 ms clock low timeout delay
 - 10 ms master cumulative clock low extension time
 - 25 ms slave cumulative clock low extension time
 - Hardware PEC generation and verification with ACK control
 - Supports Address Resolution Protocol (ARP)

30.3. I2C functional description

30.3.1. I2C block diagram

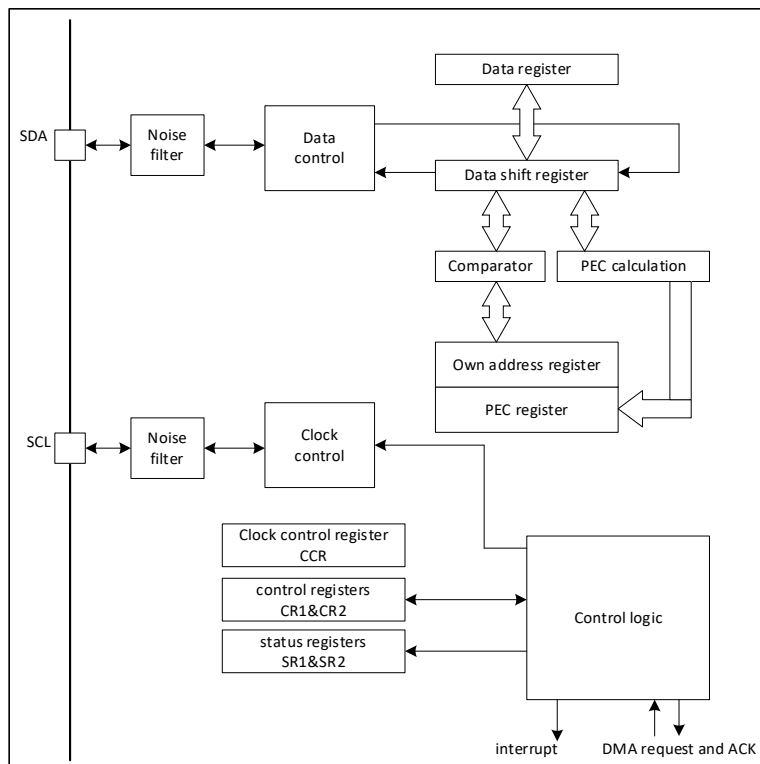


Figure 30-1 I2C block diagram

30.3.2. Mode selection

I2C supports the following four modes:

- Slave transmitter mode
- Slave receiver mode
- Master transmitter mode
- Master receiver mode

By default, it operates in slave mode. The interface automatically switches from slave to master mode after generating a start bit and reverts to slave mode upon arbitration loss or stop bit generation, enabling multi-master functionality.

30.3.2.1. Communication flow

As a master, the I2C interface initiates data transfer and generates the clock signal. Serial data transmission always begins with a start condition and ends with a stop condition. Start and stop conditions are generated under software control in master mode.

As a slave, the I2C interface can recognize its own address (7-bit/10-bit) and the general call address. Software can control whether to recognize the general call address.

Data and addresses are transmitted in 8-bit (byte) format, with the most significant bit first. The first byte following the start condition is the address. Addresses are only transmitted in master mode.

During the 9th clock cycle after 8 clocks of a byte transfer, the receiver must send an acknowledgment bit (ACK) back to the transmitter. Refer to the figure below.

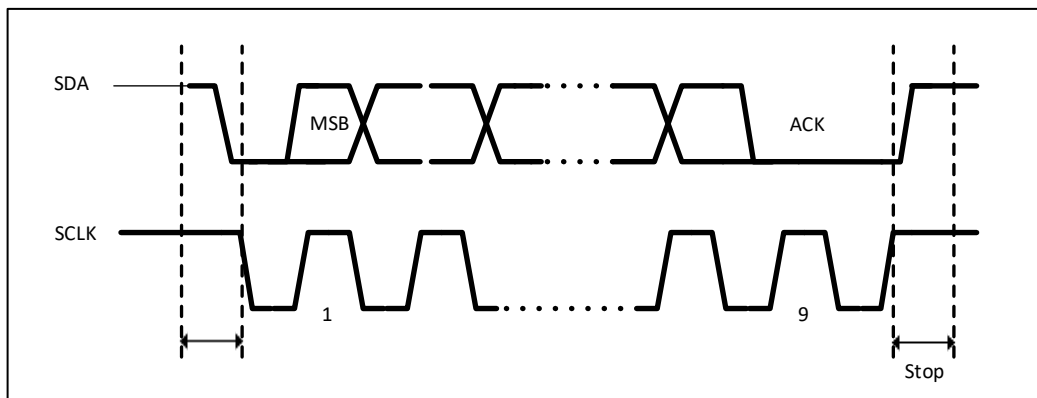


Figure 30-2 I2C bus protocol

Software can enable or disable the acknowledgment (ACK) bit. Software can also select the I2C interface address (7-bit, 10-bit, or general call address).

30.3.3. I2C initialization

■ Enable/disable I2C module

The I2C clock module must first be enabled via the I2C_EN bit in the RCC_APBENR1 register, then the I2C module is activated by setting the PE bit in I2C_CR1.

■ I2C timing configuration

The hold and setup time requirements for the data signal (SDA) must comply with the I2C standard protocol, requiring I2C timing configuration. This is achieved by writing to the I2C_CCR and I2C_TRISE registers.

30.3.4. I2C slave mode

By default, the I2C interface always operates in slave mode. Switching from slave mode to master mode requires generating a start condition.

To generate correct timing, the input clock of this module must be set in the I2C_CR2 register. The input clock frequency must be at least:

- 4 MHz in standard mode
- 8 MHz in fast mode

Once the start condition is detected, the address received on the SDA line is sent to the shift register and compared with the I2C address OAR1 or the general call address (if ENGC=1).

Header or address mismatch:

The I2C interface ignores it and waits for another start condition.

Address match:

The I2C interface generates the following sequence:

- If ACK is set to '1' by software, an acknowledge pulse is generated
- Hardware sets the ADDR bit, and an interrupt is generated if ITEVTEN bit is set

In slave mode, the TRA bit indicates whether the current mode is receiver or transmitter.

30.3.4.1. Slave transmitter

After receiving the address and clearing the ADDR bit, (if the least significant bit of the address byte is 1) the slave sends data (byte) from the DR register to SDA via the internal shift register.

The slave pulls SCL low until the ADDR bit is cleared and the data to be sent is written to the DR register.

When the acknowledge pulse is received: The TxE bit is set by hardware, and an interrupt is generated if ITEVTEN and ITBUFEN bits are set.

If the TxE bit is set but no new data is written to the I2C_DR register before the next data transmission ends, the BTF bit is set. The slave pulls SCL low until the BTF bit is cleared by software (after reading I2C_SR1, then writing to I2C_DR register).

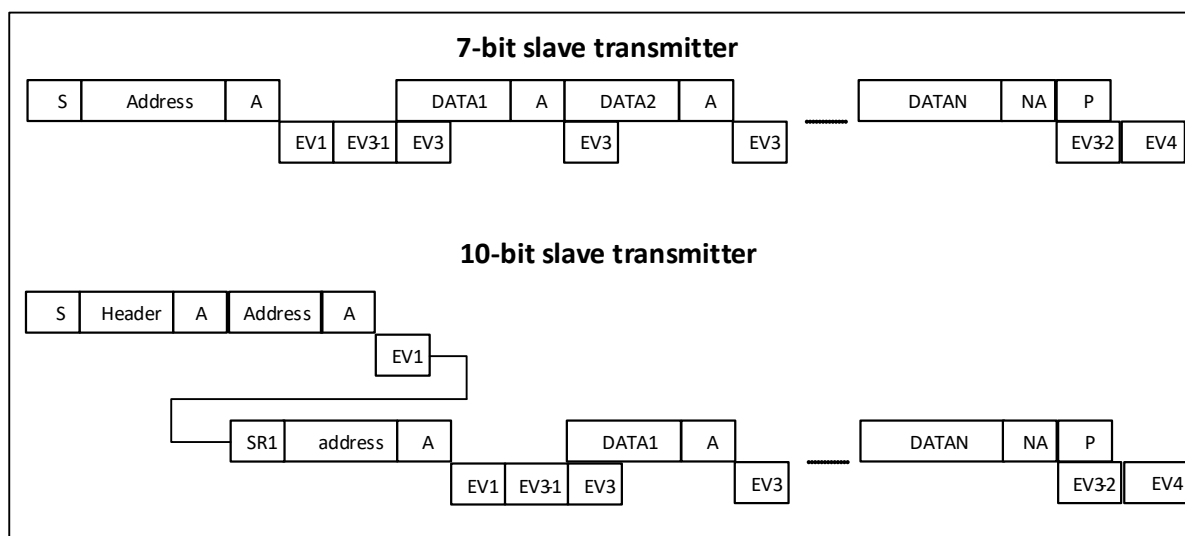


Figure 30-3 Transmission sequence diagram from transmitter

Legend: S= Start condition, Sr= Repeated Start condition, P= Stop condition, A= Acknowledge, NA= Non-acknowledge, EVx= Event (interrupt generated when ITEVFEN=1)

EV1: ADDR=1, Clear ADDR bit by first reading SR1 register, then reading SR2 register

EV3-1: TxE=1, shift register is empty, data register is empty, writing Data1 to the DR register.

EV3: TxE=1, shift register is not empty, data register is empty, writing (Data2) to the DR register clears TxE.

EV3-2: AF=1; Software writes 0 to the AF bit to clear it.

EV4: STOPF=1, cleared by first reading the SR1 register and then writing to the CR1 register.

Note:

- 1) The EV1 and EV3_1 events pull SCL low until the corresponding software sequence ends.
- 2) The EV3 software sequence must be completed before the current byte transfer ends.

30.3.4.2. Slave receiver

After receiving the address and clearing ADDR, (if the least significant bit of the address byte is 0) the slave will store the byte received from the SDA line into the DR register via the internal shift register. The I2C interface performs the following operations after receiving each byte:

- If the ACK bit is set, an acknowledgment pulse is generated.
- Hardware sets RxNE=1. An interrupt is generated if the ITEVTEN and ITBUFEN bits are set.

If RxNE is set and the DR register is not read before the reception of new data ends, the BTF bit is set, and the slave keeps SCL low until BTF is cleared (by reading I2C_SR1 followed by the I2C_DR register). (See the figure below).

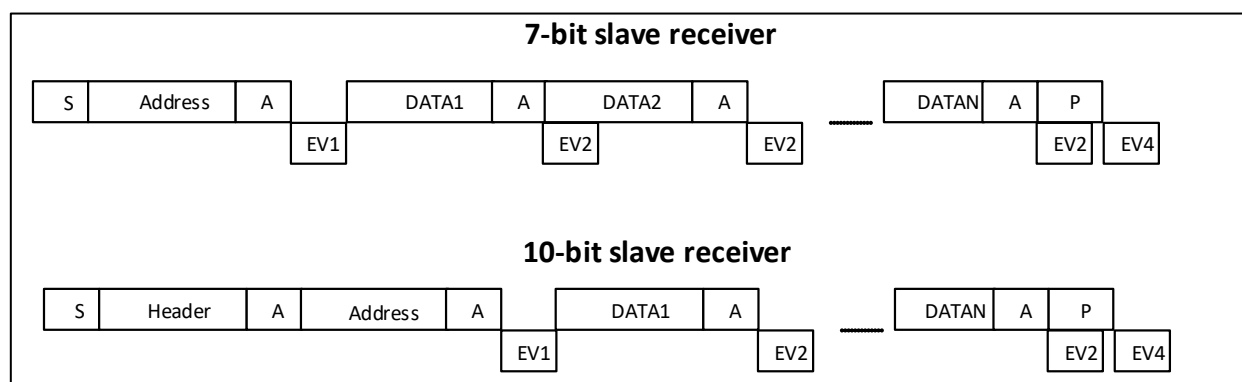


Figure 30-4 Transmission sequence diagram from the receiver

Legend: S= Start, Sr= Repeated Start, P= Stop, A= Acknowledged, EVx= Event (interrupt generated when ITEVFEN= 1).

EV1: ADDR=1, cleared by first reading the SR1 register and then reading the SR2 register.

EV2: RxNE=1, cleared by reading the DR register.

EV4: STOPF=1, cleared by first reading the SR1 register and then writing to the CR1 register.

Note:

- EV1 timing pulls SCL low until the corresponding software sequence ends.
- The EV2 software sequence must be completed before the current byte transfer finishes.
- After checking the SR1 register contents, the user should perform a complete clearing sequence for each flagged bit found set. For example, the ADDR and STOPF flags require the following sequence:

If ADDR=1, first read SR1, then read SR2; if STOPF=1, first read SR1, then write CR1. This is to ensure that if both ADDR and STOPF bits are found set, they can both be cleared.

30.3.4.3. Close communication.

After transmitting the last data byte, the master generates a stop condition, and when the slave detects this condition:

- The hardware sets STOPF, and if the ITEVTEN bit is set, an interrupt is generated.
- Clear the STOPF bit by first reading SR1 and then writing CR1.

30.3.5. I2C master mode

In master mode, the I2C interface initiates data transmission and generates the clock signal. Serial data transmission always starts with a start condition and ends with a stop condition.

When the start condition is generated on the bus via the START bit, the device enters master mode.

The following is the required sequence of operations for master mode:

- Set the input clock of this module in the I2C_CR2 register to generate the correct timing
- Configure the I2C_CCR register
- Configure the I2C_TRISE register
- Program the I2C_CR1 register to enable the peripheral
- Set the START bit in the I2C_CR1 register to 1 to generate the start condition

The input clock frequency of the I2C module must be at least:

- 4 MHz in standard mode
- 8 MHz in fast mode

30.3.5.1. The master generates the clock

The CCR register generates the high and low levels of SCL based on the rising or falling edges. Since the slave may stretch the SCL signal, after the SCL rising edge is generated, the master checks the SCL signal from the bus when the time programmed in the TRISE register is reached.

- If SCL is low level, it means the slave is stretching the SCL bus, and the high-level counter stops counting until SCL is detected as high level. This is to ensure the minimum high-level time of the SCL parameter.
- If SCL is high level, the high-level counter keeps counting.

In practice, even if the slave does not stretch SCL, the feedback loop takes some time from the generation of the SCL rising edge to its detection. This loop time is related to the SCL rise time (SCL VIH data detection), plus the analog noise filtering of the SCL input path and the internal chip synchronization of SCL using the APB clock. The maximum feedback loop time is programmed in the TRISE register, so the SCL frequency remains stable regardless of the SCL rise time.

30.3.5.2. Start condition

When BUSY=0, setting START=1 causes the I2C interface to generate a start condition and switch to master mode (MSL is set).

Note: Setting the START bit in master mode will generate a repeated start condition by hardware after the current byte transmission is completed.

Once the start condition is issued:

- The SB bit is set by hardware, and if the ITEVTEN bit is set, an interrupt is generated.

The master reads the I2C_SR1 register and then writes the slave address to the I2C_DR register.

30.3.5.3. Slave address transmission

The master sends the slave address to the SDA line through the internal shift register.

- In 10-bit address mode, sending a header sequence generates the following events:
 - The ADDR10 bit is set by hardware, and if the ITEVTEN bit is set, an interrupt is generated.

The master device then waits to read the I2C_SR1 register, followed by writing the second address byte to the DR register.

The ADDR bit is set by hardware, and if the TEVFEN bit is set, an interrupt is generated.

The master device then waits to read the I2C_SR1 register, followed by reading the I2C_SR2 register.

- In 7-bit address mode, an address byte is sent.

Once the address byte is sent,

- The ADDR bit is set by hardware, and if the ITEVTEN bit is set, an interrupt is generated.

The master then reads the I2C_SR1 register, followed by reading the I2C_SR2 register.

Based on the least significant bit of the sent slave address, the master decides whether to enter transmitter mode or receiver mode.

- In 7-bit address mode,

- To enter transmitter mode, the master device sets the least significant bit to '0' when sending the slave address.

- To enter receiver mode, the master device sets the least significant bit to '1' when sending the slave address.

The TRA bit indicates whether the master is in receiver mode or transmitter mode.

- In 10-bit address mode,

- To enter transmitter mode, the master first sends the header byte (11110xx0), then sends the slave address with LSB equal to 0. (xx in the header byte are the highest 2 bits of the 10-bit address).

- To enter receiver mode, the master first sends the header byte (11110xx0), then sends the slave address with LSB equal to 0. Then send a start condition again, followed by the header byte (11110xx1).

(xx in the header byte are the highest 2 bits of the 10-bit address). The TRA bit indicates whether the master is in receiver mode or transmitter mode.

30.3.5.4. Master transmitter

After sending the address and clearing the ADDR bit, the master sends the byte from the DR register to the SDA line through the internal shift register.

The master waits until the first data byte is written to the I2C_DR register (see EV8_1).

When the ACK pulse is received, the TxE bit is set by hardware, and if INEVFEN and ITBUFEN bits are set, an interrupt is generated.

If TxE is set and no new data byte is written to the DR register before the previous data transmission ends, BTF is set by hardware. Before clearing BTF (after reading I2C_SR1, then writing I2C_DR register), the I2C interface will keep SCL low.

Close communication.

After writing the last byte to the DR register, generate a stop condition by setting the STOP bit (see EV8_2 in the figure), then the I2C interface will automatically return to slave mode (MSL bit cleared).

Note: When TxE or BTF bit is set, the stop condition should be scheduled when EV8_2 event occurs.

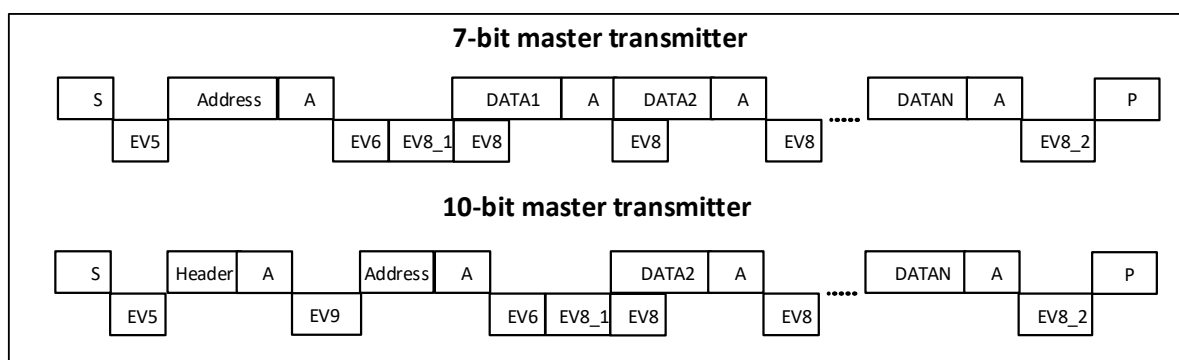


Figure 30-5 Master transmitter transfer sequence diagram

Legend: S= Start, Sr= Repeated Start, P= Stop, A= Acknowledge, EVx= Event (generates interrupt when ITEVFEN=1)

EV5: SB=1, read SR1 then write address to DR register to clear this event

EV6: ADDR=1, read SR1 then read SR2 to clear this event

EV8_1: TXE=1, shift register empty

EV8: TXE=1, write DR register to clear this event

EV8_2: TXE=1, BTF=1, cleared by hardware when stop condition is generated

EV9: ADDR10=1, read SR1 then write DR register to clear this event

Note:

- 1) During EV5, EV6, EV8_1, and EV8_2 events, the SCL low level is stretched until the corresponding software sequence is completed.
- 2) The EV8 software sequence must be completed before the current byte transmission finishes. If the EV8 software sequence cannot be completed before the end of the current byte transmission, it is recommended to use BTF instead of TxE.

30.3.5.5. Master receiver.

After sending the address and clearing ADDR, the I2C interface enters master receiver mode. In this mode, the I2C interface receives data bytes from the SDA line and transfers them to the DR register via the internal shift register. After each byte, the I2C interface sequentially performs the following operations:

- If the ACK bit is set, an acknowledgment pulse is issued.
- Hardware sets RxNE=1; if INEVFEN and ITBUFEN bits are set, an interrupt will be generated.

If the RxNE bit is set and the data in the DR register is not read before the end of new data reception, the hardware will set BTF=1; the I2C interface will hold SCL low until BTF is cleared. Reading I2C_SR1 followed by reading I2C_DR register will clear the BTF bit.

Close communication.

Method 1: This method is applied when I2C is configured as the highest priority interrupt in the application.

The master sends a NACK after receiving the last byte from the slave. Upon receiving NACK, the slave releases control of the SCL and SDA lines. The master can then send a stop/repeated start condition.

- 1) To generate a NACK pulse after receiving the last byte, the ACK bit must be cleared after reading the second-to-last data byte (after the second-to-last RxNE event).

- 2) To generate a stop/repeated start condition, the software must set the stop/start bit after reading the second-to-last data byte (after the second-to-last RxNE event).
- 3) When receiving a single byte, the bits for disabling acknowledgment and generating stop condition must be cleared right after EV6 (after clearing ADDR during EV6_1).

After generating a stop condition, the I2C interface automatically returns to slave mode (MSL bit is cleared).

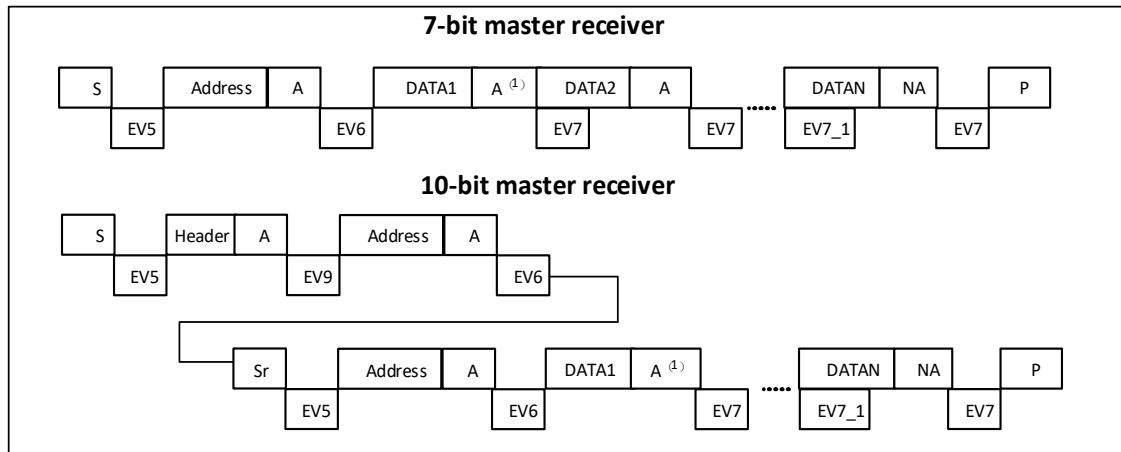


Figure 30-6 Method 1: Timing diagram for master mode transmission.

Legend: S= Start, Sr= Repeated Start, P= Stop, A= Acknowledge, EVx= Event (generates interrupt when ITEVFEN=1)

EV5: SB=1, read SR1, then write to the DR register, this bit is cleared.

EV6: ADDR=1, read SR1, then read SR2, this bit is cleared

EV6_1: No related flag event, only used for single-byte reception

EV7: RxNE=1, read the DR register, this bit is cleared

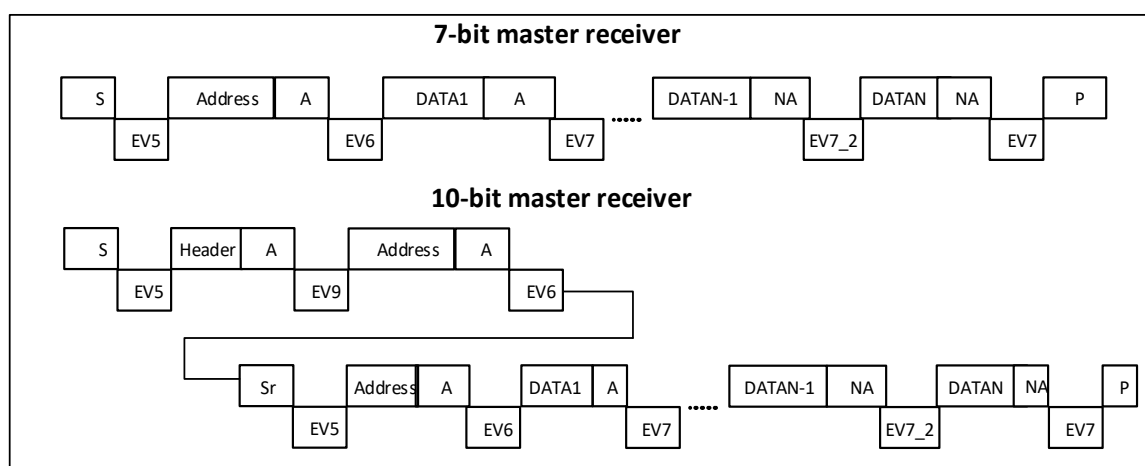
EV7_1: RxNE=1, read the DR register, write ACK=0 and set STOP

EV9: ADDR10=1, read SR1 then write DR register to clear the event

- 1) If it is a single-byte reception, the part marked as (1) above will be NA.
- 2) EV5, EV6 events stretch the SCL low level until the corresponding software sequence completes
- 3) EV7 software sequence must complete before current byte transmission finishes. If EV7 software sequence cannot complete before current byte transmission ends, using BTF instead of RXNE is recommended.
- 4) The software sequence for EV6_1 or EV7_1 must be completed before the ACK of the current byte transfer.

Method 2: This method is applicable when the I2C interrupt is not the highest priority in the application or when polling is used.

With this method, DataN-2 is not read, so after DataN-1, the communication is extended (both RxNE and BTF are set). Then, before reading DataN-2 from the DR register, clear the ACK bit to ensure it is cleared before DataN ACK. After this, following the reading of DataN-2, set the STOP/START bit and read DataN-1. After RxNE is set, read DataN

Figure 30-7 Method 2: Timing diagram for master transmission when $N > 2$

Legend: S= Start, Sr= Repeated Start, P= Stop, A= Acknowledge, EVx= Event (generates interrupt when ITEVFEN=1)

EV5: SB=1, first read SR1 register, then write DR register to clear this bit

EV6: ADDR, first read SR1, then read SR2, clear this bit

EV7: RxNE=1, read DR register to clear this bit

EV7_2: BTF=1, DataN-2 is in the DR register, DataN-1 is in the shift register, write ACK=0, read DataN-2 from the DR register. Set STOP, read DataN-1

EV9: ADDR10=1, read SR1 then write DR register to clear the event

Note:

- EV5, EV6 events stretch the SCL low level until the corresponding software sequence completes
- EV7 software sequence must complete before current byte transmission finishes. If EV7 software sequence cannot complete before current byte transmission ends, using BTF instead of RXNE is recommended.

■ **When 3 bytes need to be read:**

- RxNE=1 => No action (DataN-2 not read).
- DataN-1 received
- BTF=1, both shift and data registers are full: DR register holds DataN-2, shift register holds DataN-1 => SCL pulled low: no more data to receive on bus
- Clear ACK bit
- Read DataN-2 from DR register => This initiates shift register reception of DataN
- DataN reception complete (send NACK condition)
- Write START or STOP bit
- Read DataN-1
- RxNE=1
- Read DataN

The above procedure describes cases where $N > 2$. Different handling is required for receiving 1 byte vs 2 bytes, as described below:

■ **Case of receiving 2 bytes**

- Set POS and ACK bits

- Wait for ADDR to be set
- Clear ADDR bit
- Clear ACK bit
- Wait for BTF to be set
- Write STOP bit
- Read DR twice

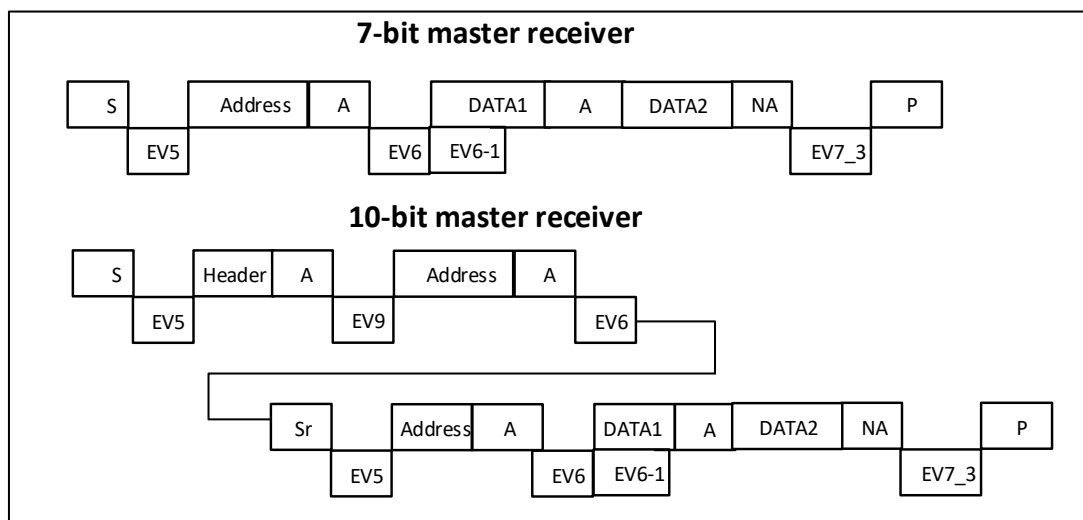


Figure 30-8 Method 2: Timing diagram for master mode transmission when N=2

Legend: S= Start, Sr= Repeated Start, P= Stop, A= Acknowledge, EVx= Event (generates interrupt when ITEVFEN=1)

EV5: SB=1, first read SR1 register, then write DR register to clear this bit

EV6: ADDR=1, first read SR1 register, then read SR2 register to clear the ADDR bit

EV6_1: No relevant flag event. After EV6, i.e., after the address is cleared, ACK should be cleared

EV7_3: BTF=1, write STOP=1, then read DR twice (Data1 and Data2)

EV9: ADDR10=1, read SR1 then write DR register to clear the event Note:

- EV5, EV6 events stretch the SCL low level until the corresponding software sequence completes
- The EV6_1 software sequence must complete before the ACK of the current byte transfer

■ Single-byte reception case

- In the ADDR event, clear the ACK bit
- Clear ADDR
- Write STOP or START bit
- Read data after the RxNE flag is set

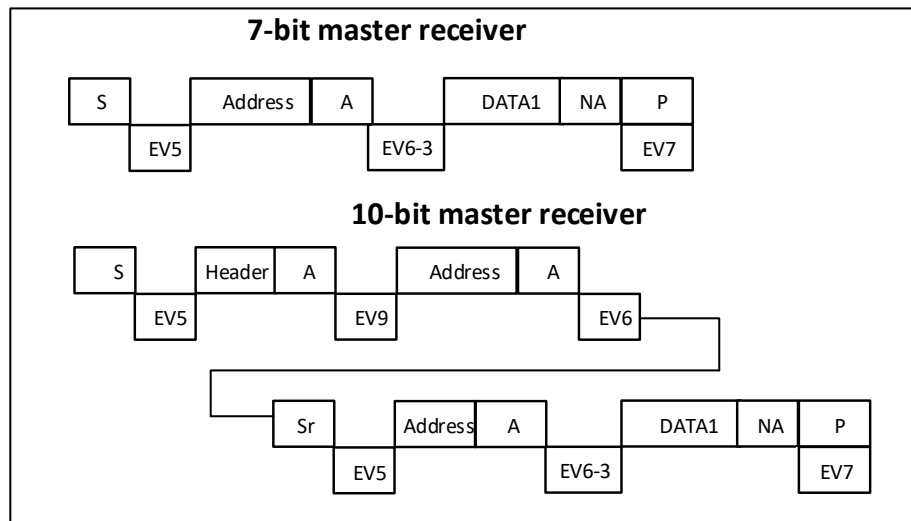


Figure 30-9 Method 2: Timing for master transmission when N=1

Legend: S= Start, Sr= Repeated Start, P= Stop, A= Acknowledge, EVx= Event (interrupt generated when ITEVFEN= 1)

EV5: SB=1, first read SR1 register, then write DR register to clear this bit

EV6_3: ADDR=1, write ACK=0. First read SR1 register, then read SR2 register to clear the ADDR bit. After ADDR is cleared, write STOP=1

EV7: RxNE=1, read DR register to clear this bit

EV9: ADDR10=1, read SR1 then write DR register to clear the event

Note:

EV5, EV6, EV8_1, and EV8_2 events stretch the SCL low level until the corresponding software sequence completes.

30.3.6. Error Status

30.3.6.1. Bus Error (BERR)

A bus error occurs when the I2C interface detects an external stop or start condition during an address or data byte transmission. At this time:

- The BERR bit is set to '1'; if the ITERREN bit is set, an interrupt is generated;
- In slave mode: data is discarded, and the hardware releases the bus:
 - If it is an erroneous start condition, the slave considers it a restart and waits for an address or stop condition
 - If it is an erroneous stop condition, the slave operates as a normal stop condition, and the hardware releases the bus
- In master mode: the hardware does not release the bus and does not affect the current transmission status. At this point, it is up to the software to decide whether to abort the current transmission.

30.3.6.2. Acknowledgment failure (AF)

An acknowledgment error occurs when the interface detects a NACK bit. At this time:

- The AF bit is set, and if the ITERREN bit is set, an interrupt is generated
- When the transmitter receives a NACK, the communication must be reset:
 - If in slave mode, the hardware releases the bus.

— If in master mode, the software must generate a stop condition or restart.

30.3.6.3. Arbitration loss (ARLO)

An arbitration loss error occurs when the I2C interface detects arbitration loss, at which time:

- The ARLO bit is set by hardware, and if the ITERREN bit is set, an interrupt is generated
- The I2C interface automatically returns to slave mode (MSL bit is cleared). When the I2C interface loses arbitration, it cannot respond to its slave address in the same transmission, but it can respond after receiving a restart condition sent by the master
- Hardware releases the bus

30.3.6.4. Overrun/underrun (OVR)

In slave mode, if clock stretching is disabled, when the I2C interface is receiving data and has already received a byte (RxNE=1), but the previous byte data in the DR register has not been read out, an overrun error occurs.

At this time:

- The last received data is discarded
- In case of an overrun error, software should clear the RxNE bit, and the transmitter should resend the last transmitted byte.

In slave mode, if clock stretching is disabled and the I2C interface is transmitting data, an underrun error occurs if new data is not written to the DR register (TxNE=1) before the clock for the next byte arrives. At this time:

- The previous byte in the DR register will be transmitted repeatedly.
- The user should ensure that the receiver discards repeatedly received data in case of an underrun error. The transmitter should update the DR register within the time specified by the I2C bus standard.

When sending the first byte, the DR register must be written after clearing ADDR and before the first SCL rising edge; if this is not possible, the receiver should discard the first data.

30.3.7. SDA/SCL Control

- If clock stretching is enabled:
 - Transmitter mode: If TxNE=1 and BTF=1: The I2C interface holds the clock line low before transmission, waiting for software to read SR1 and then write data to the data register (both DR and shift register are empty).
 - Receiver mode: If RxNE=1 and BTF=1: The I2C interface holds the clock line low after receiving a data byte, waiting for software to read SR1 and then the data register DR (both DR and shift register are full).
- If clock stretching is disabled in slave mode:
 - If RxNE=1 and DR is not read before the next byte is received, an overrun occurs. The last received byte is lost.
 - If TxNE=1 and no new data is written to DR before the next byte must be sent, an underrun occurs. The same byte will be transmitted repeatedly.
 - Hardware does not implement control for repeated write conflicts.

30.3.8. DMA Request

DMA requests (when enabled) are used only for data transfer. A DMA request is generated when the data register becomes empty during transmission or full during reception. DMA must be initialized and enabled before the end of the current byte transfer. The DMAEN bit (in I2C_CR2 register) must be enabled before the ADDR event occurs.

In master or slave mode, when clock stretching is enabled, the DMAEN bit can be set during the ADDR event before clearing ADDR. The DMA request must be responded to before the current byte transfer is completed. When the DMA transfer data length reaches the value set by DMA, DMA sends EOT (End Of Transfer) to I2C and generates a transfer complete interrupt (if the interrupt enable bit is active):

- Master transmission: In the EOT interrupt service routine, disable the DMA request, then wait for the BTF event before setting the stop condition.
- Master transmission: When the number of data to be received is greater than or equal to 2, DMA sends a hardware signal EOT_1, which corresponds to DMA transfer (number of bytes - 1). If the LAST bit is set in the I2C_CR2 register, the hardware will automatically send a NACK after the next byte following EOT_1. If interrupts are enabled, the user can generate a stop condition in the DMA transfer completion interrupt service routine.

30.3.8.1. DMA Transmission

The DMA mode can be enabled by setting the DMAEN bit in the I2C_CR2 register. As long as the TxE bit is set, data will be loaded from the preset memory area into the I2C_DR register by DMA. To allocate a DMA channel for I2C, follow these steps (x is the channel number):

1. Set the I2C_DR register address in the DMA_CPARx register. Data will be transferred from memory to this address after each TxE event.
2. Set the memory address in the DMA_CMARx register. Data is transferred from this memory area to I2C_DR after each TxE event.
3. Set the desired number of transfer bytes in the DMA_CNDTRx register. This value will be decremented after each TxE event.
4. Configure the channel priority using the PL[0:1] bits in the DMA_CCRx register.
5. Set the DIR bit in the DMA_CCRx register, and configure the interrupt request to be generated either at half or full transfer completion based on application requirements.
6. Activate the channel by setting the EN bit in the DMA_CCRx register.

When the number of data transfers set in the DMA controller is completed, the DMA controller sends an EOT/EOT_1 signal to the I2C interface to indicate the end of transmission. If interrupts are enabled, a DMA interrupt will be generated.

Note: When using DMA for transmission, do not set the ITBUFEN bit in the I2C_CR2 register.

30.3.8.2. DMA Reception

The DMA receive mode can be activated by setting the DMAEN bit in the I2C_CR2 register. Each time a data byte is received, the DMA will transfer the data from the I2C_DR register to the configured memory area (refer to the DMA documentation). To configure the DMA channel for I2C reception, follow these steps (x is the channel number):

1. Set the address of the I2C_DR register in the DMA_CPARx register. Data will be transferred from this address to the memory area after each RxNE event.

2. Set the memory area address in the DMA_CMARx register. Data will be transferred from the I2C_DR register to this memory area after each RxNE event.
3. Set the desired number of transfer bytes in the DMA_CNDTRx register. This value will be decremented after each RxNE event.
4. Configure the channel priority using the PL[0:1] bits in the DMA_CCRx register.
5. Clear the DIR bit in the DMA_CCRx register. Depending on application requirements, interrupts can be requested when half or all of the data transfer is completed.
6. Set the EN bit in the DMA_CCRx register to activate the channel.

When the number of data transfers set in the DMA controller is completed, the DMA controller sends an EOT/EOT_1 signal to the I2C interface to indicate the end of transmission. If interrupts are enabled, a DMA interrupt will be generated.

Note: When using DMA for reception, do not set the ITBUFEN bit in the I2C_CR2 register.

30.3.9. SMBus

The System Management Bus (SMBus) is a two-wire interface. Through it, devices can communicate with each other and with other parts of the system. It is based on the I2C operating principle. SMBus provides a control bus for system and power management-related tasks. A system can use SMBus to exchange information with multiple devices without requiring separate control lines.

The System Management Bus (SMBus) standard involves three types of devices. A slave device receives or responds to commands. A master device is used to issue commands, generate clocks, and terminate transmissions. The host is a dedicated master device that provides the primary interface to the system CPU. The host must have master-slave device functionality and must support the SMBus alert protocol. Only one master is allowed in a system.

The I2C1 module in this project supports SMBus/PMbus functionality

30.3.9.1. Similarities between SMBus and I2C

- 1) Two-wire bus protocol (1 clock, 1 data) + optional SMBus alert line
- 2) Master-slave communication, with the master providing the clock
- 3) Multi-master capability
- 4) SMBus data format is similar to I2C's 7-bit address format

30.3.9.2. Differences between SMBus and I2C

The following table lists the differences between SMBus and I2C:

Table 30-10 Comparison of SMBus and I2C

SMBus	I2C
Maximum transmission speed 100 kHz	Maximum transmission speed 400 kHz
Minimum transmission speed 10 kHz	No minimum transmission speed
35 ms clock low timeout	No clock timeout
Fixed logic level	Logic level determined by VDD
Different address types (reserved, dynamic, etc.)	7-bit, 10-bit, and broadcast call slave address types
Different bus protocols (quick command, process call, etc.)	No bus protocol

30.3.9.3. SMBus Application Purpose

Using the System Management Bus, devices can provide manufacturer information, tell the system their model/part number, save the status of suspend events, report different types of errors, receive control parameters, and return their status. SMBus provides a control bus for system and power management-related tasks.

30.3.9.4. Address Resolution Protocol (ARP)

SMBus slave address conflicts can be resolved by dynamically assigning a new unique address to each slave device.

ARP has the following attributes:

- Address assignment utilizes the standard SMBus physical layer arbitration mechanism.
- The assigned address remains unchanged while the device maintains power, allowing the device to retain its address during power-off.
- After address assignment, there is no additional SMBus packaging overhead (i.e., accessing a device with an assigned address takes the same time as accessing a fixed-address device).
- Any SMBus master device can traverse the bus.

30.3.9.5. SMBus Alert Mode (ALERT)

SMBus alert is an optional signal with an interrupt line, used by devices that wish to extend their control capabilities at the cost of a pin. SMBALERT, like the SCL and SDA signals, is a wired-AND signal. SMBALERT is typically used with the SMBus broadcast call address.

A single slave device can signal the host via SMBALERT to indicate it wishes to communicate, which can be achieved by setting the ALERT bit in the I2C_CR1 register. The host handles the interrupt and accesses all SMBALERT devices through the Alert Response Address ARA (address value 0001100x). Only devices that pull SMBALERT low can respond to the ARA. This status is identified by the SMBALERT status flag in the I2C_SR1 register. The host performs a modified receive byte operation. The 7-bit device address provided by the slave sending device is placed in the 7 highest bits of the byte, and the eighth bit can be 0 or 1.

If multiple devices pull SMBALERT low, the highest priority device (smallest address) will win communication rights through standard arbitration during address transmission. After acknowledging the address, this device must not pull its SMBALERT low again. If the host still sees SMBALERT low after message transmission is complete, it knows it needs to read the ARA again. A host that does not implement the SMBALERT signal can periodically access ARA. For more details on SMBus alert mode, refer to the SMBus specification version 2.0.

30.3.9.6. Bus protocol

The SMBus technical specification supports 9 bus protocols. For detailed information on these protocols and SMBus address types, refer to the SMBus specification version 2.0. These protocols are implemented by the user's software.

30.3.9.7. Timeout error (TIMEOUT)

There are many differences in timing specifications between I2C and SMBus.

SMBus defines a clock low timeout, with a 35ms timeout period. SMBus specifies TLOW: SEXT as the slave device's cumulative clock low extension time. SMBus specifies TLOW: MEXT as the master device's cumulative clock low extension time. For more timeout details, refer to the SMBus specification version 2.0.

The Timeout status flag in I2C_SR1 indicates the state of this feature.

30.3.9.8. PMBus

PMBus is derived from SMBus, with identical transmission logic. The difference lies in PMBus defining some power management-related functions (implemented by software).

30.4. I2C Interrupt

Table 30-30 I2C Interrupt Requests

Interrupt event	Event flag	Enable control bit
Start bit sent (Master)	SB	ITEVTEN
Address sent (Master) or Address matched (Slave)	ADDR	
10-bit Header Sent	ADDR10	
Stop Received (Slave)	STOPF	
Data Byte Transfer Complete	BTF	
Receive Buffer Not Empty	RxNE	ITEVTEN and ITBUFEN
Transmit Buffer Empty	TxE	
Bus Error	BERR	ITERREN
Arbitration Lost (Master)	ARLO	
Response Failure	AF	
Overload/Underload	OVR	
PEC Error	PECERR	
Timeout/Tlow Error	TIMEOUT	
SMBus Alert	SMBALERT	

30.5. I2C registers

The register can be accessed in half-word or word.

30.5.1. I2C Control Register 1 (I2C_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWRST	Res	ALERT	PEC	POS	ACK	STOP	START	NO STRETCH	ENG C	ENPEC	ENARP	SMBTYPE	Res	SMBUS	PE
RW	-	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	-	RW	

Bit	Name	R/W	Reset Value	Function
15	SWRST	RW	0	Software reset.

Bit	Name	R/W	Reset Value	Function
				<p>When set, the I2C is in reset state. Before releasing the reset, ensure the I2C pins are released and the bus is idle.</p> <p>0: The I2C module is not in reset state</p> <p>1: The I2C module is in reset state</p> <p>Note: This bit can be used to reinitialize I2C during error or lock states. When the BUSY bit is 1 and no stop condition is detected on the bus.</p>
14	Reserved	-	-	Reserved
13	ALERT	RW	0	<p>SMBus alert.</p> <p>0: Release the SMBAlert pin to let it go high. Alert response address header followed by NACK</p> <p>1: Drive the SMBAlert pin low. Alert response address header followed by ACK</p> <p>Cleared by hardware when PE=0.</p>
12	PEC	RW	0	<p>Packet error checking</p> <p>The software can set and clear this bit. Hardware can clear this bit under the following conditions: after transmitting PEC, upon start condition, stop condition, or when PE=0.</p> <p>0: No PEC transmission</p> <p>1: PEC transmission (in transmit or receive mode)</p> <p>Note: When arbitration is lost, the PEC calculation becomes invalid.</p>
11	POS	RW	0	<p>ACK/PEC position (for data reception). The software can set/clear this register, or it will be cleared by hardware when PE=0.</p> <p>0: The ACK bit controls the (N)ACK of the byte currently being received in the shift register. The PEC bit indicates the current byte in the shift register is PEC</p> <p>1: The ACK bit controls (N)ACK for the next byte received in the shift register. The PEC bit indicates the next byte received in the shift register is PEC</p> <p>Note: The POS bit can only be used in 2-byte receive configuration and must be configured before receiving data.</p> <p>To NACK the second byte, the ACK bit must be cleared after clearing ADDR.</p>

Bit	Name	R/W	Reset Value	Function
				To detect PEC of the second byte, the PEC bit must be set during ADDR stretching event after configuring the POS bit.
10	ACK	RW	0	<p>ACK enable. Software can set/clear this register, or it is cleared by hardware when PE=0.</p> <p>0: No ACK returned</p> <p>1: Returns an ACK after receiving a byte. (matched address or data)</p>
9	STOP	RW	0	<p>Stop condition generated. Software can set/clear this register, or it is cleared by hardware when a stop condition is detected; set by hardware when a timeout error is detected.</p> <p>In master mode:</p> <p>0: No stop condition generated</p> <p>1: Stop condition generated during current byte transfer or after current start condition is issued</p> <p>In slave mode:</p> <p>0: No stop condition generated</p> <p>1: Release SCL and SDA lines after current byte transfer</p>
8	START	RW	0	<p>START condition generation.</p> <p>This register can be set/cleared by software, or cleared by hardware after START condition is issued or when PE=0.</p> <p>Master mode:</p> <p>0: No START condition generated</p> <p>1: Repeated START condition generated</p> <p>Slave mode:</p> <p>0: No START condition generated</p> <p>1: Generate START condition when bus is free (and automatically switch to master mode by hardware)</p>
7	NOSTRETCH	RW	0	<p>Disable clock stretching (slave mode).</p> <p>When ADDR or BTF flag is set, this bit disables clock stretching in slave mode until cleared by software.</p> <p>0: Clock stretching enabled</p> <p>1: Clock stretching disabled</p>
6	ENGCG	RW	0	<p>General call enable.</p> <p>0: Disable general call. Respond with NACK to address 00h</p>

Bit	Name	R/W	Reset Value	Function
				1: Allow general call. Respond with ACK to address 00h
5	ENPEC	RW	0	PEC enable. 0: Disable PEC calculation 1: Enable PEC calculation
4	ENARP	RW	0	ARP enable. 0: Disable ARP; 1: Enable ARP; If SMBTYPE=0, use the default address of the SMBus device; If SMBTYPE=1, use the primary address of SMBus.
3	SMBTYPE	RW	0	SMBus type. 0: SMBus device 1: SMBus host
2	Reserved	-	-	Reserved
1	SMBUS	RW	0	SMBus mode. 0: I2C mode 1: SMBus mode
0	PE	RW	0	I2C module enabled. 0: Disabled 1: I2C enabled Note: If this bit is cleared during ongoing communication, the I2C module will be disabled and return to idle state after the current communication ends. Since PE=0 after communication ends, all bits are cleared. In master mode, this bit must never be cleared before communication ends.

30.5.2. I2C Control Register 2 (I2C_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	LAST	DMAEN	IT- BUFEN	ITEV- TEN	ITER- REN	Res	Res	FREQ[5:0]					
-	-	-	RW	RW	RW	RW	RW	-	-	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15: 13	Reserved	-	-	Reserved

Bit	Name	R/W	Reset Value	Function
12	LAST	RW	0	DMA last transfer. 0: The next DMA EOT is not the final transfer 1: The next DMA EOT is the final transfer Note: This bit is used in master receive mode to generate a NACK when receiving the last data.
11	DMAEN	RW	0	DMA request enable. 0: DMA request is disabled; 1: DMA request is enabled when TxE=1 or RxNE=1.
10	ITBUFEN	RW	0	Buffer interrupt enable. 0: No interrupt generated when TxE=1 or RxNE=1 1: Event interrupt generated when TxE=1 or RxNE=1 (regardless of DMAEN value)
9	ITEVTEN	RW	0	Event interrupt enable. 0: Disabled 1: Event interrupt enabled The interrupt will be generated under the following conditions: 1) SB=1 (master mode) 2) ADDR=1 (master/slave mode) 3) STOPF=1 (slave mode) 4) BTF=1, but no TxE or RxNE event occurs 5) If ITBUFFEN=1, TxE event is 1 6) If ITBUFEN=1, RxNE event is 1
8	ITERREN	RW	0	Error interrupt enable. 0: Error interrupt disabled; 1: Error interrupt enabled; The interrupt will be generated under the following conditions: — BERR=1 — ARLO=1 — AF=1 — OVR=1 — PECERR=1 — TIMEOUT=1 — SMBAlert=1
7: 6	Reserved	-	-	Reserved
5: 0	FREQ	RW	0	I2C module clock frequency.

Bit	Name	R/W	Reset Value	Function
				<p>This register must be configured with the APB clock frequency value to generate data setup and hold times compatible with the I2C protocol.</p> <p>The minimum configurable frequency is 4MHz (standard mode, i.e. 100k) or 8MHz (400k), and the maximum frequency is the highest APB clock frequency of the chip.</p> <p>000000: Forbidden</p> <p>000001: Forbidden</p> <p>000100: 4 MHz</p> <p>.....</p> <p>100100: 36 MHz</p> <p>.....</p> <p>110010: 50 MHz</p> <p>Greater than 100100: Disabled.</p>

30.5.3. I2C own address register 1 (I2C_OAR1)

Address offset: 0x08

Reset value: 0x0000 4000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDMODE	Res					ADD[9:8]		ADD[7:1]							ADD0
RW	-					RW									

Bit	Name	R/W	Reset Value	Function
15	ADDMODE	RW	0	<p>Addressing mode (slave mode).</p> <p>0: 7-bit slave address (does not respond to 10-bit addresses)</p> <p>1: 10-bit slave address (does not respond to 7-bit addresses)</p>
14:10	Reserved	-	-	Reserved
9:8	ADD[9:8]	RW	0	<p>Interface address.</p> <p>This register is irrelevant in 7-bit addressing mode.</p> <p>Bits 9 to 8 of the address in 10-bit addressing mode.</p>
7:1	ADD[7:1]	RW	0	Bits 7~1 of the interface address.
0	ADDR0	RW	0	<p>Interface address.</p> <p>This register is invalid in 7-bit addressing mode.</p> <p>Bit 0 of the address in 10-bit addressing mode.</p>

30.5.4. I2C Own Address Register 2 (I2C_OAR2)

Address offset: 0x0C

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	OA2MSK[2: 0]			ADD2[7: 1]							ENDUAL
-	-	-	-	-	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15:8	Reserved	-	-	Reserved
7:1	ADD2[7:1]	RW	0	Bits 7~1 of the interface address. Bits 7~1 of the address in dual-address mode.
0	ENDUAL	RW	0	Dual addressing mode enable bit. 0: In 7-bit address mode, only OAR1 is recognized 1: In 7-bit address mode, both OAR1 and OAR2 are recognized

30.5.5. I2C Data Register (I2C_DR)

Address offset: 0x10

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	DR[7:0]							
-	-	-	-	-	-	-	-	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15: 8	Reserved	-	-	Reserved
7: 0	DR[7:0]	RW	0	8-bit data register. Internally, the chip actually has two separate buffers sharing one address: one for storing received data (RX_DR) and one for placing data to be sent to the bus (TX_DR). Transmitter mode: Writing a byte to the DR register (actually TX_DR) automatically initiates data transmission. Once transmission starts (TxE=1), the I2C module will maintain a continuous data stream if the next data to be transmitted is written to the DR register in time.

Bit	Name	R/W	Reset Value	Function
				Receiver mode: The received byte is copied to the DR register (actually RX_DR) (RxNE=1). Continuous data reception can be achieved by reading the data register before the next byte is received (RxNE=1). Note: <ol style="list-style-type: none"> 1. In slave mode, the address will not be written to the data register DR. 2. The hardware does not handle write conflicts (data can still be written to the data register even if TxE=0). 3. If an ARLO event occurs while processing the ACK pulse, the received byte will not be copied to the data register and thus cannot be read.

30.5.6. I2C Status Register (I2C_SR1)

Address offset: 0x14

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMBA LERT	TIMEO UT	Res	PECE RR	OV R	AF	AR LO	BER R	Tx E	RxN E	Re s	STOP F	ADD1 0	BT F	ADD R	S B
RC_W0		-	RC_W0					R	R	-	R	R	R	R	R

Bit	Name	R/W	Reset Value	Function
15	SMBALERT	RC_W0	0	SMBus alert status. SMBus master mode: 0: No SMBus alert 1: Generates an SMBAlert event on the pin SMBus slave mode: 0: No SMBAlert response address header sequence 1: Received SMBAlert response address header sequence until SMBAlert goes low This bit is cleared by software writing 0 or by hardware when PE=0.
14	TIMEOUT	RC_W0	0	Timeout or Tlow error. 0: No timeout error; 1: The duration of SCL being low has reached 25ms (timeout); or the master's cumulative clock stretch time in

Bit	Name	R/W	Reset Value	Function
				<p>low level exceeds 10ms (Tlow:mext); or the slave's cumulative clock stretch time in low level exceeds 25ms (Tlow:sext).</p> <p>When this bit is set in slave mode: the slave resets communication and hardware releases the bus;</p> <p>When this bit is set in master mode: hardware generates a stop condition;</p> <p>This bit is cleared by writing 0 in software or by hardware when PE=0.</p> <p>Note: This function is only effective in SMBUS mode.</p>
13	Reserved	-	-	Reserved
12	PECERR	RC_W0	0	<p>PEC error occurred during reception.</p> <p>0: No PEC error, ACK is returned after receiving PEC (if ACK=1)</p> <p>1: PEC error occurred, NACK is returned after receiving PEC (regardless of ACK value)</p> <p>This bit is cleared by writing 0 in software or by hardware when PE=0.</p>
11	OVR	RC_W0	0	<p>Overflow/underrun flag.</p> <p>0: No overflow/underrun;</p> <p>1: Overflow/underrun has occurred.</p> <p>When NOSTRETCH=1, this bit is set by hardware in slave mode;</p> <p>In receive mode, when a new byte is received (including the ACK pulse) and the content of the data register has not been read, the newly received byte will be lost.</p> <p>In transmit mode, when a new byte is to be sent but no new data is written to the data register, the same byte will be transmitted twice.</p> <p>This bit is cleared by writing 0 in software or by hardware when PE=0.</p> <p>Note: If the write operation to the data register occurs very close to the rising edge of SCL, the transmitted data will be indeterminate, and a hold time violation will occur.</p>
10	AF	RC_W0	0	<p>Acknowledge failure flag.</p> <p>0: No acknowledge failure;</p> <p>1: Acknowledge failure.</p> <p>The hardware sets this register when no acknowledgment is received.</p>

Bit	Name	R/W	Reset Value	Function
				This bit is cleared by writing 0 in software or by hardware when PE=0.
9	ARLO	RC_W0	0	<p>Arbitration loss (master mode).</p> <p>0: No arbitration loss detected; 1: Arbitration loss detected.</p> <p>The hardware sets this register when the interface loses control of the bus to another master.</p> <p>This bit is cleared by writing 0 in software or by hardware when PE=0.</p> <p>After an ARLO event, the I2C interface automatically switches back to slave mode (MSL=0).</p>
8	BERR	RC_W0	0	<p>Bus error flag.</p> <p>0: No start or stop condition error; 1: Start or stop condition error.</p> <p>When the interface detects an erroneous start or stop condition, hardware sets this bit to 1.</p> <p>This bit is cleared by writing 0 via software, or by hardware when PE=0.</p>
7	TxE	R	0	<p>Data register empty (during transmission) flag.</p> <p>0: Data register is not empty; 1: Data register is empty.</p> <p>During data transmission, this bit is set to 1 when the data register is empty, and it is not set during the address transmission phase.</p> <p>Writing data to the DR register via software clears this bit, or it is automatically cleared by hardware after a start or stop condition occurs, or when PE=0.</p> <p>If a NACK is received, or when the next byte to be transmitted is PEC (PEC=1), this bit is not set.</p> <p>Note: Writing the first data to be transmitted, or writing data when BTF is set, cannot clear the TxE bit because the data register is empty at this time.</p>
6	RxNE	R	0	<p>Data register not empty (during reception) flag.</p> <p>0: Data register is empty; 1: Data register is not empty.</p> <p>During reception, this register is set when the data register is not empty. This register is not set during the address reception phase.</p>

Bit	Name	R/W	Reset Value	Function
				Software read/write operations on the data register will clear this register, or it will be cleared by hardware when PE=0. Note: When BTF is set, reading data cannot clear the RxNE bit because the data register remains full at this time.
5	Reserved	-	-	Reserved
4	STOPF	R	0	Stop condition detection bit (slave mode). 0: No stop bit detected; 1: Stop condition detected. After an acknowledgment (if ACK=1), when the slave detects a stop condition on the bus, hardware sets this bit to 1. After software reads the I2C_SR1 register, writing to the I2C_CR1 register will clear this bit, or hardware will clear this bit when PE=0.
3	ADD10	R	0	10-bit header sequence has been sent (master mode). 0: No ADD10 event has occurred. 1: The master has sent the first address byte. In 10-bit address mode, when the master has sent the first byte, hardware sets this bit to 1. After software reads the I2C_SR1 register, writing to the I2C_DR register will clear this bit, or hardware will clear this bit when PE=0. Master: After receiving NACK, this register is not set.
2	BTF	R	0	Byte transfer end flag. 0: Byte transfer not completed 1: Byte transfer successfully completed Hardware sets this register under the following conditions (in slave mode when NOSTRETCH=0; in master mode, regardless of NOSTRETCH): <ul style="list-style-type: none"> ✓ During reception, when a new byte is received (including the ACK pulse) and the data register has not yet been read (RxNE=1). ✓ During transmission, when a new data should be sent and the data register has not yet been written with new data (TxNE=1). This bit is cleared by software after reading the I2C_SR1 register and performing a read or write operation

Bit	Name	R/W	Reset Value	Function
				<p>on the data register; or by hardware when a start or stop condition is sent, or when PE=0.</p> <p>Note:</p> <p>The BTF bit will not be set after receiving a NACK.</p>
1	ADDR	R	0	<p>Address has been sent (Master mode)/Address matched (Slave mode).</p> <p>This bit is cleared by software after reading the I2C_SR2 register and then reading the I2C_SR1 register; or by hardware when a start or stop condition is sent during transmission, or when PE=0.</p> <p>Address matched (Slave):</p> <p>0: Address does not match or no address received;</p> <p>1: Received address matches.</p> <p>Hardware sets this bit when the received slave address matches the OAR register or broadcast call address, or the SMBus device default address, or when the SMBus host recognizes an SMBus alert.</p> <p>Note: In slave mode, it is recommended to perform a complete clear sequence, i.e., after ADDR is set, first read the SR1 register, then read the SR2 register.</p> <p>Address sent (Master):</p> <p>0: Address transmission not ended;</p> <p>1: Address transmission ended.</p> <p>For 10-bit addresses, it is set when an ACK for the second byte of the address is received.</p> <p>For 7-bit addresses, it is set when an ACK is received.</p> <p>Note: This register will not be set after receiving a NACK.</p>
0	SB	R	0	<p>Start bit flag (Master mode).</p> <p>0: Start condition has not been transmitted;</p> <p>1: Start condition has been transmitted;</p> <p>—This register is set when a start condition is transmitted.</p> <p>—This bit is cleared by a read or write operation to the data register after the software reads the I2C_SR1 register; or cleared by hardware when PE=0.</p>

30.5.7. I2C Status Register 2 (I2C_SR2)

Address offset: 0x18

Reset value: 0x0000 0000

Note: Even if the ADDR flag is set after reading the I2C_SR1 register, reading the I2C_SR2 register after reading I2C_SR1 will clear the ADDR flag. Therefore, the I2C_SR2 register must only be read when the ADDR bit of the I2C_SR1 register is found to be set or the STOPF bit is cleared.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PEC[7:0]								DUALF	SMBHOST	SMBDE- FAULT	GEN- CALL	Res	TRA	BUSY	MSL
R	R	R	R	R	R	R	R	R	R	R	R	-	R	R	R

Bit	Name	R/W	Reset Value	Function
15: 8	PEC	R	0	Packet error checking register. When ENPEC=1, this register stores the internal PEC value.
7	DUALF	R	0	Dual-address flag (slave mode). 0: The received address matches OAR1; 1: The received address matches OAR2. Cleared by hardware when a stop condition or a repeated start condition is generated, or when PE=0.
6	SMBHOST	R	0	Received SMBus host header sequence (slave mode) flag. 0: No SMBus host address received; 1: When SMBTYPE=1 and ENARP=1, the SMBus host address is received. Cleared by hardware when a stop condition or a repeated start condition is generated, or when PE=0.
5	SMBDEFAULT	R	0	SMBus slave default address (slave mode). 0: No default address of the SMBus device received; 1: When ENARP=1, the default address of the SMBus device is received. Cleared by hardware when a stop condition or a repeated start condition is generated, or when PE=0.
4	GENCALL	R	0	Broadcast call address (slave mode). 0: No broadcast call address received; 1: When ENGC=1, the address received for a broadcast call.

Bit	Name	R/W	Reset Value	Function
				Cleared by hardware when a stop condition or a repeated start condition is generated, or when PE=0.
3	Reserved	-	-	Reserved
2	TRA	R	0	<p>Transmit/Receive flag.</p> <p>0: Data has been received.</p> <p>1: Data has been transmitted.</p> <p>This register is set according to the R/W bit of the address byte at the end of the entire address transmission phase.</p> <p>Cleared by hardware when a stop condition (STOPF=1), repeated start condition, bus arbitration loss (ARLO=1), or when PE=0 is detected.</p>
1	BUSY	R	0	<p>Bus busy flag.</p> <p>0: No data communication on the bus.</p> <p>1: Data communication is ongoing on the bus.</p> <p>Set by hardware when SDA or SCL is detected as low level.</p> <p>Cleared by hardware when a stop condition is detected.</p> <p>This register indicates the currently ongoing bus communication. The information continues to be updated even when the interface is disabled (PE=0).</p>
0	MSL	R	0	<p>Master/slave mode.</p> <p>0: Slave mode</p> <p>1: Master mode</p> <p>—Set by hardware when the interface is in master mode (SB=1);</p> <p>—Cleared by hardware when a stop condition is detected on the bus (STOPF=1), arbitration is lost (ARLO=1), or when PE=0.</p>

30.5.8. I2C Clock Control Register (I2C_CCR)

Address offset: 0x1C

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F/S	DUTY	Res	Res	CCR[11:0]											

RW	RW	-	-	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
----	----	---	---	----	----	----	----	----	----	----	----	----	----	----	----	----

Bit	Name	R/W	Reset Value	Function
15	F/S	RW	0	I2C master mode selection. 0: Standard mode 1: Fast mode
14	DUTY	RW	0	Duty cycle in fast mode. 0: In fast mode: Tlow/Thigh=2 1: In fast mode: Tlow/Thigh=16/9
13: 12	Reserved	-	-	Reserved
11: 0	CCR[11:0]	RW	0	<p>Clock control prescaler in fast/standard mode (master mode). This prescaler is used to set the SCL clock in master mode.</p> <p>1. Standard mode:</p> <ul style="list-style-type: none"> Thigh=CCR x Tpclk Tlow =CCR x Tpclk <p>2. Fast mode:</p> <ul style="list-style-type: none"> DUTY=0: Thigh=CCR x Tpclk Tlow =2 x CCR x Tpclk DUTY=1 (to achieve 400KHz): Thigh=9 x CCR x Tpclk Tlow =16 x CCR x Tpclk <p>Note:</p> <ul style="list-style-type: none"> The minimum allowed setting is 0x04, and the minimum allowed in fast DUTY mode is 0x01 Thigh=tr(SCL)+tw(SCLH) Tlow=tr(SCL)+tw(SCLL) These delays have no filter This register can only be configured when PE=0;

30.5.9. I2C TRISE Register (I2C_TRISE)

Address offset: 0x20

Reset value: 0x0000 0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	TRISE[5:0]					
-	-	-	-	-	-	-	-	-	-	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15: 6	Reserved	-	-	Reserved
5: 0	TRISE	RW	0x0002	<p>Maximum rise time in fast/standard mode (master mode).</p> <p>These bits should provide the maximum duration of the SCL feedback loop in master mode. The purpose is to ensure that SCL maintains a stable frequency regardless of the SCL rise time duration.</p> <p>These bits must be set to the maximum SCL rise time specified in the I2C bus standard, with an increment step of 1.</p> <p>For example: The maximum allowed SCL rise time in standard mode is 1000ns. If the value in <code>FREQ[5:0]</code> of the <code>I2C_CR2</code> register equals 0x08 and <code>Tpclk=125ns</code>, then <code>TRISE</code> is configured as 0x09 ($1000\text{ns}/125\text{ns} = 8 + 1 = 9$).</p> <p>The filter value can also be added to <code>TRISE</code>.</p> <p>If the result is not an integer, the integer part is written to <code>TRISE</code> to ensure the <code>tHIGH</code> parameter.</p> <p>Note: This register can only be set when <code>PE=0</code>.</p>

31. Serial Peripheral Interface (SPI)

This project implements two SPI modules, both with identical functionality.

31.1. Introduction

The SPI interface can be configured to support either the SPI protocol or the I2S audio protocol. The SPI interface defaults to SPI mode and can be switched from SPI mode to I2S mode via software.

The Serial Peripheral Interface (SPI) allows the chip to communicate with external devices in half/full-duplex, synchronous, serial mode. This interface can be configured as a master mode and provides the communication clock (SCK) for external slave devices. The interface can also operate in a multi-master configuration. It can be used for various purposes, including two-wire simplex synchronous transmission using a bidirectional data line, as well as reliable communication with CRC checks.

I2S is also a 3-wire synchronous serial interface communication protocol. It supports four audio standards, including the Philips I2S standard, MSB and LSB justified standards, and the PCM standard. In half-duplex communication, it can operate in both master and slave modes. When acting as a master, it provides clock signals to external slave devices through the interface.

31.2. SPI main features

31.2.1. SPI main features

- Supports SPI master mode and SPI slave mode
- 3-wire full-duplex synchronous transmission
- 2-wire half-duplex synchronous transmission (with bidirectional data line)
- 2-wire simplex synchronous transmission (no bidirectional data line)
- 8-bit or 16-bit transmission frame selection
- Supports multi-master mode
- Eight master mode baud rate prescalers (maximum $f_{PCLK}/2$)
- Slave mode frequency (maximum $f_{PCLK}/4$)
- NSS management by software or hardware in both master and slave modes: dynamic switching between master/slave operation modes
- Programmable clock polarity and phase
- Programmable data order, MSB first or LSB first
- Dedicated transmit and receive flags capable of triggering interrupts
- SPI bus busy status flag
- Supports reliable communication with hardware CRC
 - In transmit mode, the CRC value can be sent as the last byte
 - Automatic CRC check on the last received byte in full-duplex mode
- Interrupt flags: Master mode fault, overrun, CRC error

- Two DMA-capable Rx and Tx FIFOs with 4-level depth and 16-bit width (8-bit width when data frame is set to 8 bits)

31.2.2. I2S main features

- Half-duplex communication (transmit or receive only)
- Master or slave operation
- 8-bit linear programmable prescaler for precise audio sampling frequency (8 kHz ~ 96 kHz)
- Data format can be 16-bit, 24-bit or 32-bit
- Fixed audio channel packet format as 16-bit (16-bit data frame) or 32-bit (16-bit, 24-bit, 32-bit data frame)
- Programmable clock polarity (steady state)
- Underrun flag in slave transmit mode and overrun flag in master/slave receive mode
- 16-bit data register for transmission and reception, with one register on each side of the channel
- Supports I2S protocol:
 - I2S Philips standard
 - MSB justified standard (left-aligned)
 - LSB justified standard (right-aligned)
 - PCM standard (long or short frame sync on 16-bit channel frame or 16-bit data frame extended to 32-bit channel frame)
- Data direction is always MSB first
- Both transmit and receive have DMA capability (16-bit wide)
- Master clock can be output to external audio devices with fixed ratio of $256 \times f_s$ (f_s is audio sampling frequency)

31.3. SPI functional description

31.3.1. Overview

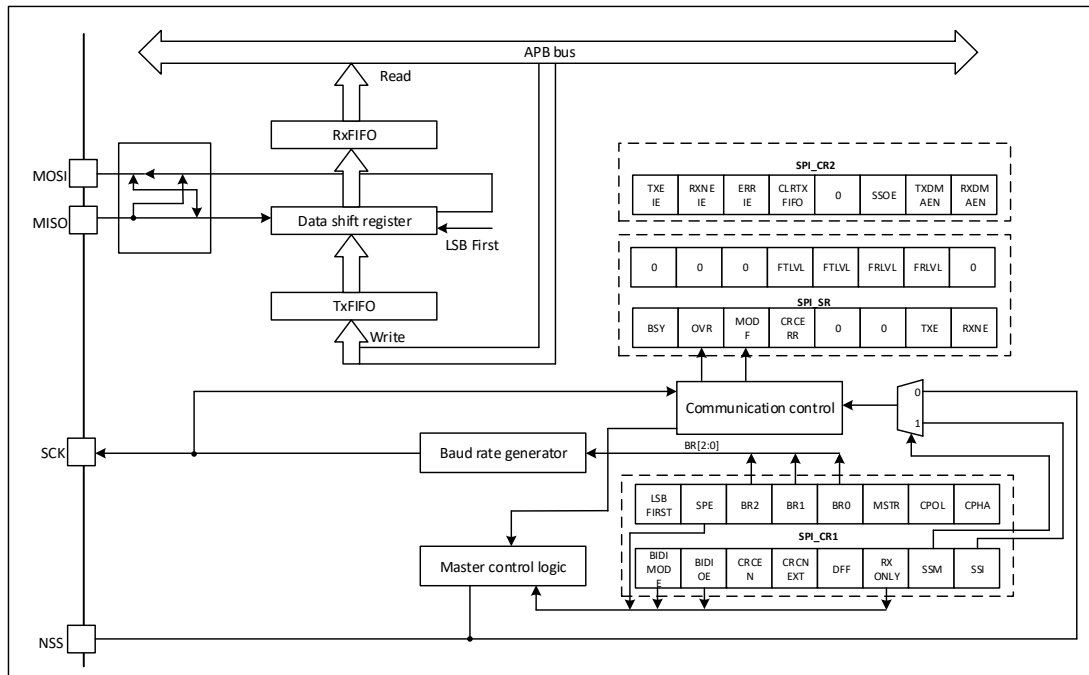


Figure 31-1 SPI block diagram

SPI connects to external devices through 4 pins:

MISO: Master input/slave output pin. This pin transmits data in slave mode and receives data in master mode.

MOSI: Master output/slave input pin. This pin transmits data in master mode and receives data in slave mode.

SCK: Serial clock, output in master mode and input in slave mode.

NSS: Slave select. Depending on SPI and NSS configuration, this pin can function as:

- Select the slave to communicate with
- Synchronize data frames
- Detect collisions between multiple masters

The SPI bus allows communication between one master and one or more slaves. The bus consists of at least two lines: one for the clock and another for synchronously transmitted data. Depending on the application scenario, an additional data line and slave NSS signal can be optionally added.

31.3.2. Single master and single slave communication

For different application scenarios, SPI can communicate using several different configurations.

These configurations use 2-wire, 3-wire (software NSS), or 4-wire (hardware NSS). Communication is always initiated by the master.

31.3.2.1. Full-duplex communication

By default, SPI is configured for full-duplex communication. In this configuration, the master and slave shift registers are connected using two unidirectional lines between MOSI and MISO. During SPI communication, data is synchronously shifted at the clock edges provided by the master. The master sends data via MOSI and receives data from the slave via MISO. When the data frame transmission is complete (all bits shifted out), the information exchange between the master and slave is completed.

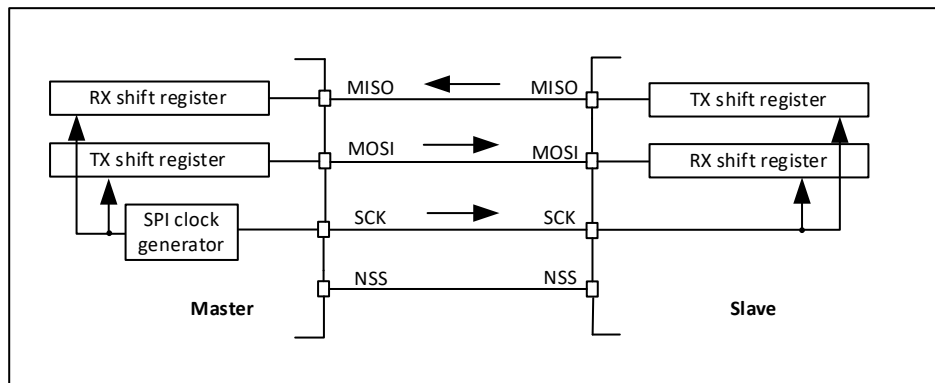


Figure 31-2 Full-duplex single master/single slave application

31.3.2.2. Half-duplex communication

By setting the BIDIMODE bit (SPI_CR1 register), the SPI can operate in half-duplex mode. In this configuration, the master and slave shift registers are connected using a single data line. During communication, data is synchronously shifted between the two shift registers at the clock edges of SCK, in the direction selected by BIDIOE (SPI_CR1 register). In this configuration, the master's MISO and the slave's MOSI are freed up as general-purpose ports for other applications.

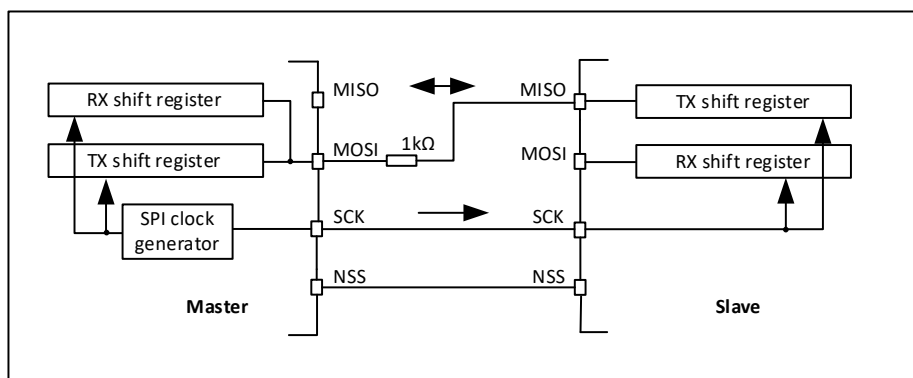


Figure 31-3 Half-duplex single master/single slave application

NSS can be used for hardware flow control between the master and slave devices. Optionally, NSS may not be used, in which case the process must be handled internally.

In this configuration, the MISO pin of the master and the MOSI pin of the slave can be used as GPIO.

A critical situation occurs when the communication direction between two nodes operating in bidirectional mode is not synchronized. The new transmitter accesses the shared data line while the previous transmitter still maintains the opposite value on the line (the value depends on SPI configuration and communication data). Two nodes may conflict, briefly providing opposite output levels on the shared line until the next node also changes its direction setting accordingly. It is recommended to insert a serial resistor between the MISO and MOSI pins in this mode to protect the output and limit current flow between them.

31.3.2.3. Simplex Communication

By using RXONLY (SPI_CR1 register), set the SPI to transmit-only or receive-only mode, enabling simplex operation. In this configuration, only one wire is used between the shift registers of the master and slave. The other pair of MISO and MOSI is unused and can be used as standard GPIO.

- Transmit-only mode (RXONLY=0): The configuration is the same as full-duplex. The application ignores information on unused ports. This port can be used as a standard GPIO.
- Receive-only mode (RXONLY=1): By setting RXONLY, the application can disable the SPI output function. In slave configuration, the MISO output is disabled, and this port is used as GPIO. When its slave NSS signal is active, the slave continues to receive data from MOSI. Whether a received data event occurs depends on the configuration of the data buffer. In master configuration, the MOSI output is disabled, and this port can be used as GPIO. As long as SPI is in use, the clock signal is continuously generated. The only way to stop the clock is to clear RXONLY or SPE until the input from MISO is completed. And then fill the data buffer based on the corresponding configuration.

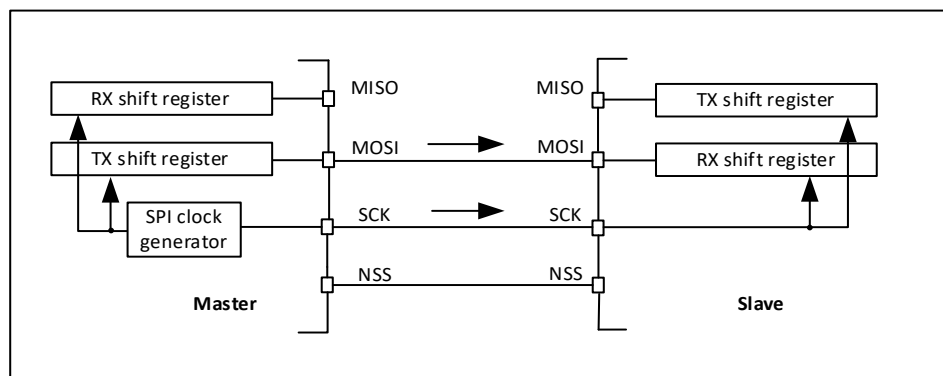


Figure 31-4 Simplex Single Slave/Single Master Application
(Master is in transmit-only mode/Slave is in receive-only mode)

- (1) Hardware flow control can be implemented between the master and slave using NSS. Optionally, NSS may also not be used. The process then proceeds to internal handling.
- (2) Capture unexpected input information on the input of the Rx shift register. In standard transmit-only mode, all events related to reception must be ignored.
- (3) In this configuration, the MISO pins on both sides are used as GPIO.

By setting the transmission direction (when the BIDIOE bit remains unchanged, bidirectional mode is enabled), any simplex communication can be replaced by half-duplex communication.

31.3.3. Multi-Slave Communication

In configurations with two or more independent slaves, the master uses GPIO to manage NSS for each slave. The master must select a slave by pulling the connected NSS low. Once this is done, standard master and dedicated slave communication is established.

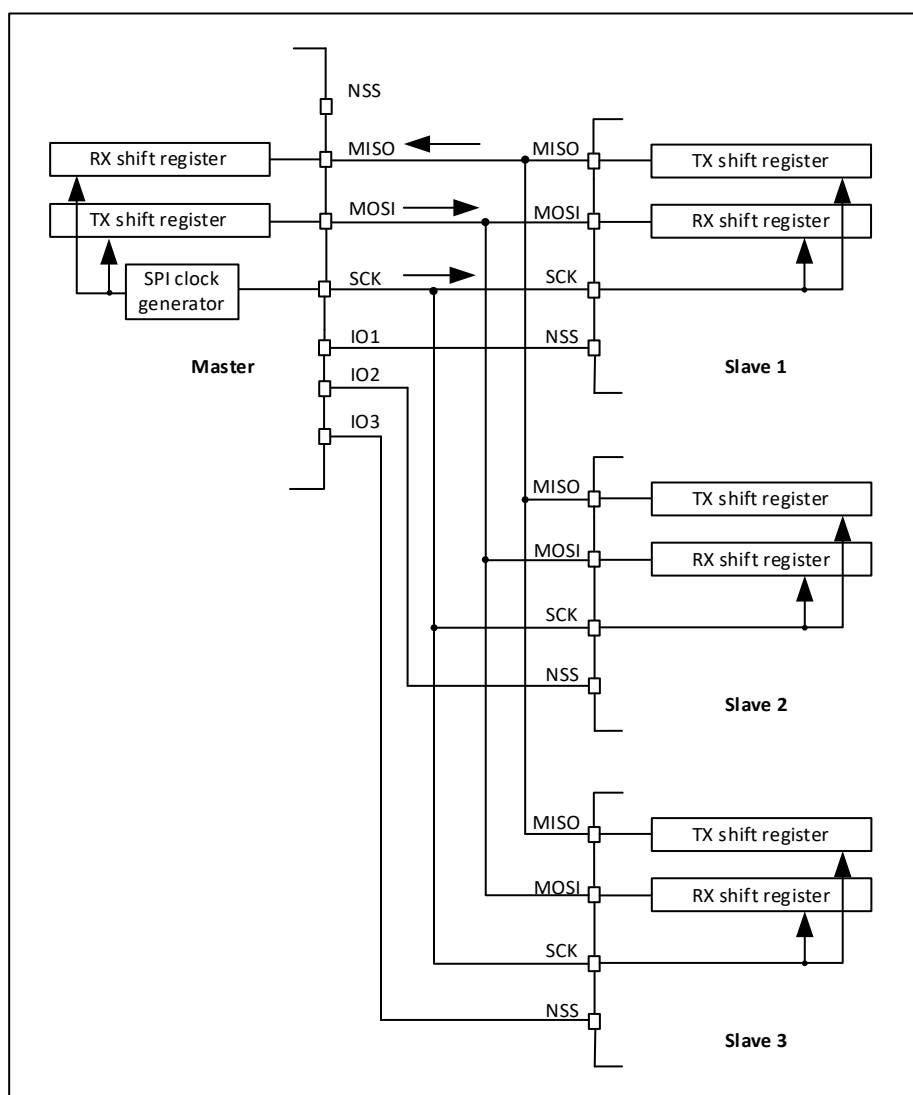


Figure 31-5 Master Communicating with Three Independent Slaves

In this configuration, the master does not use the NSS pin. Any MODF errors must be prevented by setting **SSM=1** and **SSI=1**.

Since the MISO of slaves are connected together, all slaves must configure their MISO GPIO as open-drain mode.

31.3.4. Multi-Master Communication

Unless the SPI bus is not designed with multi-master capability, users can utilize its built-in feature to detect potential conflicts when two nodes attempt to control the bus simultaneously. When such detection is needed, the NSS pin configured as a hardware input mode must be used.

In this mode, it is impossible to have more than two SPI nodes working in a connection, as only one node can transmit on the shared data line at a time.

When the node is inactive, both default to slave mode. Once a node intends to control the bus, it switches itself to master mode and then applies a valid level to the NSS pins of other slave nodes through a dedicated GPIO pin. After this process is completed, the valid slave select signal is released, and the node controlling the bus temporarily returns to passive slave mode, waiting for a new process to begin.

If two nodes issue control requests at the same time, a bus collision event will occur (refer to the MODF event). Then the user can apply some simple arbitration procedures (e.g., applying different predefined timeouts on two nodes to delay the next attempt).

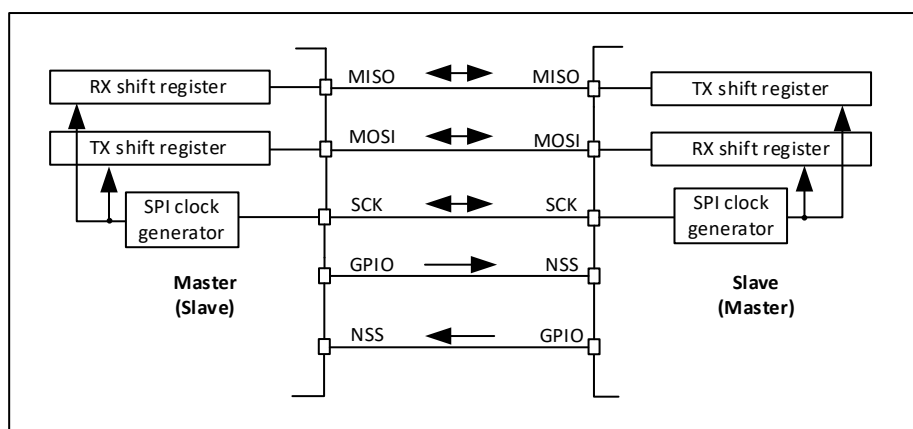


Figure 31-6 Multi-Master Application

NSS is configured in hardware input mode on both nodes. When a passive node is configured as a slave, its active level enables MISO output control.

31.3.5. Slave select (NSS) pin management

In slave mode, NSS acts as a standard chip select input, enabling the slave to communicate with the master. In master mode, NSS can function as either an output or an input. When used as an input, it prevents bus conflicts in multi-master systems; when used as an output, it drives the slave select signal for a single slave.

Through the SSM bit in the SPI_CR1 register, hardware or software slave NSS management can be selected:

- Software NSS management (SSM=1): In this configuration, the slave select signal is driven by the internal SSI bit (SPI_CR1 register). The external NSS pin is released for other applications.
- Hardware NSS management (SSM=0): In this case, several configurations are possible.
 - 1) NSS output enabled (SSM=0, SSOE=1): This configuration is used only when acting as a master. Hardware management of the NSS pin. When SPI is enabled in master mode (SPE=1), the NSS signal is pulled low and remains low until SPI is disabled (SPE=0). In multi-master applications, SPI cannot use this NSS configuration.

- 2) NSS output disabled (SSM=0, SSOE=0): If the MCU acts as a master on the bus, this configuration enables multi-master functionality. If the NSS pin is pulled low at this time, the SPI enters a master fault state, and the chip is automatically reconfigured as a slave. In slave mode, the NSS pin acts as a standard chip select input; when NSS is low, the slave is selected.

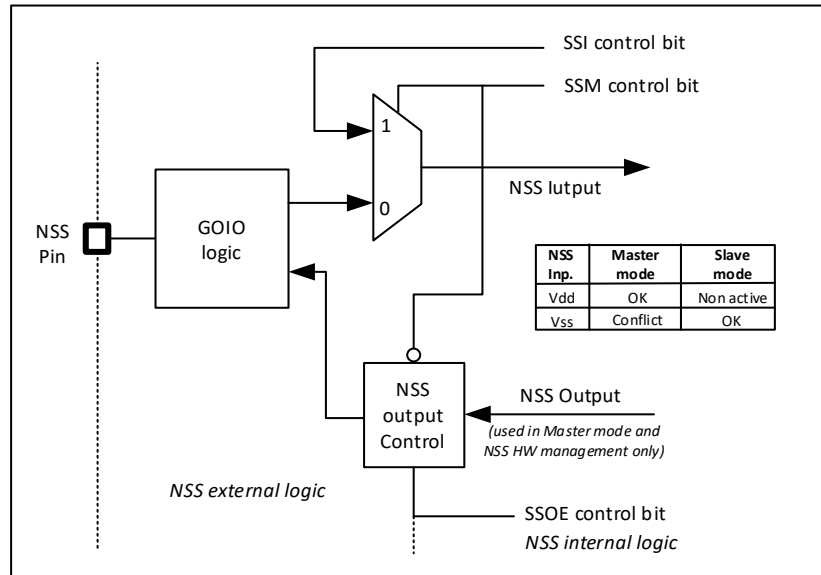


Figure 31-7 Hardware/Software Slave NSS Management

31.3.6. Communication format

During SPI communication, receive and transmit operations occur simultaneously. SCK (serial clock) synchronizes the shifting and sampling operations of data on the line. The communication format depends on clock phase, clock polarity, and data frame format. To enable communication, the master and slave must follow the same communication format.

31.3.6.1. Clock phase and polarity control

Through the CPOL and CPHA bits (SPI_CR1 register), software can configure four possible timing modes. CPOL (clock polarity) controls the idle state of the clock when no data is being transmitted. This bit affects both master and slave. If CPOL is reset, the SCK pin remains low in the idle state. If CPOL is set, the SCK pin remains high in the idle state.

If CPHA is set, the second edge of SCK captures the first transmitted data bit (falling edge if CPOL is reset, otherwise rising edge). Data is latched at each occurrence of this clock edge. If CPHA is reset, the first edge of SCK captures the first transmitted data bit (falling edge if CPOL is set, otherwise rising edge). Data is latched at each occurrence of this clock edge.

The combination of CPOL and CPHA is used to select the data capture clock edge.

SPI must be disabled (SPE=0) before switching CPOL/CPHA.

The idle state of SCK must correspond to the polarity selected in the SPI_CR1 register (pull SCK high if CPOL=1; pull SCK low if CPOL=0).

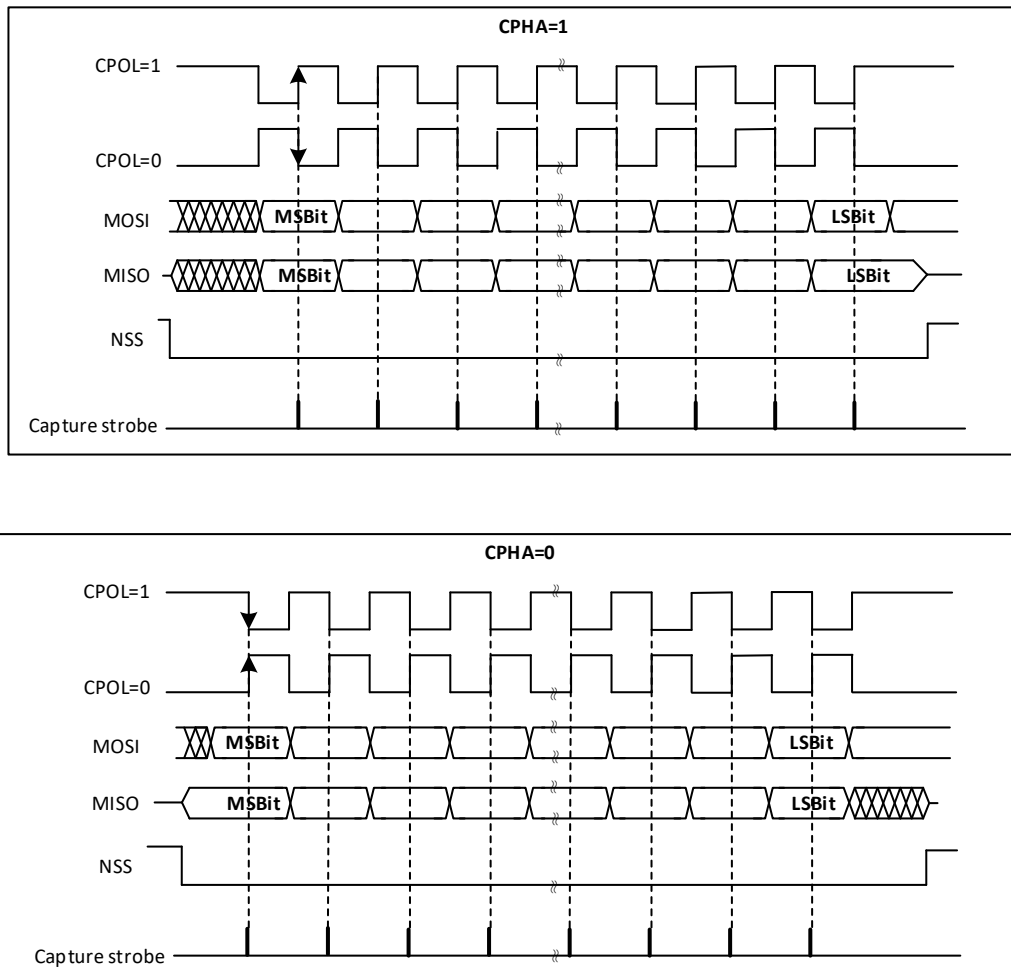


Figure 31-8 Data Clock Timing Diagram

The order of data bits depends on the LSBFIRST bit setting.

31.3.6.2. Data frame format

Through the LSBFIRST bit (SPI_CR1 register), the SPI shift register can be set to MSB-FIRST or LSB-FIRST. Select the number of bits in the data frame using the DFF bit (SPI_CR1 register). Optional 8-bit or 16-bit length, applicable to both transmission and reception.

31.3.7. SPI Configuration

The SPI configuration process is nearly identical for both master and slave. For specific mode setup, refer to the dedicated section. When performing standard communication, follow these steps:

- 1) Write to the relevant GPIO registers: configure the MOSI, MISO, and SCK pins.
- 2) Write to the SPI_CR1 register.
 - Set the clock baud rate through BR[2:0] (not required in slave mode).
 - Configure CPOL and CPHA to define the relationship between data transmission and the serial clock (one of four possible relationships).
 - Select simplex or half-duplex mode via RXONLY or BIDIMODE and BIDIOE (RXONLY and BIDIMODE cannot be active simultaneously)
 - Configure LSBFIRST to define the frame format

- Configure SSM and SSI
 - Configure the MSTR bit (in multi-master NSS configurations, avoid NSS conflict states if the host is configured to prevent MODF errors)
 - Configure DFF to select the data frame bit length
- 3) Write the SPI_CR2 register
 - Configure SSOE (not required in slave mode)
 - 4) Write the SPI_CRCPR register: configure the CRC polynomial if needed.
 - 5) Write the corresponding DMA registers: enable TXDMAEN or RXDMAEN, and configure SYSCFG_CFGR3.DMAX_MAP and SYSCFG_CFGR4.DMAX_MAP registers to select the SPI's DMA channel.

31.3.8. SPI enable process

It is recommended to enable the SPI slave before the host sends the clock. If not handled this way, data transmission may be abnormal. The slave's data register must contain the data to be sent before starting communication with the host (either at the first clock edge or, if the clock signal is continuous, before the ongoing communication ends). Before enabling the SPI slave, the SCK signal must stabilize to the idle state level corresponding to the selected polarity.

In full-duplex mode (or transmit-only mode), the host starts communication when SPI is enabled and TXFIFO is not empty, or when the next write operation to TXFIFO is performed.

In any host receive-only mode (RXONLY=1, or BIDIMODE=1 and BIDIOE=0), the host starts communication by enabling SPI and immediately outputs the clock.

For DMA handling, refer to the specific section.

31.3.9. Data transmission and reception process

RXFIFO and TXFIFO

All SPI data communication goes through FIFOs with a depth of 4 and a width of 16-bit/8-bit. This feature enables SPI to operate with a continuous data stream and prevents communication overflow caused by the CPU's inability to process data in time. There are separate FIFOs for transmission and reception, called TXFIFO and RXFIFO. These FIFOs are used in all SPI modes.

FIFO handling depends on various parameters, including: data exchange mode (full-duplex, half-duplex), and data frame format.

Reading the SPI_DR register retrieves the earliest unread data result stored in the RXFIFO. Writing to the SPI_DR register stores the written data at the last position of the FIFO transmit queue. Read and write accesses must be aligned with the data width. The FTLVL[1:0] and FRLVL[1:0] bits indicate the current occupancy levels of the two FIFOs.

Read access to the SPI_DR register must be managed by the RXNE event. This event is triggered when data is stored in the RXFIFO. When RXNE is cleared, the RXFIFO is considered empty.

Similarly, write access for data frames to be transmitted is managed by the TXE event. This event is triggered when the occupancy level of the TXFIFO is less than or equal to half of its total capacity. Otherwise, TXE is cleared, and the TXFIFO is considered full.

In this way, both the RXFIFO and TXFIFO can store up to 4 data frames.

TXE and RXNE events can be handled via polling or interrupts.

Another way to manage data exchange is by using DMA.

An overflow event occurs when the RXFIFO is full and the next data is received. Overflow events can be handled via polling or interrupts.

The BSY bit being set to 1 indicates that a data frame is currently being processed. When the clock signal runs continuously, the BSY flag remains set to 1 between data frames in the host, but in the slave device, the BSY flag goes low for the shortest duration (one SPI clock cycle) between data frame transmissions.

In certain application scenarios, when writing data to the TXFIFO, the TXFIFO can be cleared by setting the CLR_TXFIFO bit, allowing new data to be written for re-establishing communication.

Sequence handling

Multiple data frames can be transmitted in a single sequence to complete a message. After enabling transmission, the sequence begins when there is any data in the host's TXFIFO and continues as long as data exists in the host's TXFIFO. The clock signal is continuously provided by the host until the TXFIFO is empty, after which the clock signal stops and waits for additional data.

In receive-only mode, either half-duplex (BIDIMODE=1, BIDIOE=0) or simplex mode (BIDIMODE=0, RXONLY=1), the host starts receiving immediately once SPI is enabled and the receive-only mode is activated. The host continuously provides the clock and receives data until the host stops SPI or disables the receive-only mode.

When the master can provide all communication in continuous mode (SCK signal is continuous), the master must consider the slave's ability to handle the data stream. When necessary, the master must reduce the communication speed and provide a slower clock or individual frames or data segments with sufficient delay. Note that for either master or slave, there is no underflow error signal; data from the slave is typically exchanged and processed by the master (even if the slave cannot prepare the data in time). For slaves, a better approach is to use DMA, especially when the data frame is small and the baud rate is high.

Each sequence must be controlled by an NSS pulse, while selecting one of the slaves for communication in a multi-slave system. In a single-slave system, there is no need to use NSS to control the slave, but it is still advisable to provide an NSS pulse to synchronize the slave system with the beginning of each data sequence. NSS can be managed by either software or hardware.

When BSY is set, it indicates an ongoing data frame exchange. When the dedicated frame exchange is completed, the RXNE flag is set. The last bit is sampled, and the entire data frame is stored in RXFIFO.

SPI shutdown procedure

SPI must be shut down following a specific shutdown procedure. When the peripheral clock stops, it is crucial to ensure this before the system enters low-power mode. Otherwise, it may damage the ongoing exchange. In certain modes, the disable step is the only way to stop ongoing continuous communication.

In full-duplex or transmit-only mode, the host can terminate any transaction by stopping the provision of data to be sent. In this case, the clock stops after the last data exchange. In these modes, the user must follow the standard shutdown procedure before turning off SPI. During SPI master mode transmission, if SPI is turned off during data frame processing or when there is data to be transmitted in TXFIFO, the state of SPI becomes unpredictable.

When the host is in receive-only mode, the only way to stop the continuous clock is to disable the peripheral (SPE=0). This must be done within a specific time window during the last data frame transmission, between the sampling of the first bit and the start of the last bit transmission (to receive the full number of expected data frames and prevent reading any additional "empty" data after the last valid data frame). If this is not done, the master will continue generating the clock. In this mode, a dedicated SPI shutdown procedure must be followed.

After the SPI is disabled, received but unread data remains in the RXFIFO and must be processed after the next SPI enable before starting a new sequence. To prevent unread data, ensure the RXFIFO is empty when the SPI is disabled (using the correct shutdown procedure or by resetting all SPI registers via software initialization).

The standard shutdown procedure is based on the BSY status and checking FTLVL[1:0] to ensure the transmission is fully completed. This check can also be performed in specific cases where it is necessary to identify whether the ongoing transmission has ended, such as:

1. When the NSS signal is managed by software, the master must provide the slave with the correct NSS pulse. Or
2. The data stream from DMA or FIFO completes while the last data frame or CRC frame transmission is still being processed on the peripheral bus.

The correct shutdown procedure is (except for receive-only mode):

- Wait for FTLVL[1:0]=00 (no data to transmit)
- Wait for BSY=0 (the last data has been processed)
- Disable the SPI (SPE=0)
- Read data until FRLVL[1:0]=00 (read all received data)

For the specific receive-only mode, the correct shutdown procedure is:

- Interrupt the reception process by disabling the SPI (SPE=0) during the transmission of the last data frame
- Wait for BSY=0 (the last data frame has been processed)

- Read data until FRLVL[1:0]=00 (read all received data)

31.3.9.1. Using DMA communication

To operate at maximum speed and accelerate the data read/write process to avoid overflow, the SPI provides DMA functionality, which uses a simple request/acknowledge protocol.

A DMA request is generated when either TXE or RXNE is set. Tx and Rx buffers have independent requests.

1. During transmission, a DMA request is generated each time TXE is set to 1. Then DMA writes data to the SPI_DR register.
2. During reception, a DMA request is generated each time RXNE is set to 1. Then DMA reads the data from the SPI_DR register.

When SPI is used only for transmitting data, only the SPI Tx DMA channel can be enabled. In this case, the OVR flag is set because the received data is not read. When SPI is used only for receiving data, the SPI Rx DMA channel can be enabled.

During transmission, when DMA has written all data to be sent (the TCIF flag in the DMA_ISR register is set), the BSY flag can be checked to ensure SPI communication is complete. This is to avoid corruption of the last transmission when disabling SPI or entering stop low-power mode. The software must first wait for FTLVL[1:0]=00, then wait for BSY=0.

When starting DMA communication, the following steps must be followed to avoid error events on the DMA channel:

1. Enable the DMA Rx buffer (RXDMAEN bit in SPI_CR2) (if Rx DMA is used).
2. Configure the SYSCFG_CFGR3.DMAx_MAP and SYSCFG_CFGR4.DMAx_MAP registers to select the DMA channel used by SPI, and configure the DMA registers (refer to the DMA module description).
3. Enable the DMA Tx buffer (TXDMAEN bit in the SPI_CR2 register) (if Tx DMA is used).
4. Enable SPI via the SPE bit.

To forcibly terminate communication, the following steps must be followed:

- If DMA is enabled, disable the DMA module by operating the DMA registers (refer to the DMA module description).
- Disable SPI via the SPI disable procedure.
- Disable the DMA Tx and Rx buffers (if DMA Tx and Rx are used) by clearing TXDMAEN and RXDMAEN (SPI_CR2 register).

31.3.9.2. Communication Timing

This section describes some typical timings, which are valid for polling, interrupts, or DMA. For simplicity, assume LSBFIRST=0, CPOL=0, CPHA=1. Full DMA operation configuration is not provided.

- After activating NSS and enabling SPI, the slave takes control of MISO; the slave loses control of MISO when NSS is released or SPI is disabled. Sufficient time must be provided for the slave to prepare host-specific data before the transmission begins.

On the host side, the SPI peripheral controls the MOSI and SCK signals (including the NSS signal) only after SPI is enabled. If the SPI is turned off, the SPI peripheral is disconnected from the GPIO, and the levels on these lines depend on the GPIO configuration.

- On the master side, if communication is continuous, BSY remains active between frames. On the slave side, the BSY signal typically remains low for at least one clock cycle between data frames.
- The TXE signal is only cleared when the TXFIFO is full.
- After the TXDMAEN bit is set, the DMA arbitration process begins. After TXEIE is set, a TXE interrupt is generated. When the TXE signal is active, data begins transferring to the TxFIFO until the TxFIFO is full or the DMA transfer is complete.
- If all data to be transmitted can fit into the TxFIFO, the DMA Tx TCIF flag is raised before SPI bus transmission begins. This flag remains high until the SPI transaction is completed.

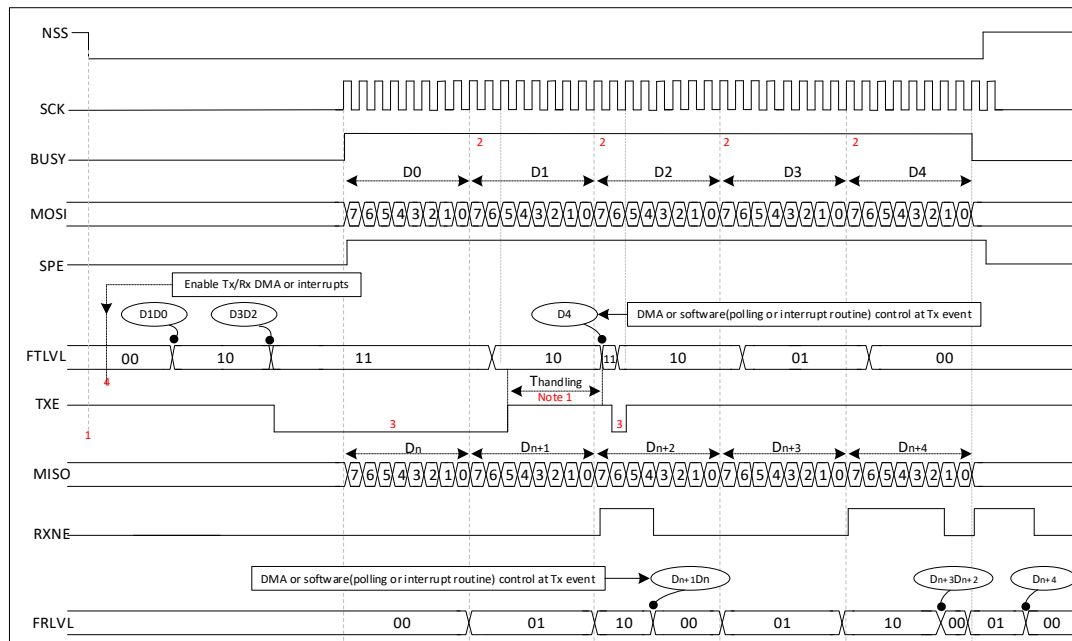


Figure 31-9 Master full-duplex communication timing (data width=8, FRXTH=0)

31.3.10. Status flags

The application can fully monitor the SPI bus status through three status flags.

Transmit buffer empty flag (TXE)

When the TXFIFO has enough space to store the data to be transmitted, the TXE flag is set. The TXE flag is related to the TXFIFO occupancy level. This flag goes high and remains high until the TXFIFO occupancy level is less than or equal to 1/2 FIFO depth, at which point it is cleared by hardware. If TXEIE (SPI_CR2) is set, an interrupt request is generated. When the TXFIFO occupancy level is greater than 1/2, this bit is automatically cleared.

Receive buffer not empty flag (RXNE)

The RXNE flag is set:

- When data is received, RXNE is set.

When the above conditions are no longer met, RXNE is automatically cleared by hardware.

Busy flag (BSY)

The BSY flag is set and cleared by hardware (writing to this bit has no effect).

When set to '1', it indicates the SPI is busy with communication, with one exception: in master mode bidirectional receive mode (MSTR=1, BDM=1, and BDOE=0), the BSY flag remains low during reception.

Before the software disables the SPI module and enters stop low-power mode (or turns off the device clock), the BSY flag can be used to detect whether the transmission has ended, thus avoiding corruption of the last transmission. Therefore, the following procedure must be strictly followed.

The BSY flag can also be used to avoid write conflicts in multi-master systems.

Except for the bidirectional receive mode in master mode (MSTR=1, BDM=1, and BDOE=0), the BSY flag is set to '1' when the transmission starts.

The flag will be cleared to '0' under the following conditions:

- Properly Disabling SPI
- In master mode, when MODF=1 is generated.
- In master mode, when the transmission is complete and no valid data remains to be sent.
- In slave mode, between each data transmission, the BSY flag is set to 0 and remains for at least one SPI clock cycle.

Note: Do not use the BSY flag to handle each data transmission and reception. Using TXE and RXNE is more appropriate.

31.3.11. Error Flags

Master Mode Failure (MODF)

Master mode failure (MODF) occurs only when: NSS is used as an input signal (SSOE=0), the NSS pin is under hardware mode management, and the NSS pin of the master is pulled low; or when the NSS pin is under software mode management and the SSI bit is set to '0'. At this point, the MODF bit is automatically set.

A master mode failure has the following effects on the SPI device:

- The MODF bit is set to '1', and if the ERRIE bit is set, an SPI interrupt is generated;
- The SPE bit is cleared to '0'. This stops all outputs and disables the SPI interface;
- The MSTR bit is cleared to '0', forcing the device into slave mode.

The following steps are used to clear the MODF bit:

1. When the MODF bit is set to '1', perform a read or write operation on the SPI_SR register;
2. Then write to the SPI_CR1 register.

In a system with multiple MCUs, to avoid conflicts with multiple slave devices, the NSS pin of the master device must be pulled high before clearing the MODF bit. After the clearing is completed, the SPE and MSTR bits can be restored to their original state.

For security reasons, when the MODF bit is '1', the hardware does not allow setting the SPE and MSTR bits.

Under normal configuration, the MODF bit of the slave cannot be set to '1'. However, in a multi-master configuration, a device can be in slave mode with the MODF bit set; in this case, the MODF bit

indicates a possible multi-master conflict. The interrupt routine can execute a reset or return to the default state to recover from the error state.

Overflow flag (OVR)

An overflow condition occurs when data is received by the master or slave, and RXFIFO does not have enough space to store the received data. This situation occurs if the software or DMA does not have enough time to read the previously received data (stored in RXFIFO).

When an overflow occurs, newly received data will not overwrite the previously stored data in RXFIFO. The newly received data is ignored, and all subsequent data sent will be discarded.

The OVR bit in the SPI_SR register can be cleared by sequentially reading the SPI_DR register.

CRC error (CRCERR)

When the CRCEN bit in the SPIx_CR1 register is set to 1, this flag is used to verify the validity of the received data. If the value received in the shift register does not match the value of SPIx_RXCRCR, the CRCERR flag in the SPIx_SR register will be set to 1. This flag is cleared by software.

31.3.12. SPI interrupt

Table 31-1 SPI interrupt requests

Interrupt event	Event flag	Enable control bit
TXFIFO waiting to be loaded	TXE	TXEIE
Data received in RXFIFO	RXNE	RXNEIE
Master mode fault event	MODF	ERRIE
Overflow error	OVR	ERRIE
CRC protocol error	CRCERR	ERRIE

31.3.13. CRC calculation

CRC check is used to ensure communication reliability, with separate CRC calculators for data transmission and data reception. CRC is calculated by performing a programmable polynomial operation on each received bit. SPI provides CRC8 or CRC16 calculation independent of the frame data length, which can be fixed at 8-bit or 16-bit.

31.3.13.1. CRC principle

CRC calculation is enabled by setting the CRCEN bit in the SPIx_CR1 register before enabling SPI (SPE = 1). The CRC value is calculated using an odd programmable polynomial for each bit. This calculation is processed on the sampling clock edge defined by the CPHA and CPOL bits in the SPIx_CR1 register. The calculated CRC value is automatically checked at the end of the data block, and its transmission is managed by the CPU or DMA. When a mismatch is detected between the internally calculated CRC based on the received data and the CRC sent by the transmitter, the CRCERR flag is set to indicate a data error. The correct processing steps for CRC calculation depend on the SPI configuration and the selected transmission management method.

Note: The polynomial value should only be odd; any even value is not supported.

31.3.13.2. CPU-managed CRC transmission

Communication will continue until the last data frame in the SPIx_DR register must be sent or received. Then the CRCNEXT bit must be set in the SPIx_CR1 register to indicate that the CRC frame transmission will start after the currently processed data frame. The CRCNEXT bit must be set before the end of the last data frame transmission. Stop CRC calculation during CRC transmission.

The received CRC data is stored in the RXFIFO in byte or word format. This is why, in CRC mode only, the receive buffer must be treated as a single 16-bit buffer, used to receive only one data frame at a time. CRC format transmission typically involves sending one additional data frame at the end of data transmission. However, when configuring 8-bit data frames with 16-bit CRC checks, two additional frames are required to transmit the complete CRC.

When the last CRC data is received, an automatic check is performed, comparing the received value with the SPIx_RXCRC register. Software must check the CRCERR flag in the SPIx_SR register to determine if a data transmission error occurred. Software clears the CRCERR flag by writing '0' to it. After receiving the CRC, the CRC value is stored in the RXFIFO, and the SPIx_DR register must be read to clear the RXNE flag.

31.3.13.3. DMA-Managed CRC Transmission

When SPI is enabled and DMA mode with CRC is activated for communication, CRC is automatically sent and received at the end of communication (except for reading CRC data in receive-only mode), and the CRCNEXT bit does not need to be handled by software. The counter of the SPI transmit DMA channel must be set to the number of data frames to be transmitted, excluding CRC frames.

On the receiving end, the received CRC value is automatically handled by the DMA at the end of transmission, but the user must ensure to clear the received CRC information from the RXFIFO, as it is always stored there. In full-duplex mode, the counter of the receive DMA channel can be set to the number of data frames to be received, including CRC.

In receive-only mode, the DMA receive channel counter should only include the amount of data transmitted, excluding CRC calculations. Then, for a complete DMA-based transmission, since it operates as a single buffer in this mode, all CRC values must be read back from the FIFO by software. At the end of data and CRC transmission, if an error occurs during transmission, the CRCERR flag in the SPIx_SR register is set.

frequency of the output clock signal is preset to $256 \times F_s$, where F_s is the sampling frequency of the audio signal.

When configured in master mode, I2S uses its own clock generator to produce the clock signal for communication. This clock generator is also the clock source for the master clock output. In I2S mode, there are two additional registers: one is the SPI_I2SPR register related to clock generator configuration, and the other is the SPI_I2SCFGR register for general I2S configuration (which can set parameters such as audio standard, slave/master mode, data format, packet frame, clock polarity, etc.).

In I2S mode, the SPI_CR1 register and all CRC registers are not used. Similarly, in I2S mode, the SSOE bit in the SPI_CR2 register and the MODF and CRCERR bits in the SPI_SR register are not used.

I2S uses the same SPI_DR register as SPI for 16-bit wide mode data transmission.

31.4.2. Audio Protocol

The three-wire bus only needs to handle audio data that is typically time-division multiplexed on the left and right channels, but only one 16-bit register is used for transmission or reception. Therefore, the software must write the value corresponding to each channel into the data register or read data from the data register, and identify the corresponding channel by checking the CHSIDE bit in the SPIx_SR register. The left channel always transmits data before the right channel (the CHSIDE bit is meaningless under the PCM protocol). There are four available combinations of data and packet frames. Data can be sent in the following four formats:

- 16-bit data packed into a 16-bit frame
- 16-bit data packed into a 32-bit frame
- 24-bit data packed into a 32-bit frame
- 32-bit data packed into a 32-bit frame

When extending 16-bit data to a 32-bit frame, the first 16 bits (MSB) are meaningful data, and the last 16 bits (LSB) are forced to 0. This operation requires no software intervention and no DMA request (only one read/write operation is needed). 24-bit and 32-bit data frames require the CPU to perform two read or write operations on the SPI_DR register. When using DMA, two DMA transfers are needed. For 24-bit data, after extending to 32 bits, the lowest 8 bits are set to 0 by hardware. For all data formats and communication standards, the most significant bit (MSB) is always transmitted first.

The I2S interface supports four audio standards, which can be selected by configuring the I2SSTD[1:0] bits and PCMSYNC bit in the SPI_I2SCFGR register.

31.4.2.1. I2S Philips Protocol

Under this standard, the WS pin indicates which channel the transmitted data belongs to. There is one clock cycle before the first bit of data (MSB) becomes valid.

The 16/32-bit full precision timing diagram is shown below. The transmitter changes data on the falling edge of the clock signal (CK), and the receiver reads data on the rising edge. The WS signal also changes on the falling edge of CK.

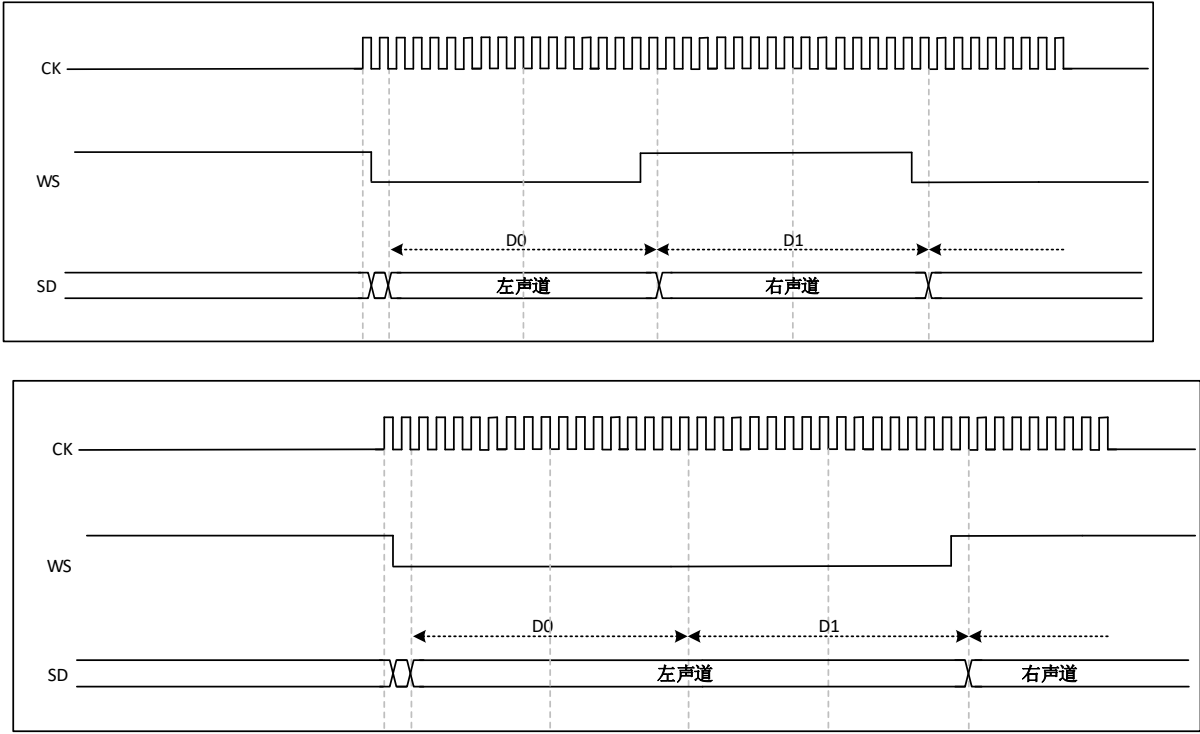


Figure 31-11 I2S Philips protocol waveform (16/32-bit full precision, CKPOL = 0)

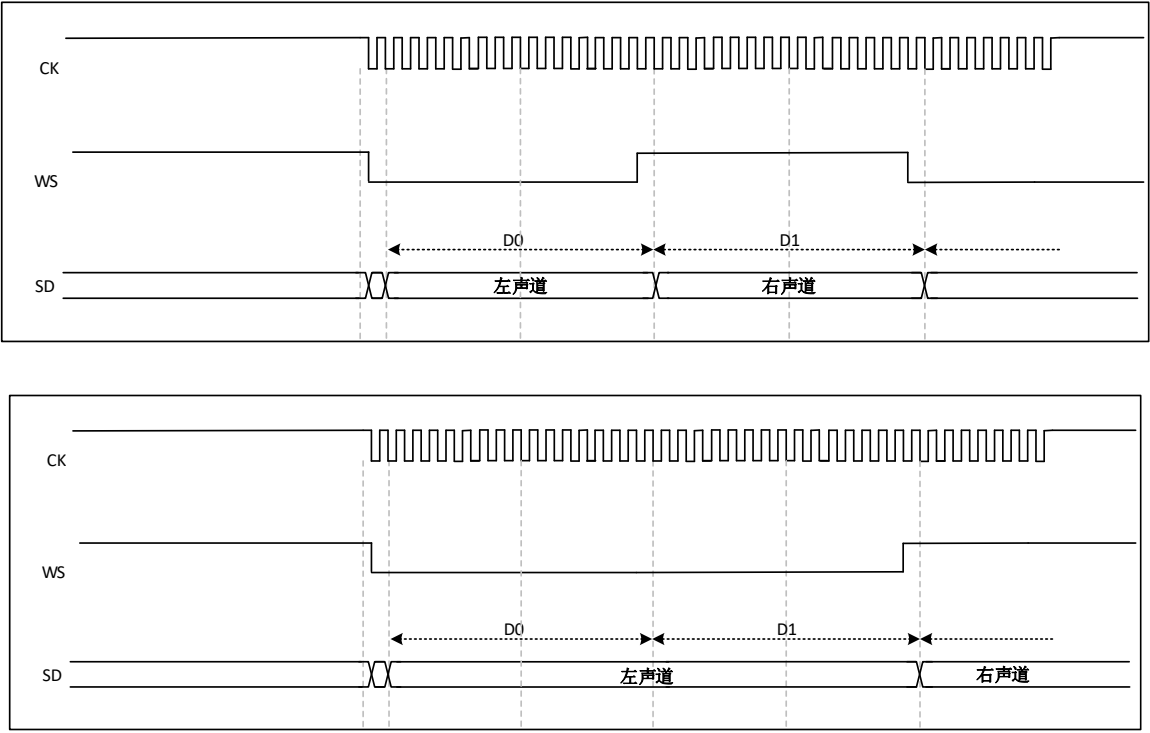


Figure 31-12 I2S Philips protocol waveform (16/32-bit full precision, CKPOL = 0)

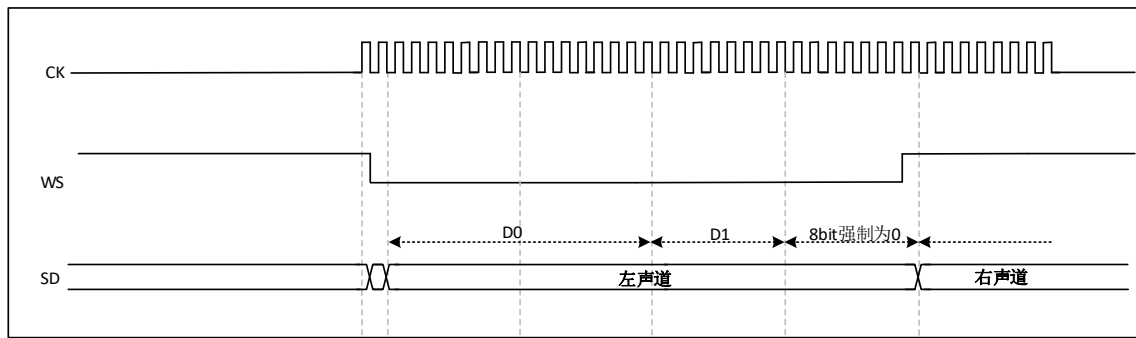


Figure 31-13 I2S Philips protocol standard waveform (24-bit frame, CKPOL = 0)

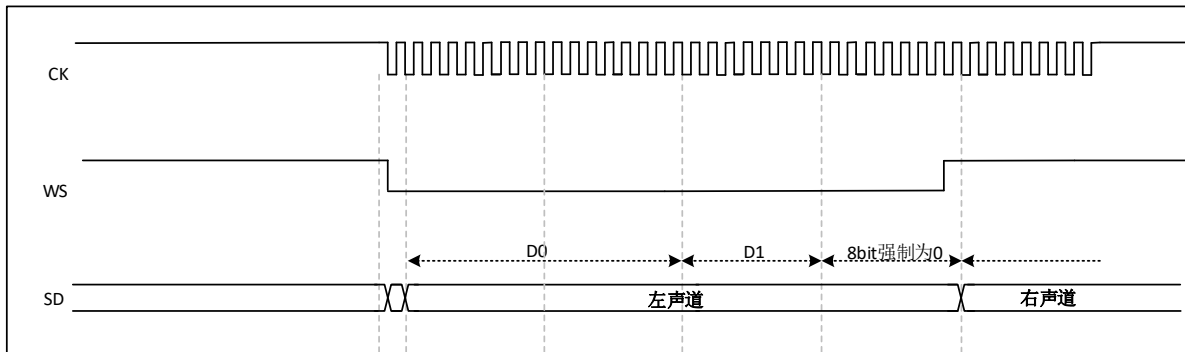


Figure 31-14 I2S Philips protocol standard waveform (24-bit frame, CKPOL = 1)

This mode requires two read or write operations to the SPI_DR register.

- In transmit mode: If transmitting 0x8EAA33 (24-bit), the first write is 0x8EAA, and the second write is 0x33xx. The lower 8 bits are meaningless.
- In receive mode: If receiving 0x8EAA33 (24-bit), the first reception is 0x8EAA, and the second readout is 0x3300. Only the upper 8 bits are meaningful data, while the lower 8 bits are always 0.

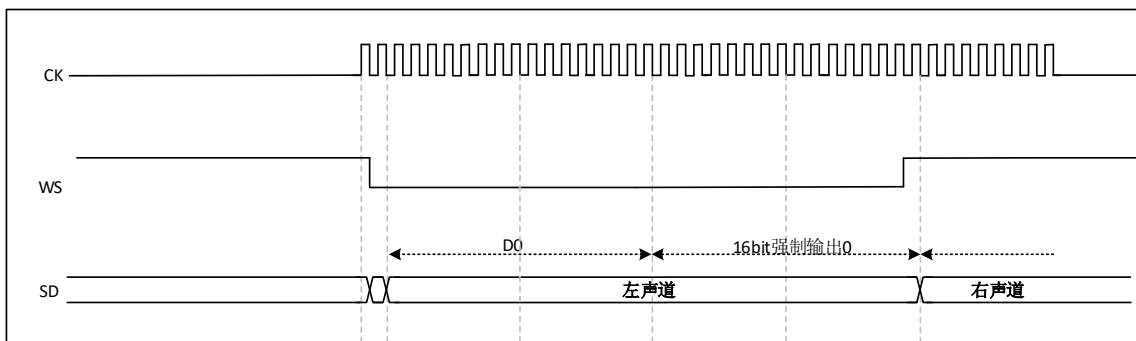


Figure 31-15 I2S Philips protocol standard waveform (16-bit extended to 32-bit packet frame, CPOL = 0)

During I2S configuration, if you choose to extend 16-bit data to a 32-bit channel frame, only one access to the SPI_DR register is required. The lower 16 bits used for extension to 32-bit are set to 0x0000 by hardware.

If the data to be transmitted or received is 0x76A3 (extended to 32-bit as 0x76A30000), only one operation on SPI_DR is needed to write the data 0x76A3.

During transmission, the MSB must be written to the SPI_DR register: the TXE flag is set to 1, indicating that new data can be written, and an interrupt can be generated if the corresponding interrupt is enabled. Transmission is handled by hardware. Even if the lower 16 bits (0x0000) have not been

transmitted yet, TXE is set, and the corresponding interrupt can be generated. During reception, the RXNE flag is set to 1 each time the upper 16-bit half-word (MSB) is received, and an interrupt can be generated if the corresponding interrupt is enabled.

This extends the time interval between two write or read operations, thereby preventing underflow or overflow (depending on the data transmission direction).

31.4.2.2. MSB-aligned standard

In this standard, the WS signal and the first data bit, i.e., the most significant bit (MSB), are generated simultaneously.

The 16/32-bit full precision timing diagram is shown below. The transmitter changes data on the falling edge of the clock signal, and the receiver reads data on the rising edge.

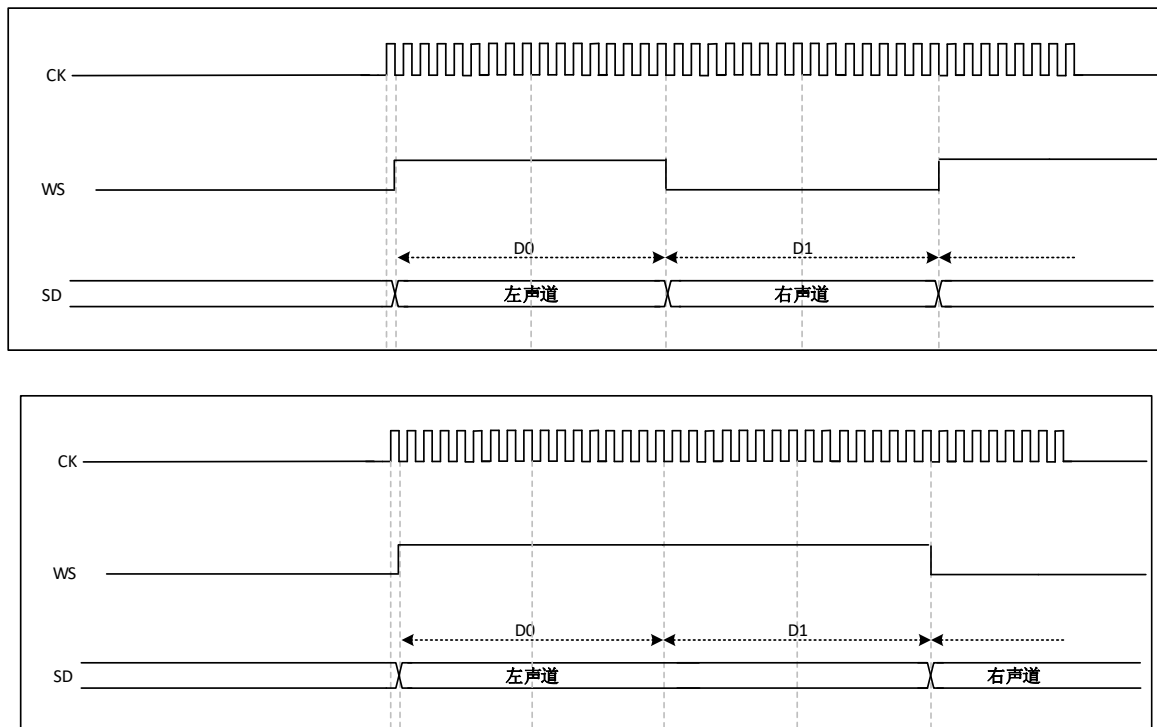
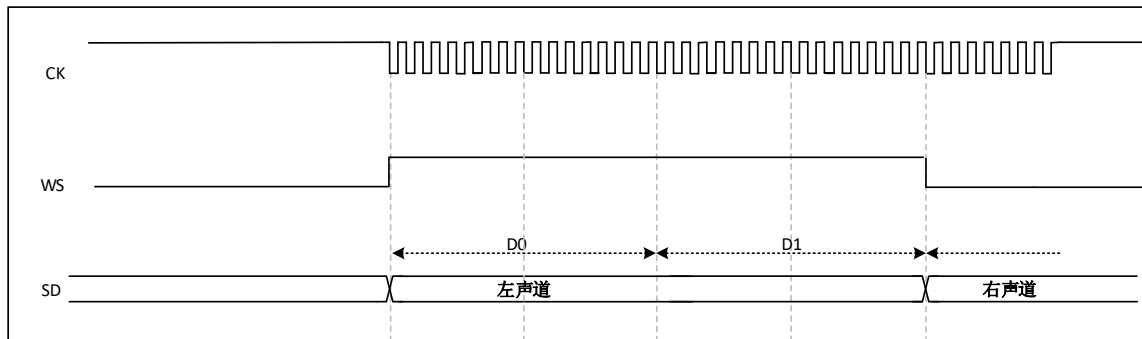


Figure 31-16 MSB-aligned 16-bit or 32-bit full precision, CPKOL = 0



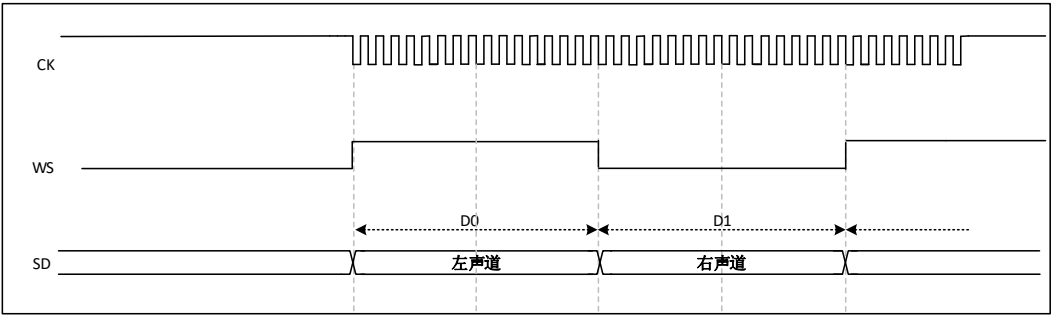


Figure 31-17 MSB-aligned 16-bit or 32-bit full precision, CPKOL = 1

The 24-bit frame timing diagram is as follows:

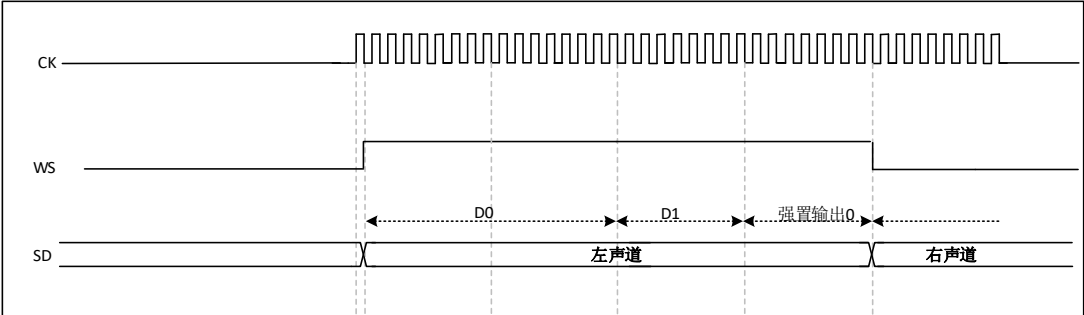


Figure 31-18 MSB-aligned 24-bit frame, CPKOL = 0

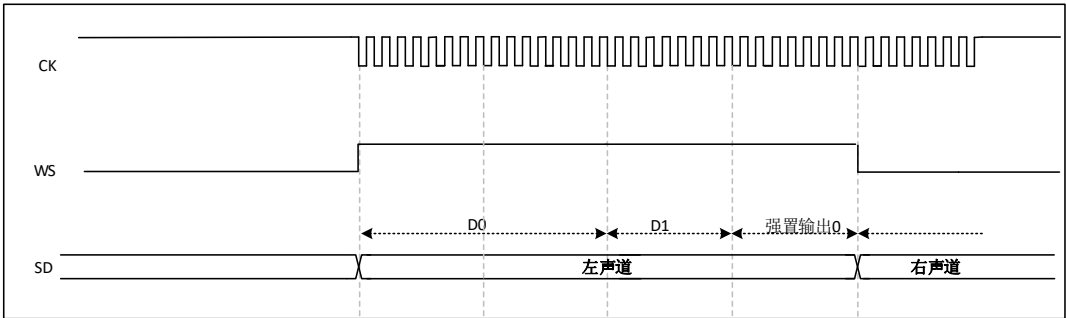


Figure 31-19 MSB-aligned 24-bit frame, CPKOL = 1

The following figure shows the timing for 16-bit data extended to a 32-bit channel frame:

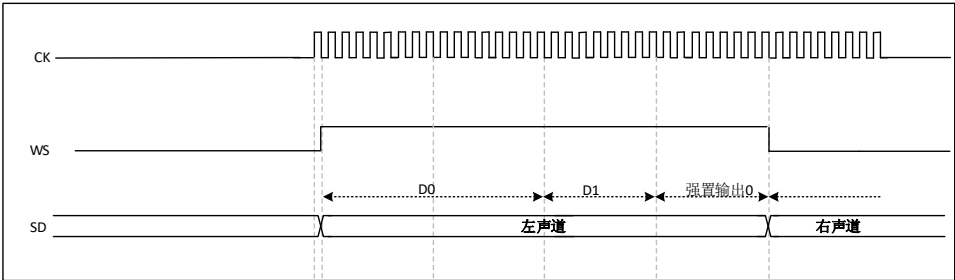


Figure 31-20 MSB-aligned standard, 16-bit extended to 32-bit, CPKOL = 0

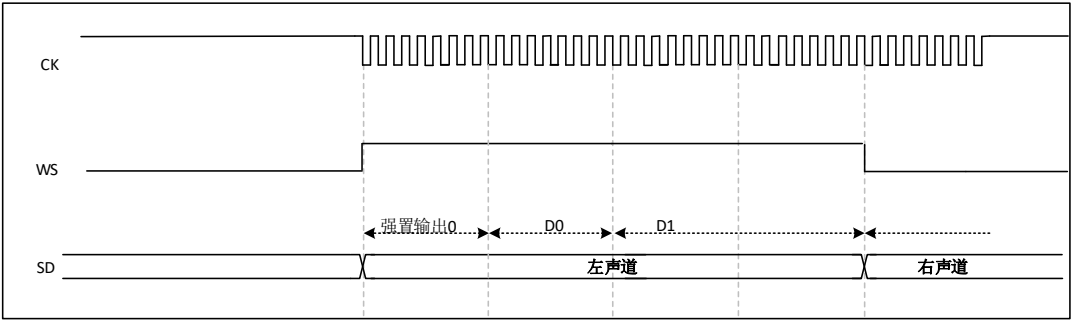


Figure 31-21 MSB-aligned standard, 16-bit extended to 32-bit, CPKOL = 1

The next TXE event occurs as soon as valid data starts being transmitted from the SD pin. During reception, an RXNE event occurs once valid data (not the 0x0000 part) is received.

31.4.2.3. **LSB-aligned standard**

This standard is similar to the MSB-aligned standard (no difference in 16-bit or 32-bit full precision frame formats).

The 16/32-bit full precision timing diagram is shown below. The transmitter changes data on the falling edge of the clock signal, and the receiver reads data on the rising edge.

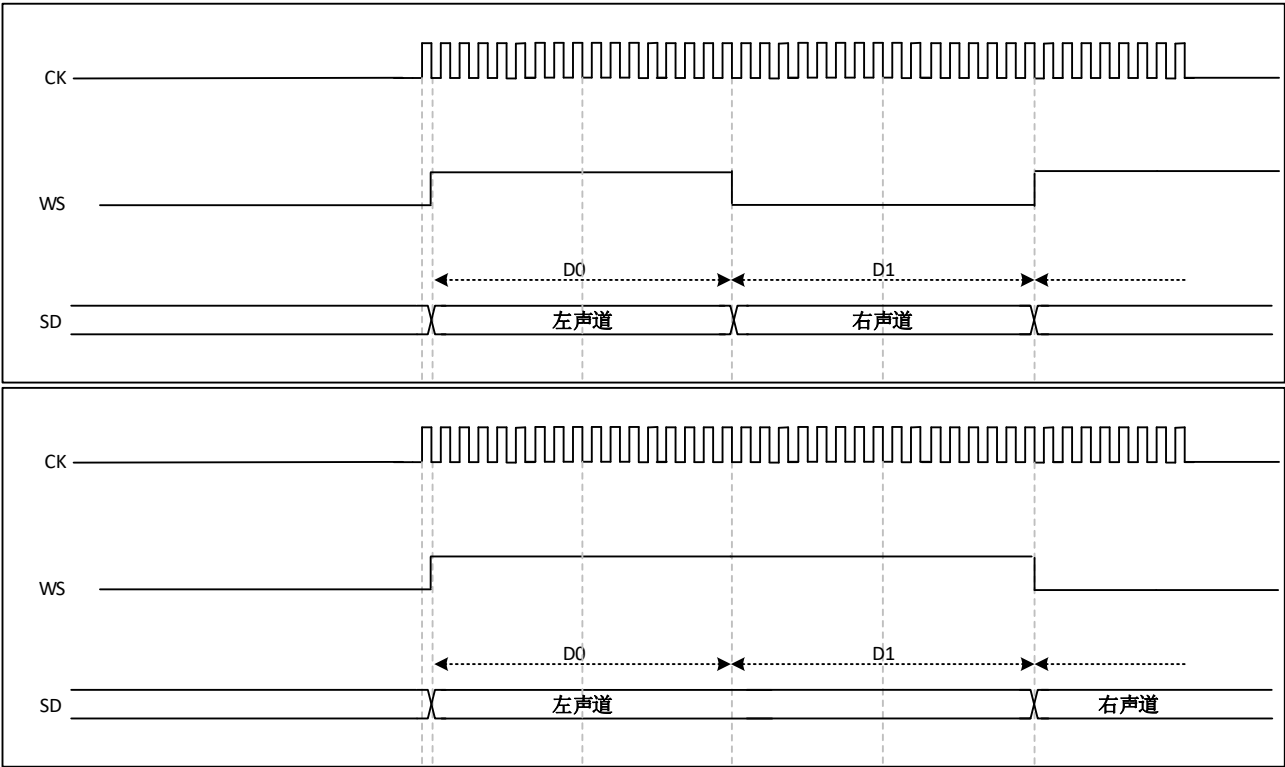


Figure 31-22 LSB-aligned 16-bit or 32-bit full precision, CKPOL = 0

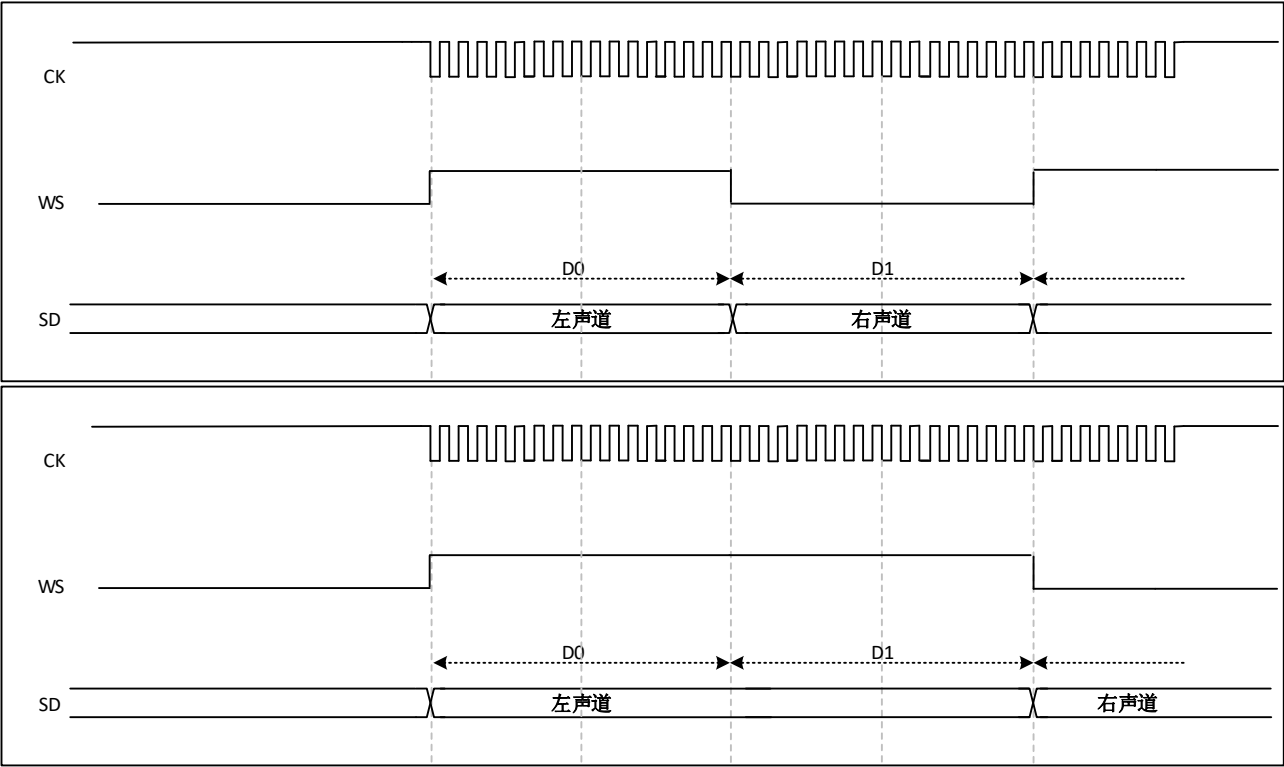


Figure 31-23 LSB-aligned 16-bit or 32-bit full precision, CKPOL = 0

The 24-bit frame timing diagram is as follows:

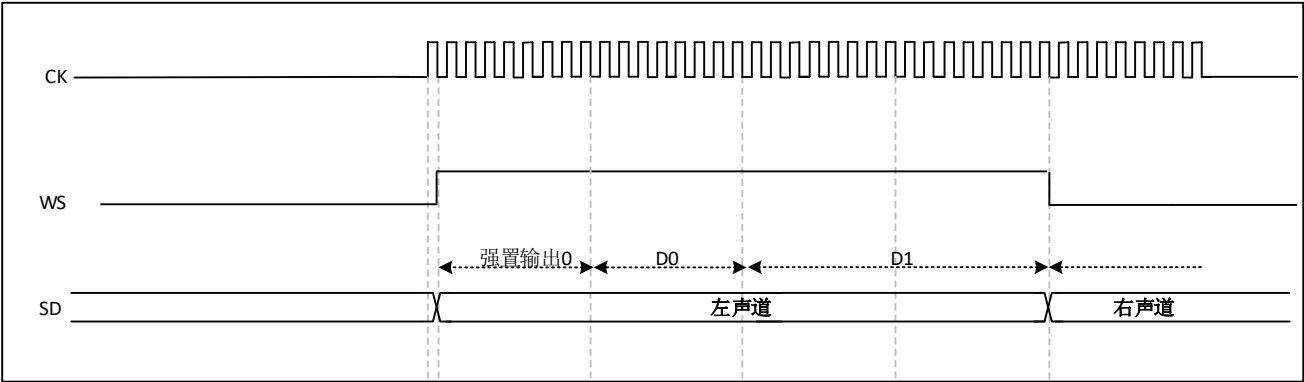


Figure 31-24 LSB-aligned 24-bit data, CKPOL=0

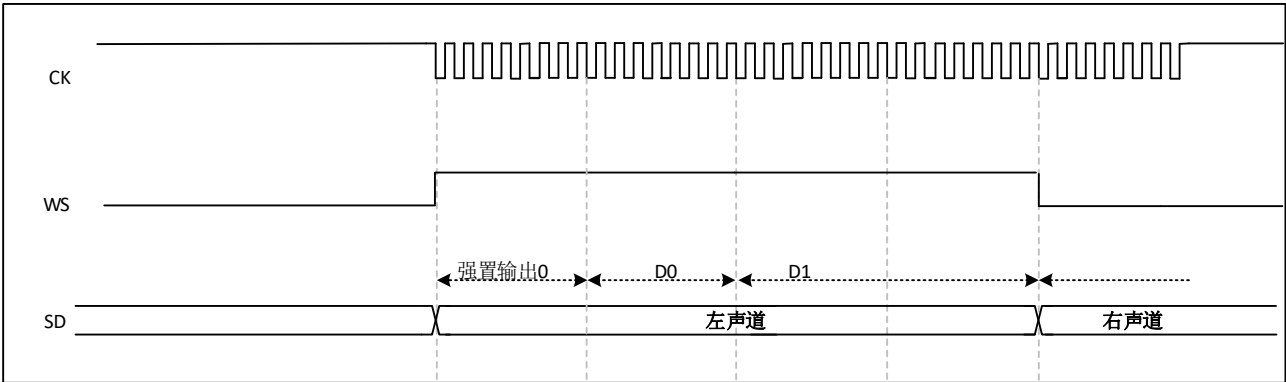


Figure 31-25 LSB-aligned 24-bit data, CKPOL=1

■ In transmit mode

To send data 0x3478AE, two write operations on the SPI_DR register are required via software or DMA. First, write 0xXX34 to the data register; second, write 0x78AE to the data register.

■ In receive mode

To receive data 0x3478AE, one read operation on the SPI_DR register is required for each of the two consecutive RXNE events. The first read is 0x0034, where only the lower 8 bits are meaningful; the second read is 0x78AE.

The following figure shows the timing for 16-bit data extended to a 32-bit channel frame:

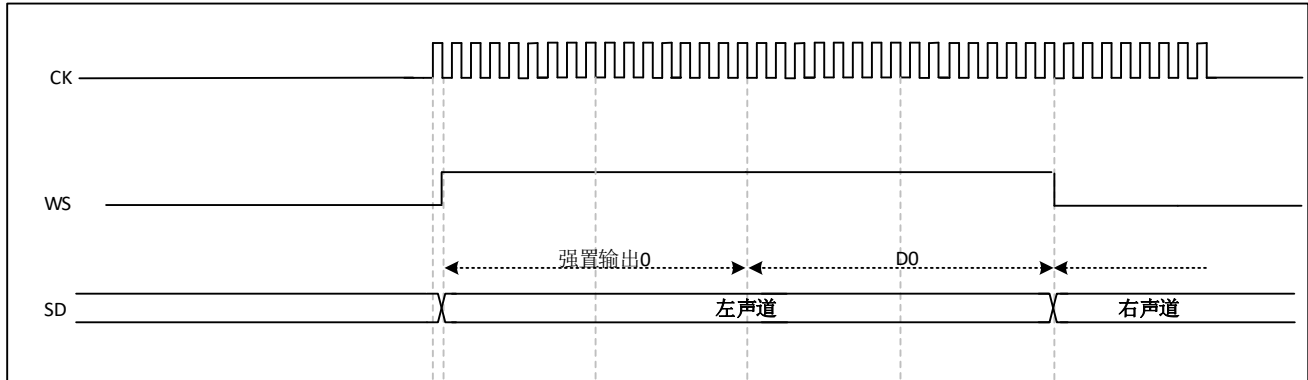


Figure 31-26 LSB-aligned 16-bit data extended to 32-bit packet frame, CKPOL=0

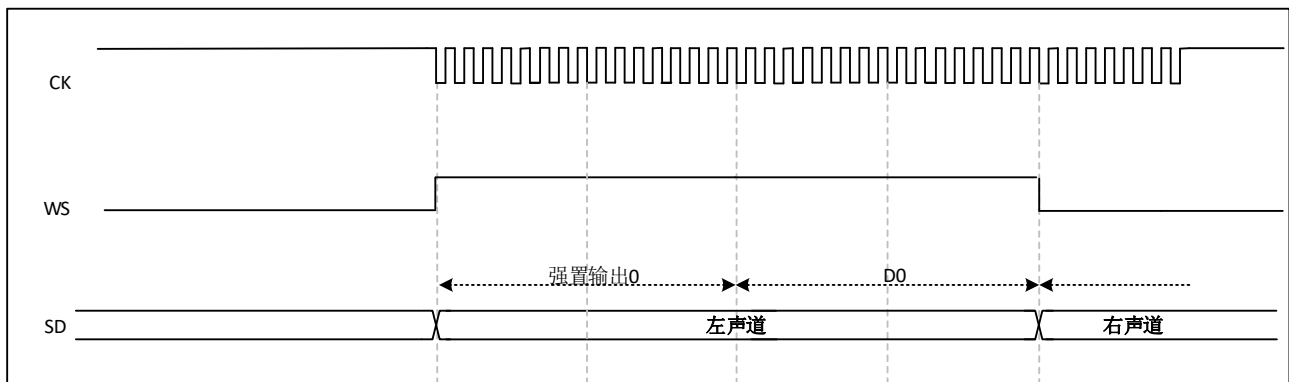


Figure 31-27 LSB-aligned 16-bit data extended to 32-bit packet frame, CKPOL=1

During the I2S configuration phase, if 16-bit data is selected to be extended to a 32-bit channel frame, only one access to the SPI_DR register is needed. At this point, the upper half-word (16-bit MSB) after extension to 32 bits is set to 0x0000 by hardware.

If the data to be transmitted or received is 0x76A3 (extended to 32 bits as 0x0000 76A3), only one operation on the SPI_DR register is needed, writing 0x76A3.

During transmission, when a TXE event occurs, the user needs to write the data to be sent (i.e., 0x76A3). The 0x0000 part used for extension to 32 bits is first sent out by hardware. Once valid data starts being output from the SD pin, the next TXE event occurs.

During reception, an RXNE event occurs once valid data (not the 0x0000 part) is received.

This provides more time between two read and write operations, preventing underflow or overflow situations.

31.4.2.4. PCM standard

In the PCM standard, there is no channel selection information. The PCM standard has two available frame structures, short frame or long frame, which can be selected by setting the PCMSYNC bit in the SPI_I2SCFGR register.

In PCM mode, output signals (WS and SD) are sampled on the rising edge of the CK signal. Input signals (WS and SD) are captured on the falling edge of CK.

Note that CK and WS are configured as outputs in master mode.

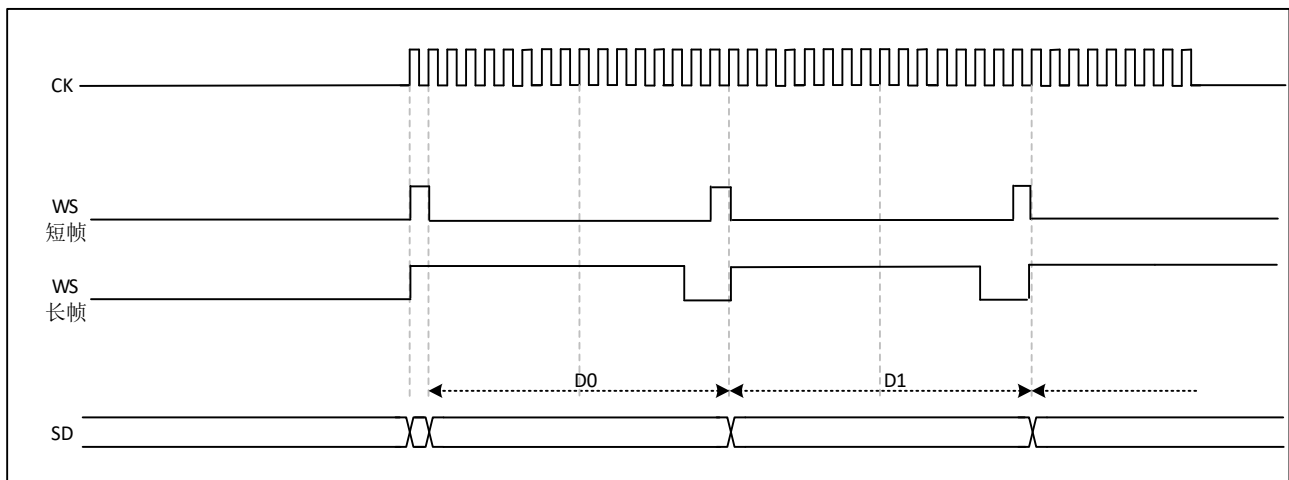


Figure 31-28 PCM standard waveform (16-bit, CKPOL=0)

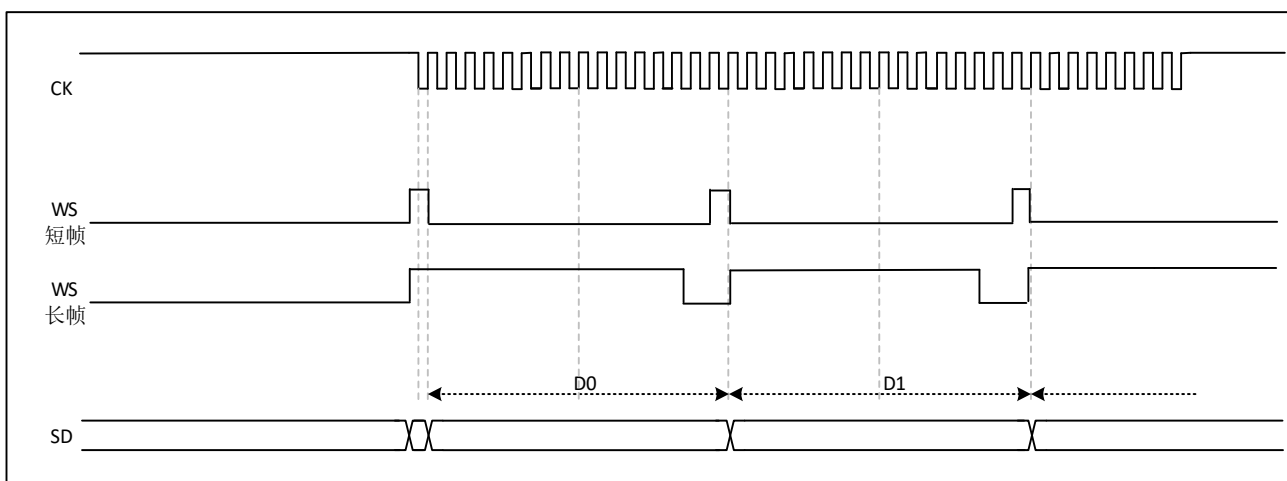


Figure 31-29 PCM Standard Waveform (16-bit, CKPOL=1)

For long frames, in master mode, the WS signal used for synchronization remains active for a fixed duration of 13 cycles.

For short frames, the WS signal used for synchronization lasts only 1 cycle.

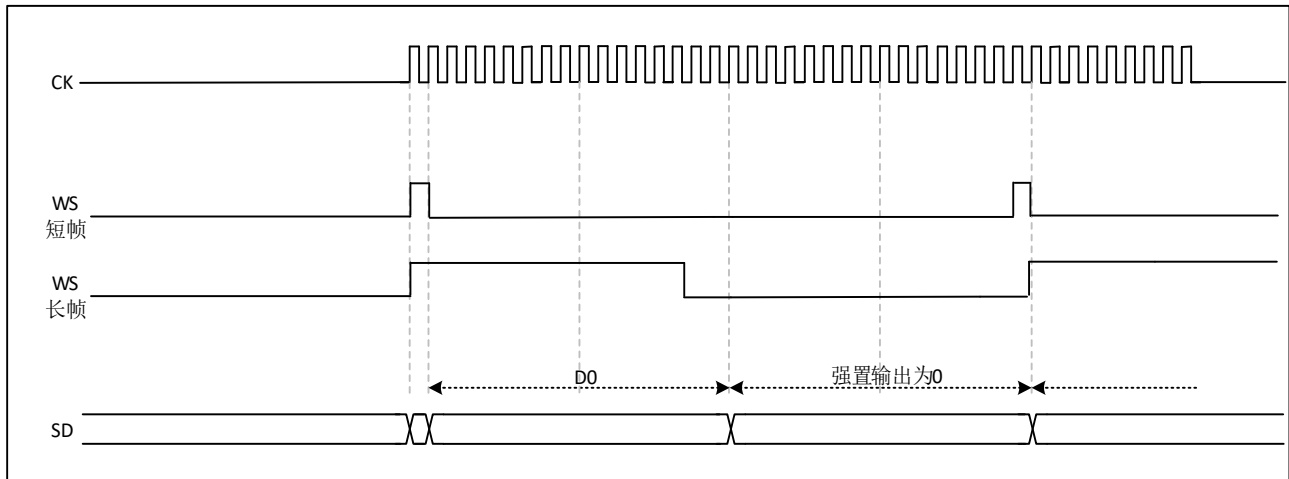


Figure 31-30 PCM Standard Waveform (16-bit extended to 32-bit packet frame, CKPOL=0)

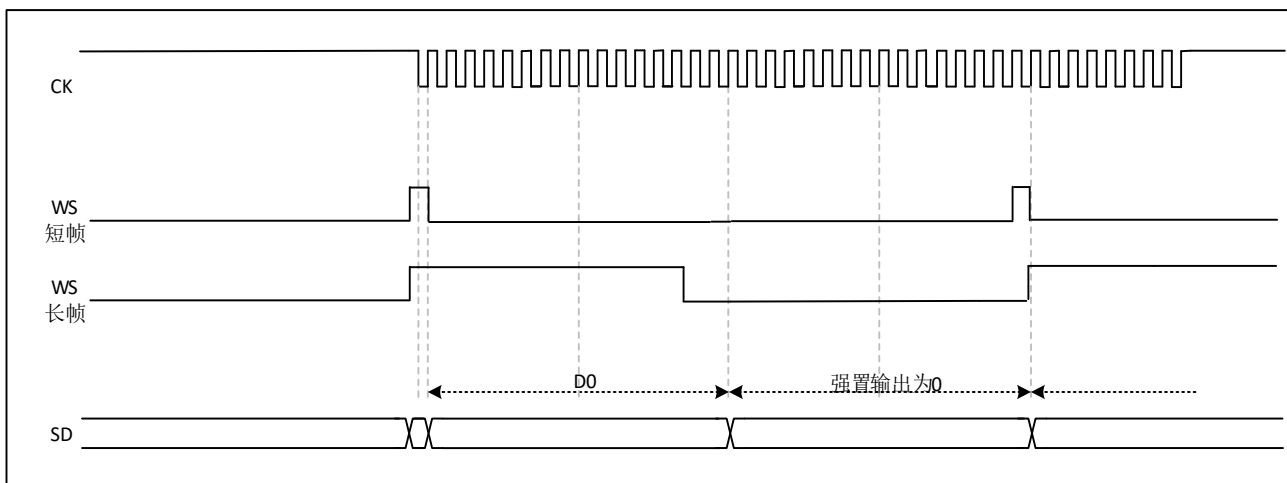


Figure 31-31 PCM Standard Waveform (16-bit extended to 32-bit packet frame, CKPOL=1)

Note: Regardless of the mode (master or slave) or synchronization method (short frame or long frame), the time difference between two consecutive frames and two synchronization signals (even in slave mode) must be determined by setting the DATLEN and CHLEN bits in the SPI_I2SCFGR register.

31.4.3. Clock Generator

The I2S bit rate determines the data flow on the I2S data line and the frequency of the I2S clock signal. That is,

I2S bit rate = Number of bits per channel × Number of channels × Audio sampling frequency

For a stereo audio signal with 16 bits per channel, the I2S bit rate is calculated as follows:

$$\text{I2S bit rate} = 16 \times 2 \times f_s;$$

If the packet length is 32 bits, then: I2S bit rate = $32 \times 2 \times F_s$

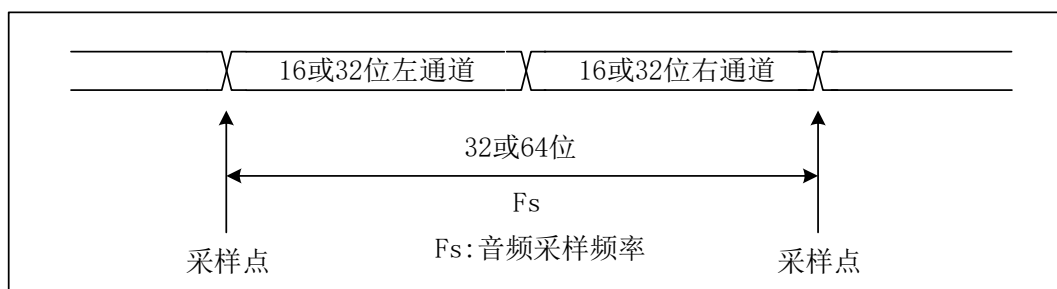


Figure 31-32 Audio Sampling Definition

In master mode, to obtain the desired audio frequency, the linear divider must be correctly configured.

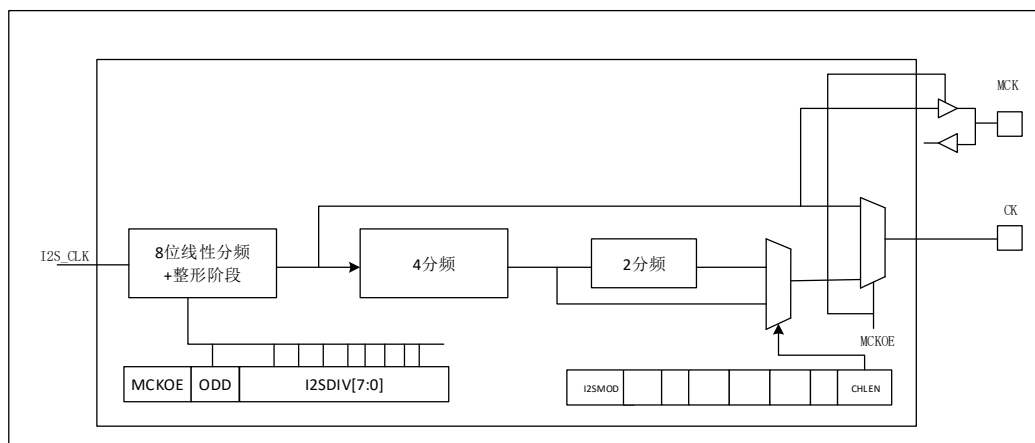


Figure 31-33 I2S Clock Generator Structure

The audio sampling frequency can be 96 kHz, 48 kHz, 44.1 kHz, 32 kHz, 22.05 kHz, 16 kHz, 11.025 kHz, or 8 kHz (or any value within this range). To obtain the desired frequency, the linear divider must be set according to the following formula when generating the master clock (MCKOE bit in register SPI_I2SPR is '1'):

When the frame length of the channel is 16 bits, $F_s = I2SxCLK / [(16 \times 2) \times ((2 \times I2SDIV) + ODD) \times 8]$

When the frame length of the channel is 32 bits, $F_s = I2SxCLK / [(32 \times 2) \times ((2 \times I2SDIV) + ODD) \times 4]$

When the master clock is turned off (MCKOE bit is '0'):

When the frame length of the channel is 16 bits, $F_s = I2SxCLK / [(16 \times 2) \times ((2 \times I2SDIV) + ODD)]$

When the frame length of the channel is 32 bits, $F_s = I2SxCLK / [(32 \times 2) \times ((2 \times I2SDIV) + ODD)]$

Using a standard 8 MHz HSE clock to obtain precise audio frequencies, as shown in the table below:

SYSCLK (MHz)	I2S_DIV		I2S_ODD		MCLK	De- sired fs (Hz)	Actual fs (Hz)		Error	
	16-bit	32-bit	16-bit	32-bit			16-bit	32-bit	16-bit	32-bit
72	11	6	1	0	None	96000	97826.09	93750	1.90%	2.34%
72	23	11	1	1	None	48000	47872.34	48913.04	0.27%	1.90%
72	25	13	1	0	None	44100	44117.65	43269.23	0.04%	1.88%
72	35	17	0	1	None	32000	32142.86	32142.86	0.45%	0.45%
72	51	25	0	1	None	22050	22058.82	22058.82	0.04%	0.04%
72	70	35	1	0	None	16000	15957.45	16071.43	0.27%	0.45%
72	102	51	0	0	None	11025	11029.41	11029.41	0.04%	0.04%
72	140	70	1	1	None	8000	8007.117	7978.723	0.09%	0.27%

72	2	2	0	0	Yes	96000	70312.5	70312.5	26.76%	26.76%
72	3	3	0	0	Yes	48000	46875	46875	2.34%	2.34%
72	3	3	0	0	Yes	44100	46875	46875	6.29%	6.29%
72	4	4	1	1	Yes	32000	31250	31250	2.34%	2.34%
72	6	6	1	1	Yes	22050	21634.62	21634.62	1.88%	1.88%
72	9	9	0	0	Yes	16000	15625	15625	2.34%	2.34%
72	13	13	0	0	Yes	11025	10817.31	10817.31	1.88%	1.88%
72	17	17	1	1	Yes	8000	8035.714	8035.714	0.45%	0.45%

31.4.4. I2S Master Mode

Configure the I2S to operate in master mode, with the serial clock output through the CK pin and the word select signal generated by the WS pin. The master clock (MCK) output can be enabled or disabled by configuring the MCKOE bit in the SPI_I2SPR register.

The steps are as follows:

1. Set the I2SDIV[7:0] bits in the SPI_I2SPR register to define the serial clock baud rate that matches the audio sampling frequency. Also define the ODD bit in the SPI_I2SPR register.
2. Configure the CKPOL bit to define the idle state level of the communication clock. If the master clock (MCK) needs to be provided to an external DAC/ADC audio device, set the MCKOE bit in the SPI_I2SPR register to 1. Calculate the values for I2SDIV and ODD based on the required MCK output conditions.
3. Set the I2SMOD bit in the SPI_I2SCFGR register to 1 to enable the I2S function. Configure the I2SSTD[1:0] and PCMSYNC bits to select the desired I2S standard, and set the CHLEN bit to define the number of data bits per audio channel. Additionally, configure the I2SCFG[1:0] bits in the SPI_I2SCFGR register to select the I2S master mode and direction.
4. If necessary, enable the required interrupt and DMA functions by configuring the SPI_CR2 register.
5. The I2SE bit in the SPI_I2SCFGR register must be set to '1'.
6. The WS and CK pins must be configured in output mode. If the MCKOE bit in the SPI_I2SPR register is '1', the MCK pin must also be configured in output mode.

1) Transmission process

The transmit process begins when a half-word (16-bit) of data is written to the transmit buffer.

Assume the first data written to the transmit buffer corresponds to the left channel data. When data is moved from the transmit buffer to the shift register, the TXE flag is set to '1'. At this time, the data corresponding to the right channel should be written to the transmit buffer. The CHSIDE flag indicates which channel the current data to be transmitted corresponds to. The value of the CHSIDE flag is updated when TXE is '1', so it is meaningful only when TXE is '1'. A complete data frame is only recognized after both the left and right channel data have been transmitted. Partial data frames, such as only left channel data, cannot be transmitted.

When the first bit of data is transmitted, the half-word data is transferred in parallel to the 16-bit shift register. Subsequent bits are then transmitted sequentially from the MOSI/SD pin in MSB-first order.

Each time data is moved from the transmit buffer to the shift register, the TXE flag is set to '1'. If the TXEIE bit in the SPI_CR2 register is '1', an interrupt is generated.

The operation of writing data depends on the selected I2S standard. To ensure continuous audio data transmission, it is recommended to write the next data to be transmitted to the SPI_DR register before the current transmission is completed.

It is recommended to wait for the TXE flag to be '1' and the BSY flag to be '0' before disabling the I2S function by clearing the I2SE bit to '0'.

2) Receive procedure

The configuration steps for the receive process are the same as those for the transmit process (refer to the aforementioned "Transmit Process"), except for step 3. The master receive mode must be selected by configuring I2SCFG[1:0].

Regardless of data or channel length, audio data is always received in 16-bit packets. Each time the receive buffer is filled, the RXNE flag is set to '1'. If the RXNEIE bit in the SPI_CR2 register is '1', an interrupt is generated. Depending on the configured data and channel length, receiving left or right channel data may require one or two transfers to the receive buffer. Reading the SPI_DR register clears the RXNE flag. CHSIDE is updated after each reception. Its value depends on the WS signal generated by the I2S unit. The data reading operation depends on the selected I2S standard.

If new data is received before the previously received data is read, an overflow occurs, and the OVR flag is set to '1'. If the ERRIE bit in the SPI_CR2 register is '1', an interrupt is generated, indicating an error.

To disable the I2S function, special operations are required to ensure the I2S module can properly complete the transmission cycle without initiating new data transfers. The operation process depends on the data configuration, channel length, and the selected audio protocol mode:

- 16-bit data extended to 32-bit channel length (DATLEN=00 and CHLEN=1), using LSB (least significant bit) alignment mode (I2SSTD=10)
 - a) Wait for the second-to-last RXNE=1 (n-1);
 - b) Wait for 17 I2S clock cycles (using software delay);
 - c) Disable I2S (I2SE=0).
- 16-bit data extended to 32-bit channel length (DATLEN=00 and CHLEN=1), using MSB (most significant bit) alignment, I2S, or PCM mode (I2SSTD=00, I2SSTD=01, or I2SSTD=11, respectively)
 - a) Wait for the last RXNE=1;
 - b) Wait for one I2S clock cycle (using software delay);
 - c) Disable I2S (I2SE=0).
- For all other combinations of DATLEN and CHLEN, and any audio mode selected by I2SSTD, disable I2S using the following method:
 - a) Wait for the second-to-last RXNE=1 (n-1);
 - b) Wait for one I2S clock cycle (using software delay);
 - c) Disable I2S (I2SE=0).

Note: The BSY flag remains low during transmission.

31.4.5. I2S Slave Mode

In slave mode, I2S can be set to transmit or receive mode. The configuration process for slave mode essentially follows the same procedure as configuring master mode. In slave mode, the I2S interface does not need to provide a clock. The clock signal and WS signal are provided by the external master I2S device and connected to the corresponding pins. Therefore, the user does not need to configure the clock.

1. Set the I2SMOD bit in the SPI_I2SCFGR register to activate the I2S function; set I2SSTD[1:0] to select the I2S standard; set DATLEN[1:0] to select the data bit length; set CHLEN to select the number of bits per channel. Set the I2SCFG[1:0] bits in the SPI_I2SCFGR register to select the data direction for I2S slave mode.
2. As needed, configure the SPI_CR2 register to enable the required interrupt and DMA functions.
3. The I2SE bit in the SPI_I2SCFGR register must be set to '1'.

1) Transmission process

The transmission process begins when the external master sends the clock signal and the NSS_WS signal requests data transmission. The slave must be enabled and the I2S data register must be written before the external master can start communication.

For I2S MSB-aligned and LSB-aligned modes, the first data item written to the data register corresponds to the left channel data. When communication starts, data is transferred from the transmit buffer to the shift register, and the TXE flag is set to '1'. At this point, the data item corresponding to the right channel should be written to the I2S data register.

The CHSIDE flag indicates which channel the current data to be transmitted corresponds to. Compared to the transmission process in master mode, in slave mode, CHSIDE depends on the WS signal from the external master I2S. This means the slave I2S must prepare the first data to be transmitted before receiving the clock signal generated by the master. A WS signal of '1' indicates the left channel is transmitted first.

Note: The I2SE bit should be set to '1' at least 2 PCLK clock cycles before the master I2S clock signal appears on the CK pin.

When the first bit of data is transmitted, the half-word data is transferred in parallel through the I2S internal bus to the 16-bit shift register, then the remaining bits are sequentially transmitted from the MOSI/SD pin with the most significant bit first. Each time data is transferred from the transmit buffer to the shift register, the TXE flag is set to '1'. If the TXEIE bit in the SPI_CR2 register is '1', an interrupt is generated. Note: Before writing data to the transmit buffer, ensure the TXE flag is set to '1'.

The operation for writing data depends on the selected I2S standard.

To ensure continuous audio data transmission, it is recommended to write the next data to be transmitted to the SPI_DR register before the current transmission is completed. If new data is still not written to the SPI_DR register before the first clock edge representing the next data transmission arrives, the underrun flag is set to '1' and may generate an interrupt, indicating a software transmission error. If the ERRIE bit in the SPI_CR2 register is '1', an interrupt is generated when the UDR

flag in the SPI_SR register is high. It is recommended to disable I2S at this point and then restart data transmission from the left channel.

It is recommended to wait for TXE=1 and BSY=0 before clearing the I2SE bit to disable I2S.

2) Receive procedure

The configuration steps are the same as the transmission procedure, except for the first point. Slave receive mode must be selected by configuring I2SCFG[1:0].

Regardless of data and channel length, audio data is always received in 16-bit packets. Each time the receive buffer is filled, the RXNE flag is set to '1'. If the RXNEIE bit in the SPI_CR2 register is '1', an interrupt is generated. Depending on the data and channel length settings, receiving left or right channel data may require one or two transfers to the receive buffer. CHSIDE is updated each time data is received (to be read from SPI_DR) and corresponds to the WS signal generated by the I2S unit. Reading the SPI_DR register clears the RXNE bit. The operation for reading data depends on the selected I2S standard. See Section 5.2 for details.

There are three status flags for monitoring the state of the I2S bus.

31.4.5.1. Busy flag (BSY)

The BSY flag is set and cleared by hardware (writing to this bit has no effect). This flag indicates the status of the I2S communication layer. When this bit is '1', it indicates that I2S communication is in progress, with one exception: in master receive mode (I2SCFG=11), the BSY flag remains low during reception. Before closing the I2S module, the BSY flag can be used to detect whether the transmission has ended, thus avoiding corruption of the last transmission. Therefore, the following procedure must be strictly followed. When transmission starts, the BSY flag is set to '1', unless the I2S module is in master receive mode.

This flag is cleared under the following conditions:

When the transmission ends (except in master transmit mode, where communication is continuous);

When the I2S module is turned off.

When communication is continuous:

In master transmit mode, the BSY flag remains high throughout the entire transmission.

In slave mode, the BSY flag goes low between each transmission and lasts for one I2S clock cycle.

Note: Do not use the BSY flag to handle each data transmission or reception; it is better to use the TXE and RXNE flags instead.

31.4.5.2. Transmit buffer empty flag (TXE)

This flag set to '1' indicates that the transmit buffer is empty and new data to be transmitted can be written to the transmit buffer. When data is already present in the transmit buffer, this flag is cleared to '0'. When I2S is turned off (I2SE bit is '0'), this flag is also '0'.

31.4.5.3. Receive buffer not empty flag (RXNE)

This flag set to '1' indicates that valid received data is present in the receive buffer. This bit is cleared to '0' when reading the SPI_DR register.

31.4.5.4. Channel flag (CHSIDE)

In transmit mode, this flag is refreshed when TXE is high, indicating the channel of the data being transmitted from the SD pin. If an underrun error occurs in slave transmit mode, the value of this flag

is invalid, and the I2S must be turned off and then on again before restarting communication. In receive mode, this flag is refreshed when data is received in register SPI_DR, indicating the channel of the received data. If an error occurs (such as overflow OVR), this flag is meaningless, and the I2S needs to be turned off and then on again (while modifying the I2S configuration if necessary).

In PCM standard, this flag is meaningless regardless of short frame or long frame format.

If the OVR or UDR flag in register SPI_SR is '1' and the ERRIE bit in register SPI_CR2 is '1', an interrupt will be generated. The interrupt flag can then be cleared by reading the SPI_SR register.

31.4.6. Error flags

The I2S unit has 2 error flags.

31.4.6.1. Underrun flag (UDR)

In slave transmit mode, if new data is still not written to the SPI_DR register when the first clock edge of data transmission arrives, this flag is set to '1'. This flag is valid only after the I2SMOD bit in the SPI_I2SCFGR register is set to '1'. An interrupt is generated if the ERRIE bit in the SPI_CR2 register is '1'. Clear this flag by reading the SPI_SR register.

31.4.6.2. Overflow flag (OVR)

If new data is received before the previously received data is read, an overflow occurs, and this flag is set to '1'. If the ERRIE bit in the SPI_CR2 register is '1', an interrupt is generated to indicate an error. At this time, the content of the receive buffer will not be refreshed with new data from the transmitting device. Reading the SPI_DR register returns the last correctly received data. All other 16-bit data sent by the transmitting device after the overflow occurs will be lost. Clear this flag by first reading the SPI_SR register and then reading the SPI_DR register.

31.4.7. I2S interrupt

Table 31-2 I2S interrupt request

Interrupt event	Event flag	Enable flag
Transmit buffer empty flag	TXE	TXEIE
Receive buffer not empty flag	RXNE	RXNEIE
Overflow flag	OVR	ERRIE
Underflow flag	UDR	ERRIE

31.4.8. DMA Function

Same as SPI, except for the CRC feature. Because there is no data transmission protection system in I2S mode.

31.5. SPI and I2S Registers

The corresponding SPI registers support 16-bit and 32-bit access, while the DR register supports 32-bit, 16-bit, and 8-bit access.

31.5.1. SPI Control Register 1 (SPI_CR1) (Not used in I2S mode)

Address offset: 0x00

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BI-DIMODE	BIDIOE	CRCE N	CRCNEX T	DF F	RXONL Y	SS M	SS I	LSBFIRS T	SP E	BR[2:0]			MST R	CPO L	CPH A
RW															

Bit	Name	R/W	Reset Value	Function
15	BIDIMODE	RW	0	<p>Bidirectional data mode enable</p> <p>0: Select "dual-wire bidirectional" mode; 1: Select "single-wire bidirectional" mode.</p> <p>Note: Not used in I2S mode.</p>
14	BIDIOE	RW	0	<p>Output enable in bidirectional mode</p> <p>Together with the BIDIMODE bit, determines the data output direction in "single-wire bidirectional" mode</p> <p>0: Output disable (receive-only mode); 1: Output enable (transmit-only mode).</p> <p>This "single-wire" data line is the MOSI pin on the master device and the MISO pin on the slave device.</p> <p>Note: Not used in I2S mode.</p>
13	CRCE N	RW	0	<p>Hardware CRC check enable</p> <p>0: Disable CRC calculation; 1: Enable CRC calculation.</p> <p>Note: This bit can only be written when SPI is disabled (SPE=0), otherwise an error will occur.</p> <p>Note: Not used in I2S mode.</p>
12	CRCNEXT	RW	0	<p>Next transmit CRC</p> <p>0: Next transmitted value comes from the transmit buffer. 1: Next transmitted value comes from the CRC transmit register.</p> <p>Note: This bit should be set immediately after writing the last data to the SPI_DR register.</p> <p>Note: Not used in I2S mode.</p>
11	DFF	RW	0	Data frame format

Bit	Name	R/W	Reset Value	Function
				<p>0: Use 8-bit data frame format for transmission/reception;</p> <p>1: Use 16-bit data frame format for transmission/reception.</p> <p>Note: This bit can only be written when SPI is disabled (SPE=0), otherwise an error occurs.</p> <p>Note: Not used in I2S mode.</p>
10	RXONLY	RW	0	<p>Receive only</p> <p>This bit, together with the BIDIMODE bit, determines the transmission direction in "Two-line unidirectional" mode. In a multi-slave configuration, this bit is set to 1 on unaddressed slaves so that only the addressed slave has output, preventing data collisions on the bus.</p> <p>0: Full duplex (Transmit and receive)</p> <p>1: Output disabled (Receive-only mode)</p> <p>Note: Not used in I2S mode.</p>
9	SSM	RW	0	<p>Software slave management</p> <p>When SSM is set, the level on the NSS pin is determined by the value of the SSI bit.</p> <p>0: Disable software slave management</p> <p>1: Enable software slave management</p> <p>Note: Not used in I2S mode.</p>
8	SSI	RW	0	<p>Internal slave select</p> <p>This register is only valid when SSM=1. This register determines the level on NSS, and I/O operations on the NSS pin are invalid.</p> <p>Note: Not used in I2S mode.</p>
7	LSBFIRST	RW	0	<p>Frame format</p> <p>0: Send MSB first</p> <p>1: Send LSB first</p> <p>The value of this register cannot be changed during communication.</p> <p>Note: Not used in I2S mode.</p>
6	SPE	RW	0	<p>SPI enable</p> <p>0: Disable SPI</p> <p>1: Enable SPI</p> <p>Note: Not used in I2S mode.</p>
5: 3	BR[2:0]	RW	0	Baud rate control

Bit	Name	R/W	Reset Value	Function
				000: fPCLK/2 001: fPCLK/4 010: fPCLK/8 011: fPCLK/16 100: fPCLK/32 101: fPCLK/64 110: fPCLK/128 111: fPCLK/256 The value of this register cannot be changed during communication. Note: Not used in I2S mode. In slave mode, the maximum baud rate only supports fPCLK/4.
2	MSTR	RW	0	Master selection 0: Configured as slave device 1: Configured as master device The value of this register cannot be changed during communication. Note: Not used in I2S mode.
1	CPOL	RW	0	Clock polarity 0: SCK remains low during idle state 1: SCK remains high during idle state The value of this register cannot be changed during communication. Note: Not used in I2S mode.
0	CPHA	RW	0	Clock phase 0: Data sampling starts from the first clock edge 1: Data sampling starts from the second clock edge The value of this register cannot be changed during communication. Note: Not used in I2S mode.

31.5.2. SPI Control Register 2 (SPI_CR2)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res								TXEIE	RXNEIE	ERRIE	CLRTXFIFO	Res	SSOE	TXDMAEN	RXDMAEN
-								RW	RW	RW	RW	-	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
15: 8	Reserved	-	-	Reserved
7	TXEIE	RW	0	Transmit buffer empty interrupt enable. 0: Disable TXE interrupt. 1: Enable TXE interrupt. An interrupt request is generated when TXE=1.
6	RXNEIE	RW	0	Receive buffer not empty interrupt enable. 0: Disable RXNE interrupt. 1: Enable RXNE interrupt. An interrupt request is generated when RXNE=1.
5	ERRIE	RW	0	Error interrupt enable. When an error (CRCERR, OVR, MODF) occurs, this bit controls whether an interrupt is generated. 0: Disable error interrupt. 1: Enable error interrupt.
4	CLRTXFIFO	RW	0	Clear TXFIFO Set by software, reset by hardware 0: No effect 1: Clear TXFIFO Note: This bit can only be written when SPI is disabled (SPE=0). Otherwise, it is invalid.
3	Res	-	-	Reserved
2	SSOE	RW	0	SS output enable 0: Disable SS output in master mode. The device can operate in multi-master mode. 1: Enable SS output in master mode. The device cannot operate in multi-master mode. Note: Not used in I2S mode.
1	TXDMAEN	RW	0	Transmit buffer DMA enable When this bit is set, a DMA request is issued once the TXE flag is set 0: Disable transmit buffer DMA 1: Enable transmit buffer DMA.
0	RXDMAEN	RW	0	Receive buffer DMA enable When this bit is set, a DMA request is issued once the RXNE flag is set 0: Disable receive buffer DMA 1: Enable receive buffer DMA.

31.5.3. SPI Status Register (SPI_SR)

Address offset: 0x08

Reset value: 0x0000 0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res			FTLVL [1: 0]		FRLVL [1: 0]		Res	BSY	OVR	MODF	CRCERR	UDR	CHSIDE	TXE	RXNE
-			R	R	R	R	-	R	R	R	RC_W0	R	R	R	R

Bit	Name	R/W	Reset Value	Function
15: 13	Reserved	-	-	Reserved
12: 11	FTLVL	R	0	FIFO send level. Set by hardware, cleared by hardware 00: FIFO empty 01: 1/4 FIFO 10: 1/2 FIFO 11: FIFO full (considered full when FIFO threshold is greater than 1/2) Note: This bit is not used in I2S mode.
10: 9	FRLVL	R	0	FIFO receive level. Set by hardware, cleared by hardware 00: FIFO empty 01: 1/4 FIFO 10: 1/2 FIFO 11: FIFO full Note: These bits are not used in I ² S mode or SPI receive-only mode with CRC check.
8	Reserved	-	-	Reserved
7	BSY	R	0	Busy flag. 0: SPI is not busy 1: SPI is busy in communication, or the transmit buffer is not empty. This bit is set or cleared by hardware.
6	OVR	R	0	Overflow flag 0: No overflow error occurred 1: Overflow error occurred This bit is set by hardware and cleared by software sequence. (Overflow and underflow sequences differ).
5	MODF	R	0	Mode fault 0: No mode fault occurred 1: Mode fault occurred

Bit	Name	R/W	Reset Value	Function
				This bit is set by hardware and cleared by software sequence. Note: Not used in I2S mode.
4	CRCERR	RC_W0	0	CRC error flag 0: Received CRC value matches the value in the SPI_RXCRCR register 1: Received CRC value does not match the value in the SPI_RXCRCR register This bit is set by hardware and reset by writing '0' in software. Note: Not used in I2S mode.
3	UDR	R	0	Underrun flag 0: No underrun occurred 1: Underrun occurred This flag is set to '1' by hardware and cleared to '0' by a software sequence. Note: Not used in SPI mode.
2	CHSIDE	R	0	Channel 0: Left channel needs to be transmitted or received; 1: Right channel needs to be transmitted or received. Note: Not used in SPI mode. Meaningless in PCM mode.
1	TXE	R	1	Transmit buffer is empty. 0: Transmit buffer is not empty 1: Transmit buffer is empty
0	RXNE	R	0	Receive buffer is not empty. 0: Receive buffer is not empty 1: Receive buffer is empty

31.5.4. SPI Data Register (SPI_DR)

Address offset: 0x0C

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
15: 0	DR[15:0]	RW	0	Data register. Data to be transmitted or received.

				<p>The data register serves as the interface for both RxFIFO and TxFIFO. When reading data, the actual access is to the RxFIFO, and when writing data, the actual access is to the TxFIFO.</p> <p>Note: Data is always right-aligned. Unused bits are ignored when writing to the register and read as zero when reading the register. The Rx threshold setting must always correspond to the currently used read access.</p>
--	--	--	--	---

31.5.5. SPI CRC polynomial register (SPI_CRCPR)

Address offset: 0x10

Reset value: 0x0000 0007

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

Bit	Name	R/W	Reset Value	Function
15: 0	CRCPOLY[15:0]	RW	0x7	<p>CRC Polynomial Register</p> <p>This register contains the polynomial used in CRC calculation.</p> <p>Its reset value is 0x0007, and other values can be set according to the application.</p> <p>Note: Not used in I2S mode.</p> <p>Note: The polynomial value can only be odd; even values are not supported.</p>

31.5.6. SPI Rx CRC Register (SPI_RXCRCR)

Address offset: 0x14

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
15: 0	RxCRC[15:0]	R	0	<p>Receive CRC Register</p> <p>When CRC calculation is enabled, RXCR[15:0] contains the CRC value calculated based on the received bytes. This register is reset when '1' is</p>

Bit	Name	R/W	Reset Value	Function
				<p>written to the CRCEN bit in SPI_CR1. CRC calculation uses the polynomial in SPI_CRCPR.</p> <p>When the data frame format is set to 8-bit, only the lower 8 bits are involved in the calculation, following the CRC8 method; when the data frame format is 16-bit, all 16 bits in the register participate in the calculation, adhering to the CRC16 standard.</p> <p>Note: Reading this register when the BSY flag is '1' may return incorrect values.</p> <p>Note: Not used in I2S mode.</p>

31.5.7. SPI Tx CRC Register (SPI_TXCRCR)

Address offset: 0x18

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
15: 0	TxCRC[15:0]	R	0	<p>Transmit CRC Register</p> <p>When CRC calculation is enabled, TXCRC[15:0] contains the CRC value calculated based on the bytes to be transmitted. This register is reset when '1' is written to the CRCEN bit in SPI_CR1. CRC calculation uses the polynomial in SPI_CRCPR.</p> <p>When the data frame format is set to 8 bits, only the lower 8 bits are involved in the calculation, following the CRC8 method; when the data frame format is 16 bits, all 16 bits in the register participate in the calculation, following the CRC16 standard.</p> <p>Note: Reading this register when the BSY flag is '1' may return incorrect values.</p> <p>Note: Not used in I2S mode.</p>

31.5.8. SPI_I2S Configuration Register (SPI_I2S_CFGR)

Address offset: 0x1C

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res				I2SMOD	I2SE	I2SCFG		PCMSYNC	Res	I2SSTD		CKPOL	DATLEN		CHLEN
-				RW	RW	RW		RW	-	RW		RW	RW		RW

Bit	Name	R/W	Reset Value	Function
15: 12	Reserved	-	-	Reserved
11	I2SMOD	RW	0	I2S mode selection 0: Select SPI mode 1: Select I2S mode Note: This bit can only be set when SPI or I2S is disabled.
10	I2SE	RW	0	I2S enable 0: I2S disabled; 1: I2S enabled. Note: Not used in SPI mode.
9: 8	I2SCFG	RW	0	I2S mode setting 00: Slave transmit 01: Slave receive 10: Master transmit 11: Master receive Note: This bit can only be set when I2S is disabled. Not used in SPI mode.
7	PCMSYNC	RW	0	PCM frame sync 0: Short frame sync 1: Long frame sync Note: This bit is only meaningful when I2SSTD = 11 (using PCM standard). Not used in SPI mode.
6	Reserved	-	-	Reserved
5: 4	I2SSTD	RW	0	I2S standard selection 00: I2S Philips standard 01: High-byte aligned standard (left-aligned) 10: Low-byte aligned standard (right-aligned) 11: PCM standard

Bit	Name	R/W	Reset Value	Function
				Note: For proper operation, this bit can only be set when I2S is disabled. Not used in SPI mode.
3	CKPOL	RW	0	Inactive state clock polarity 0: I2S clock inactive state is low level 1: I2S clock inactive state is high level Note: For proper operation, this bit can only be set when I2S is disabled. Not used in SPI mode. The CKPOL bit does not affect the CK edge sensitivity used for receiving or transmitting SD and WS signals
2: 1	DATLEN	RW	0	Data length to be transmitted 00: 16-bit data length 01: 24-bit data length 10: 32-bit data length 11: Not allowed Note: For proper operation, this bit can only be set when I2S is disabled. Not used in SPI mode.
0	CHLEN	RW	0	Channel length (number of data bits per audio channel) 0: 16-bit width 1: 32-bit width Writing to this bit is only meaningful when DATLEN = 00; otherwise, the channel length is fixed at 32 bits by hardware. Note: For proper operation, this bit can only be set when I2S is disabled. Not used in SPI mode.

31.5.9. SPI_I2S Prescaler Register (SPI_I2SPR)

Address offset: 0x20

Reset value: 0x0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res						MCKOE	ODD	I2SDIV							
-						RW	RW	RW							

Bit	Name	R/W	Reset Value	Function
15: 10	Reserved	-	-	Reserved
9	MCKOE	RW	0	<p>Master clock output enable</p> <p>0: Master clock output disabled</p> <p>1: Master clock output enabled</p> <p>Note: For proper operation, this bit can only be set when I2S is disabled. This bit is used only in I2S master mode.</p> <p>Not used in SPI mode.</p>
8	ODD	RW	0	<p>Odd coefficient prescaler</p> <p>0: Actual division factor = $I2SDIV * 2$</p> <p>1: Actual division factor = $(I2SDIV * 2) + 1$</p> <p>Note: For proper operation, this bit can only be set when I2S is disabled. This bit is used only in I2S master mode.</p> <p>Not used in SPI mode.</p>
7: 0	I2SDIV	RW	0x2	<p>I2S linear prescaler</p> <p>Do not set $I2SDIV [7:0] = 0$ or $I2SDIV [7:0] = 1$</p> <p>Note: For proper operation, this bit can only be set when I2S is disabled. This bit is used only in I2S master mode.</p> <p>Not used in SPI mode.</p>

32. Universal Synchronous Asynchronous Receiver Transmitter (USART)

32.1. Introduction

The Universal Synchronous Asynchronous Receiver Transmitter (USART) provides a flexible method for full-duplex data exchange with external devices using the industrial standard NRZ asynchronous serial data format. The USART uses a fractional baud rate generator to provide a wide range of baud rate selections.

It supports synchronous one-way communication and half-duplex single-wire communication, as well as LIN (Local Interconnect Network), smart card protocol, IrDA (Infrared Data Association) ENDEC specification, and modem (CTS/RTS) operation. It also allows for multiprocessor communication.

High-speed data communication using DMA with multi-buffer configuration.

32.2. USART main features

- 2 full-featured USARTs (USART1 and USART2), and 2 USARTs without LIN, SCEN, IRDA support (USART3 and USART4)
- Full-duplex asynchronous communication
- Programmable baud rate generator shared between transmitter and receiver, up to 4.5 Mbit/s
- Configurable data word length (8-bit or 9-bit)
- Configurable stop bits—supports 0.5, 1, 1.5, or 2 stop bits
- Transmitter provides clock for synchronous transmission
- Single-wire half-duplex communication
- Independent transmitter and receiver enable bits
- Parity control
 - Transmit parity bit
 - Parity check for received data
- Detection flag
 - Receive buffer full
 - Transmit buffer empty
 - End of transmission flag
- LIN master capability to transmit sync break character and LIN slave capability to detect break character
 - When USART is configured as LIN, generates 13-bit break character and detects 10/11-bit break character
- IRDA SIR encoder decoder
 - Supports 3/16-bit duration in normal mode
- Smartcard emulation function
 - Smartcard interface supports asynchronous smartcard protocol defined in ISO7816-3 standard
 - 0.5 and 1.5 stop bits for smartcard

- Four error detection flags
 - Overflow error
 - Noise error
 - Frame error
 - Parity error
- 10 interrupt sources with flags
 - CTS change
 - LIN break character detection
 - Transmit data register empty
 - Transmission complete
 - Receive data register full
 - Bus idle detected
 - Overflow error
 - Frame error
 - Noise error
 - Parity error
- Multiprocessor communication. If the address does not match, enter mute mode
- Wake-up from mute mode (via idle bus detection or address flag detection)

Two methods to wake up the receiver: address bit (MSB, 9th bit), bus idle

32.3. USART functional description

The USART interface connects to other devices through three pins. Any USART bidirectional communication requires at least two pins: receive data input (RX) and transmit data output (TX).

RX: Receive data serial input. Data is recovered by distinguishing it from noise through oversampling techniques.

TX: Transmit data output. When the transmitter is disabled, the output pin returns to its I/O port configuration. When the transmitter is enabled and not sending data, the TX pin remains high. In single-wire and smartcard modes, this I/O pin is used for both transmitting and receiving data.

- 1) The bus should be in idle state before transmitting or receiving
- 2) A start bit
- 3) A data word (8 or 9 bits), least significant bit first
- 4) 0.5, 1, 1.5, or 2 stop bits, indicating the end of the data frame
- 5) Using fractional baud rate generator: representation method with 12-bit integer and 4-bit fraction
- 6) A status register (USART_SR)
- 7) Data Register (USART_DR)
- 8) A baud rate register (USART_BRR) with 12-bit integer and 4-bit fraction
- 9) A guard time register (USART_GTPR) in smartcard mode

The following pins are required in synchronous mode:

CK: Transmitter clock output.

This pin outputs the clock for synchronous transmission (no clock pulses during Start and Stop bits, optionally, a clock pulse can be sent after the last data bit). Data can be received synchronously on RX. This can be used to control external devices with shift registers (e.g., LCD drivers). Clock phase and polarity are both software-programmable.

The following pins are required in hardware flow control mode:

- **nCTS: Clear to Send.** If high, it blocks the next data transmission after the current data transfer ends.
- **nRTS: Request to Send.** If low, it indicates the USART is ready to receive data.

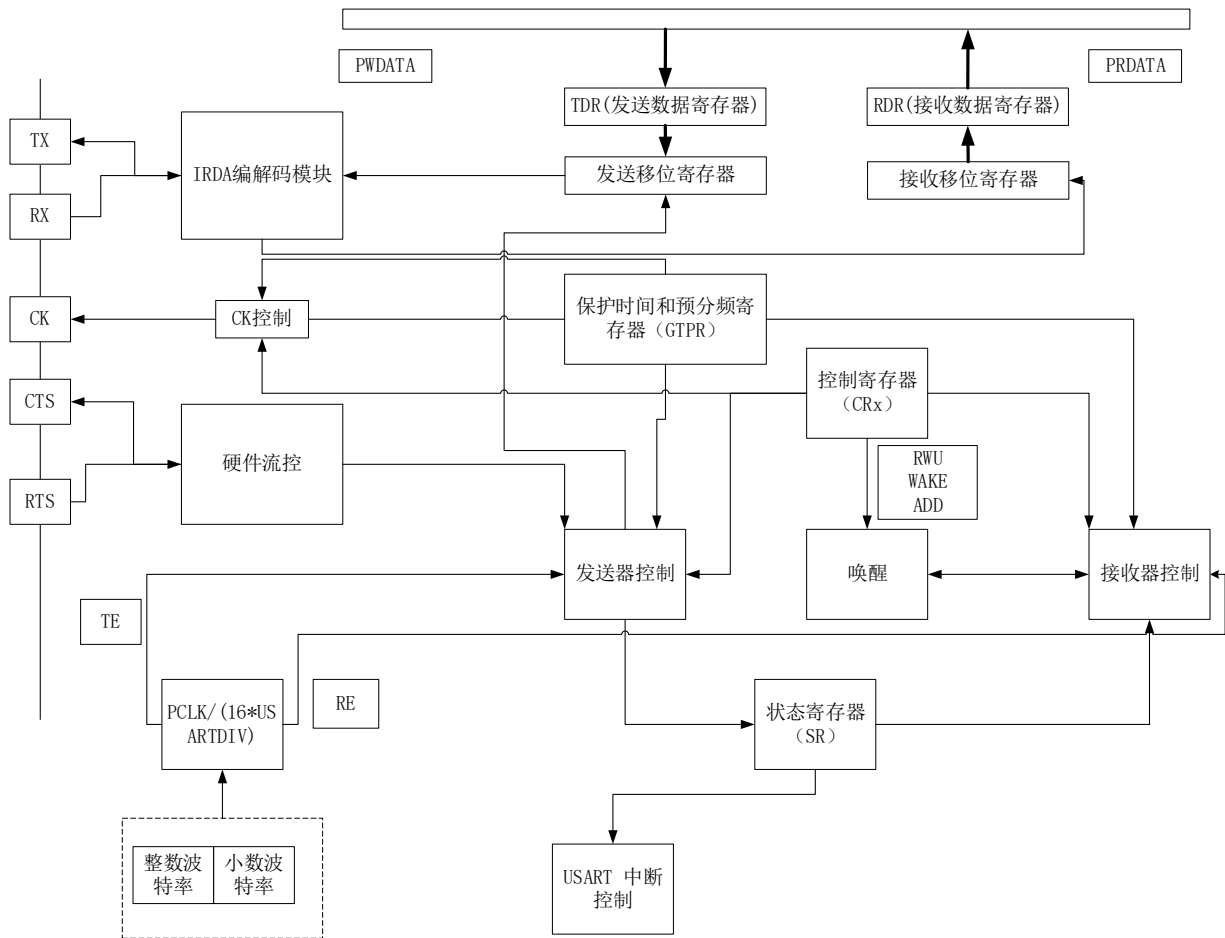


Figure 32-1 USART block diagram

32.3.1. USART feature description

The word length can be programmed to 8 or 9 bits by setting the M bit in the USART_CR1 register. During the start bit, the TX pin is at a low level, and during the stop bit, it is at a high level.

The idle symbol is considered as a complete data frame consisting entirely of '1's, followed by the start bit of the next frame containing data (the number of '1's includes the stop bits).

The break symbol is considered as receiving all '0's during one frame period (including the stop bit period, which is also '0'). At the end of the break symbol, the transmitter inserts 1 or 2 stop bits ('1') in response to the start bit.

Transmission and reception are driven by a shared baud rate generator, which generates clocks for each when the transmitter and receiver enable bits are set, respectively.

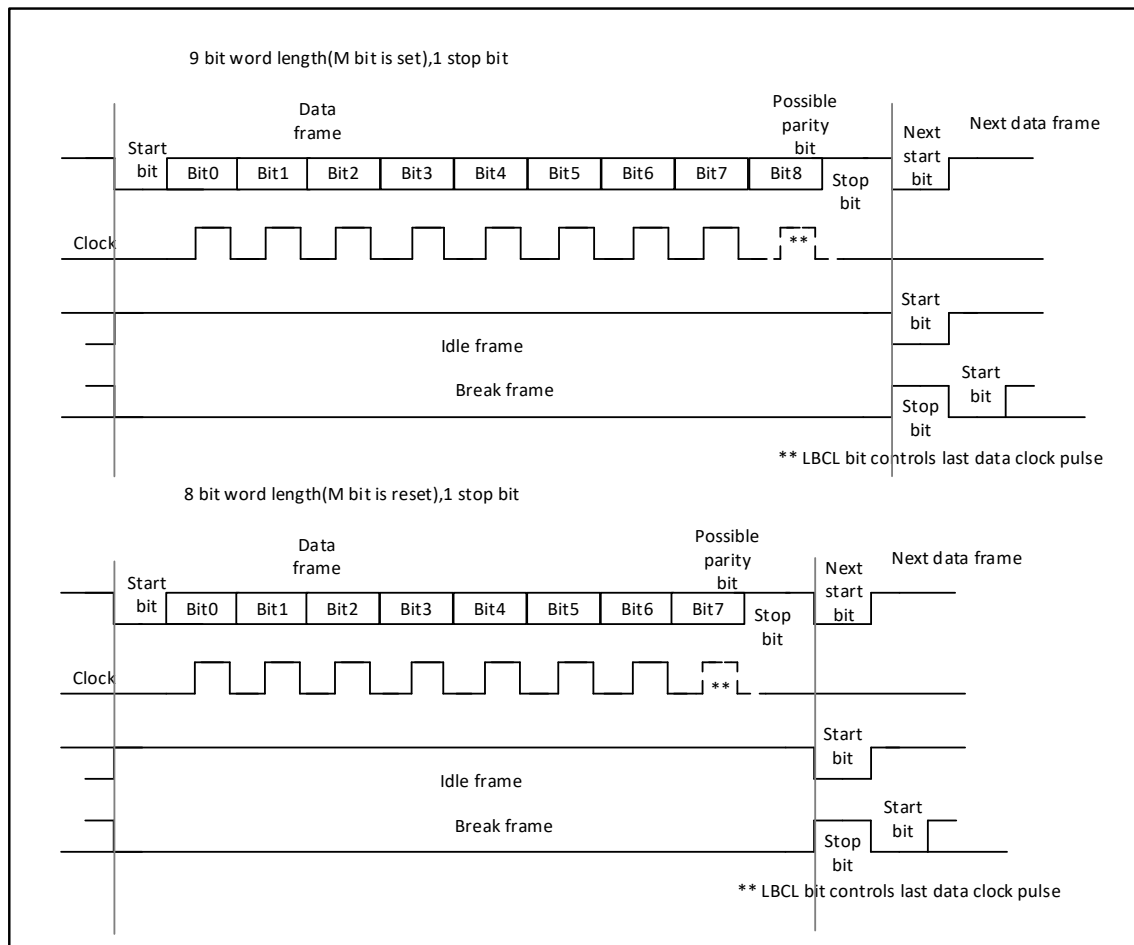


Figure 32-2 Word length setting

32.3.2. Transmitter

The transmitter sends 8-bit or 9-bit data words depending on the state of the M bit. When the Transmit Enable (TE) bit is set, the data in the transmit shift register is output on the TX pin, and the corresponding clock pulses are output on the CK pin.

32.3.2.1. Character transmission

During USART transmission, the least significant bit of the data is shifted out first on the TX pin. In this mode, the USART_DR register contains a buffer between the internal bus and the transmit shift register.

Each character is preceded by a low-level start bit; followed by a configurable number of stop bits. The USART supports multiple stop bit configurations: 0.5, 1, 1.5, and 2 stop bits.

Note:

The TE bit must not be reset during data transmission, as this will corrupt the data on the TX pin because the baud rate counter stops counting. The current data being transmitted will be lost.

An idle frame will be sent after the TE bit is activated.

32.3.2.2. Configurable Stop Bits

The number of stop bits sent with each character can be programmed using bits 13 and 12 of the control register 2.

- 1) One stop bit: The default value for the number of stop bits.
- 2) Two stop bits: Can be used in normal USART mode, single-wire mode, and modem mode.

The idle frame includes the stop bit.

A break frame is 10 bits of low level followed by a stop bit (when $m=0$); or 11 bits of low level followed by a stop bit (when $m=1$). It is not possible to transmit a longer break frame (length greater than 10 or 11 bits).

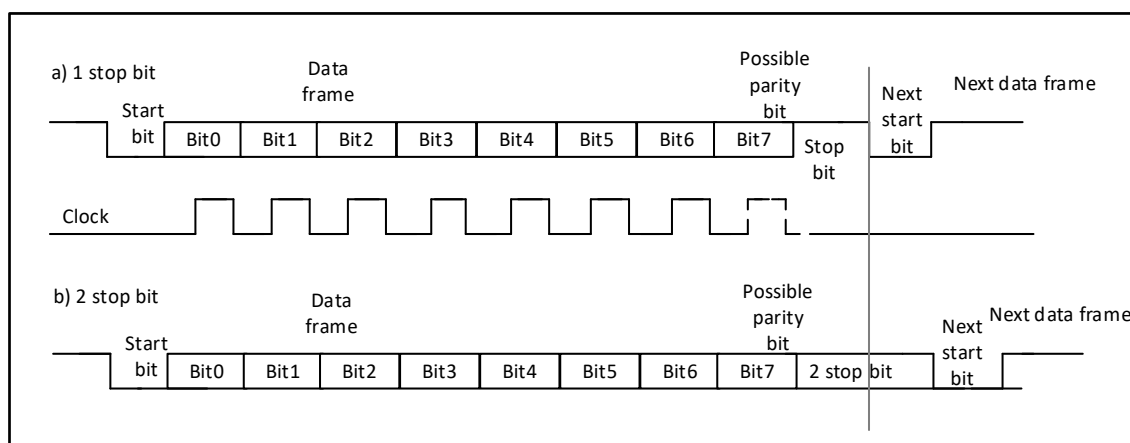


Figure 32-3 Configuring Stop Bits

Configuration steps:

- 1) Activate the USART by setting the UE bit in the USART_CR1 register.
- 2) Program the M bit in USART_CR1 to define the word length.
- 3) In USART_CR2. Program the number of stop bits in STOP.
- 4) If multi-buffer communication is used, configure the DMA enable bit (DMAT) in USART_CR3. Configure the DMA registers as described in multi-buffer communication.
- 5) Select the desired baud rate using the USART_BRR register.
- 6) Set the TE bit in USART_CR1 to send an idle frame as the first data transmission.
- 7) Write the data to be transmitted into the USART_DR register (this action clears the TXE bit). In the case of a single buffer, repeat step 7 for each data to be transmitted.
- 8) After writing the last data word into the USART_DR register, wait for TC=1, which indicates the end of transmission for the last data frame. Before disabling the USART or entering stop mode, ensure the transmission is complete to avoid corrupting the last transmission.

32.3.2.3. Single-byte communication

Clearing the TXE bit is always done by writing to the data register. The TXE bit is set by hardware and indicates:

- Data has been moved from TDR to the shift register, and data transmission has begun.
- The TDR register is emptied.

- The next data can be written to the USART_DR register without overwriting the previous data.

If the TXEIE bit is set, this flag will generate an interrupt.

If the USART is currently transmitting data, writing to the USART_DR register stores the data in the TDR register and copies it into the shift register at the end of the current transmission.

If the USART is not transmitting data and is idle, writing to the USART_DR register directly places the data into the shift register, transmission begins, and the TXE bit is immediately set.

When a frame transmission is complete (after the stop bit is sent) and the TXE bit is set, the TC bit is raised. If the TCIE bit in the USART_CR1 register is set, an interrupt will be generated.

After writing the last data word to the USART_DR register, you must wait for TC=1 before disabling the USART module or setting the microcontroller to low-power mode (see the figure below for details).

Use the following software procedure to clear the TC bit:

1. Read the USART_SR register once;
2. Write to the USART_DR register once.

Note: The TC bit can also be cleared by writing '0' to it via software. *This clearing method is only recommended for use in multi-buffer communication mode.*

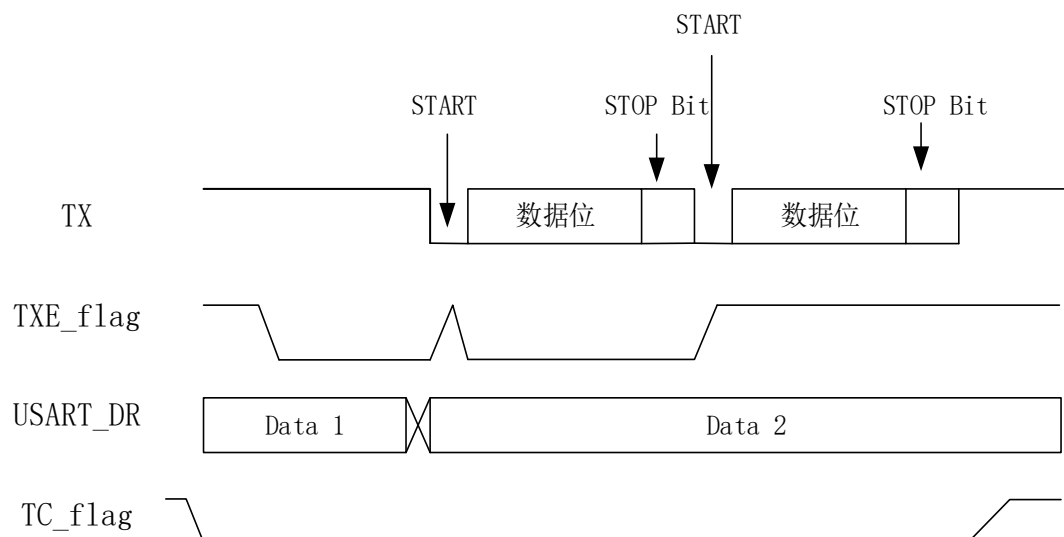


Figure 32-4 Changes in TC/TXE during USART transmission

32.3.2.4. Break symbol

Setting SBK sends a break character. The length of the break frame depends on the M bit. If SBK=1 is set, a break character will be transmitted on the TX line after the current data transmission is completed. SBK is reset by hardware when the break character transmission is complete (at the stop bit of the break character). The USART inserts a logical '1' at the end of the last break frame to ensure recognition of the start bit of the next frame.

Note: If the software resets the SBK bit before starting to transmit the break frame, the break character will not be sent. If two consecutive break frames are to be sent, the SBK bit should be set after the stop bit of the previous break symbol.

32.3.2.5. Idle symbol

Setting TE will cause the USART to send an idle frame before the first data frame.

32.3.3. Receiver

The USART can receive 8-bit or 9-bit data words based on the M bit in USART_CR1.

32.3.3.1. Start bit detection

In the USART, a start bit is considered detected if a specific sampling sequence is recognized. The sequence is: 1 1 1 0 X 0 X 0 X 0 0 0 0

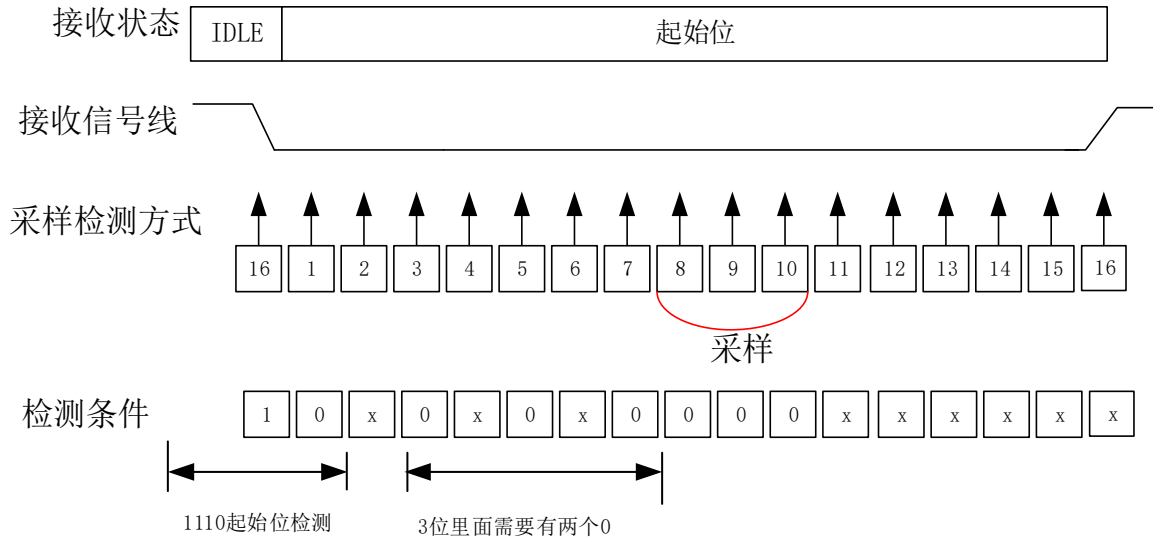


Figure 32-5 Start bit detection

If the sequence is incomplete, the receiver exits start bit detection and returns to the idle state (no flags are set) to wait for a falling edge. If all three sampling points are '0' (first sampling at bits 3, 5, 7 and second sampling at bits 8, 9, 10), the start bit is confirmed, the RXNE flag is set, and an interrupt is generated if RXNEIE=1.

If only two out of three sampling points are '0' (sampling points at bits 3, 5, 7 and bits 8, 9, 10), the start bit is still valid, but the NE noise flag is set. If this condition is not met, the start bit detection process is aborted, and the receiver returns to the idle state (no flags are set).

If only two out of three sampling points are '0' (sampling points at bits 3, 5, 7 or bits 8, 9, 10), the start bit is still valid, but the NE noise flag is set.

32.3.3.2. Character reception

During USART reception, the least significant bit of the data is shifted in first from the RX pin. In this mode, the USART_DR register contains a buffer located between the internal bus and the receive shift register.

Configuration steps:

1. Set the UE bit in the USART_CR1 register to activate the USART.
2. Program the M bit in USART_CR1 to define the word length.
3. In USART_CR2, STOP specifies the number of stop bits.
4. If multi-buffer communication is needed, select the DMA enable bit (DMAR) in USART_CR3. Configure the DMA registers as required for multi-buffer communication.
5. Use the baud rate register USART_BRR to select the desired baud rate.
6. Set the RE bit in USART_CR1. Enable the receiver to start searching for the start bit.

When a character is received:

- The RXNE bit is set. It indicates that the content of the shift register has been transferred to the RDR. In other words, the data has been received and can be read (including its associated error flags).
- An interrupt is generated if the RXNEIE bit is set.
- If a frame error, noise, or overrun error is detected during reception, the error flag will be set.
- In multi-buffer communication, RXNE is set after each byte is received and cleared by a DMA read operation on the data register.
- In single buffer mode, the RXNE bit is cleared by software reading the USART_DR register. The RXNE flag can also be cleared by writing 0 to it. The RXNE bit must be cleared before the end of the next character reception to avoid an overrun error.

Note: The RE bit should not be reset during data reception. *If the RE bit is cleared during reception, the current byte being received is lost.*

32.3.3.3. Break symbol

When a break frame is received, the USART handles it the same way as a frame error.

32.3.3.4. Idle symbol

When an idle frame is detected, it is processed the same way as a normal data frame, but an interrupt is generated if the IDLEIE bit is set.

32.3.3.5. Overrun error

If RXNE has not been reset and another character is received, an overrun error occurs. Data is transferred from the shift register to the RDR register only after the RXNE bit is cleared. The RXNE flag is set after each byte is received. An overrun error occurs if the next data has already been received or the previous DMA request has not been serviced while the RXNE flag remains set.

When an overrun error occurs: the ORE bit is set.

- RDR content will not be lost. Reading the USART_DR register will still retrieve the previous data.
- The previous contents of the shift register will be overwritten. All subsequently received data will be lost.
- An interrupt is generated if the RXNEIE bit is set or both EIE and DMAR bits are set.
- Performing sequential read operations on the USART_SR and USART_DR registers can reset the ORE bit.

Note: When the ORE bit is set, it indicates that at least 1 data has been lost. *There are two possibilities:*

- If RXNE=1, the previous valid data remains in the receive register RDR and can be read.
- If RXNE=0, it means the previous valid data has been read and there is nothing left to read in RDR. This situation may occur when new (i.e., lost) data is received while the previous valid data is being read from RDR. This situation may also occur when new data is received during the read sequence (between USART_SR register read access and USART_DR read access).

32.3.3.6. Noise error

Using oversampling techniques (except in synchronous mode), data recovery is performed by distinguishing valid input data from noise.

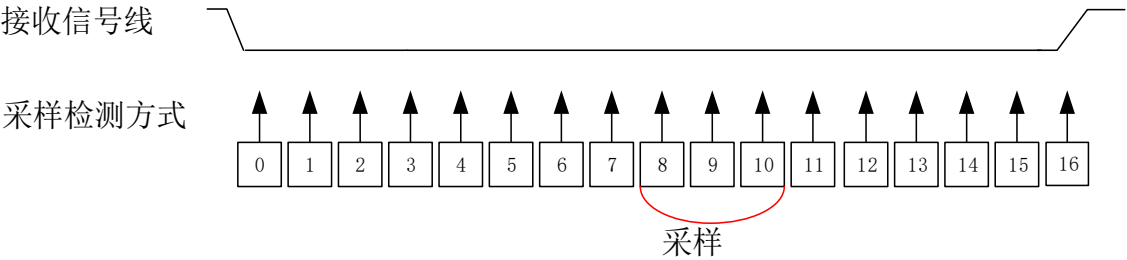


Figure 32-6 Data sampling for noise detection

Table 32-1 Data sampling for noise detection

Sampled value	NE status	Received bit value	Data validity
000	0	0	Valid
001	1	0	Invalid
010	1	0	Invalid
011	1	1	Invalid
100	1	0	Invalid
101	1	1	Invalid
110	1	1	Invalid
111	0	1	Valid

When noise is detected in the received frame:

- The NE flag is set on the rising edge of the RXNE bit.
- Invalid data is transferred from the shift register to the USART_DR register.
- In single-byte communication, no interrupt is generated. However, since the NE flag and the RXNE flag are set simultaneously, RXNE will generate an interrupt. In multi-buffer communication, an interrupt will be generated if the EIE bit in the USART_CR3 register is already set.
- Reading the USART_SR first and then the USART_DR register clears the NE flag.

32.3.3.7. Frame error

A frame error is detected when the following occurs:

Due to lack of synchronization or excessive noise, the stop bit was not received and recognized at the expected time.

When a frame error is detected:

- The FE bit is set by hardware.
- Invalid data is transferred from the shift register to the USART_DR register.
- In single-byte communication, no interrupt is generated. However, this bit is set simultaneously with the RXNE bit, and the latter will generate an interrupt. In multi-buffer communication, an interrupt will be generated if the EIE bit in the USART_CR3 register is set.
- Performing sequential read operations on the USART_SR and USART_DR registers resets the FE bit.

32.3.3.8. Configurable stop bits during reception

The number of received stop bits can be configured via the control bits in Control Register 2. In normal mode, it can be 1 or 2, while in smart card mode, it can be 0.5 or 1.5.

- 0.5 stop bits (reception in smart card mode): No sampling is performed for 0.5 stop bits. Thus, if 0.5 stop bits are selected, frame errors and break frames cannot be detected.
- 1 stop bit: Sampling for 1 stop bit is performed at the 8th, 9th, and 10th sampling points.
- 1.5 stop bits (smart card mode): When transmitting in smart card mode, the device must check whether the data has been correctly transmitted. Therefore, the receiver function block must be activated (RE=1 in the USART_CR1 register), and the signal on the data line must be sampled during the transmission of the stop bit. If a parity error occurs, the smart card will pull the data line low when the sender samples the NACK signal, i.e., at the time corresponding to the stop bit on the bus, to indicate a frame error. FE is set together with RXNE at the end of 1.5 stop bits. Sampling for 1.5 stop bits is performed at the 16th, 17th, and 18th sampling points. 1.5 stop bits can be divided into 2 parts: one is 0.5 clock cycles, during which nothing is done. Followed by 1 clock cycle of stop bit, sampled at the midpoint of this period.
- 2 stop bits: Sampling for 2 stop bits is done at the 8th, 9th, and 10th sampling points of the first stop bit. If a frame error is detected during the first stop bit, the frame error flag will be set. The second stop bit no longer checks for frame errors. The RXNE flag will be set at the end of the first stop bit.

32.3.4. Fractional baud rate generation

Fractional baud rate generation The baud rate for the receiver and transmitter should be set to the same value in the integer and fractional registers of USARTDIV.

$$\text{Tx / Rx Baud Rate} = \text{fCK} / (16 * \text{USARTDIV})$$

Here fCK is the clock for the peripheral USARTDIV is an unsigned fixed-point number. *This 12-bit value is set in the USART_BRR register.*

Note: After writing to USART_BRR, the baud rate counter will be replaced by the new value of the baud rate register. *Therefore, do not change the baud rate register value during communication.*

How to get USARTDIV from USART_BRR register value

Example 1:

If DIV_Mantissa = 27, DIV_Fraction = 12 (USART_BRR=0x1BC),

Then:

Mantissa (USARTDIV) = 27

Fraction (USARTDIV) = 12/16 = 0.75

Therefore USARTDIV = 27.75

Example 2:

Require USARTDIV = 25.62,

Therefore:

DIV_Fraction = 16*0.62 = 9.92

The closest integer is: 10 = 0x0A

$DIV_Mantissa = mantissa (25.620) = 25 = 0x19$

Thus, $USART_BRR = 0x19A$

Example 3:

Required $USARTDIV = 50.99$

Therefore:

$DIV_Fraction = 16 \times 0.99 = 15.84$

The closest integer is: $16 = 0x10 \Rightarrow DIV_frac[3:0]$ overflow \Rightarrow Carry must be added to the fractional part

$DIV_Mantissa = mantissa (50.990 + carry) = 51 = 0x33$

Thus: $USART_BRR = 0x330$, $USARTDIV=51$

Baud rate		fPCLK=36 MHz			fPCLK=72 MHz		
Se- quence num- ber	Kbps	Actual	Value placed in the baud rate reg- ister	Error (%)	Actual	Value placed in the baud rate reg- ister	Error (%)
1	2.4	2.400	937.5	0%	2.4	1875	0%
2	9.6	9.600	234.375	0%	9.6	468.75	0%
3	19.2	19.2	117.1875	0%	19.2	234.375	0%
4	57.6	57.6	39.0625	0%	57.6	78.125	0%
5	115.2	115.384	19.5	0.15%	115.2	39.0625	0%
6	230.4	230.769	9.75	0.16%	230.769	19.5	0.16%
7	460.8	461.538	4.875	0.16%	461.538	9.75	0.16%
8	921.6	923.076	2.4375	0.16%	923.076	4.875	0.16%
9	2250	2250	1	0%	2250	2	0%
10	4500	Impossible	Impossible	Impossible	4500	1	0%

Note: The lower the CPU clock frequency, the lower the error for a specific baud rate. *The achievable maximum baud rate can be derived from this set of data.*

32.3.5. USART receiver tolerance

The USART asynchronous receiver can only function properly when the overall clock system variation is within the tolerable range of the USART asynchronous receiver. Factors affecting these variations include:

- DTRA: Variation caused by transmitter error (including transmitter-side oscillator variation)
- DQUANT: Error caused by baud rate rounding at receiver side
- DREC: Variation of receiver-side oscillator
- DTCL: Variation caused by transmission line (usually due to inconsistency between transmitter's low-to-high and high-to-low transition timing).

The following condition must be met: $DTRA + DQUANT + DREC + DTCL < \text{Tolerance of USART receiver}$.

For normal data reception, the tolerance of the USART receiver equals the maximum tolerable variation, which depends on the following selections:

- 10 or 11-bit character length defined by the M bit in $USART_CR1$ register

Table 32-2 Tolerance of USART receiver when DIV_Fraction is 0

M bit	NF is an error	NF is don't care
0	3.75%	4.375%
1	3.41%	3.97%

Table 32-3 USART receiver tolerance when DIV_Fraction is non-zero

M bit	NF is an error	NF is don't care
0	3.33%	3.88%
1	3.03%	3.53%

32.3.6. USART auto-baud rate detection

The USART can detect and automatically set the value of the USARTx_BRR register based on the reception of a character. Auto-baud rate detection is useful in the following scenarios:

- The system communication rate is not confirmed in advance.
- The system is using a relatively low-precision clock source, and this mechanism allows obtaining the correct baud rate without measuring clock deviation.

The clock source frequency must be compatible with the expected communication rate. (16x oversampling must be selected, and the choice must be between $f_{CK}/65535$ and $f_{CK}/16$)

Before activating auto-baud rate detection, the auto-baud rate detection mode must be selected (via the ABRMOD[1:0] bits in the USARTx_CR3 register). There are various modes based on different character patterns.

In these auto-baud rate modes, the baud rate is measured several times during synchronous data reception, and each measurement is compared with the previous one.

These modes are:

MODE 0: Any character starting with 1. In this case, the USART measures the width of the start bit (falling edge to rising edge).

MODE 1: Any character starting with 10xx bits. In this case, the USART measures the width of the start bit and the first data bit. This measurement is performed between falling edges to ensure better accuracy in cases of slow signal rise times.

Another check for each RX transition is also performed in parallel. If the transition on RX is not sufficiently synchronized with the receiver (the receiver calculates the baud rate based on bit 0), an error will occur. Before activating auto-baud rate detection, the USARTx_BRR register must be initialized by writing a non-zero baud rate value.

Auto-baud rate detection can be activated by setting the ABREN bit in the USARTx_CR3 register. The USART will wait for the first character on RX. Setting the ABRF flag in the USARTx_ISR register indicates the completion of auto-baud rate detection. If the communication line is noisy, the correct execution of auto-baud rate detection cannot be guaranteed. In this case, the BRR value may be corrupted, and the ABRE error flag will be set. If the communication speed is incompatible with the auto-baud rate detection range (bit width is not between 16 and 65535 clock cycles @16x oversampling), an ABRE error will also occur.

The RXNE interrupt will indicate the completion of the operation. At any later time, auto-baud rate detection can be restarted by resetting the ABRF flag (by writing 0).

Note: If UE is cleared during auto-baud, the BRR value may be corrupted.

32.3.7. Multiprocessor communication

Multiprocessor communication can be achieved through USART (connecting several USARTs in a network). For example, a USART device can be the master, with its TX output connected to the RX inputs of other USART slave devices; the TX outputs of the USART slave devices are logically ANDed together and connected to the RX input of the master device.

In a multiprocessor configuration, it is generally desired that only the addressed receiver is activated to receive subsequent data, thereby reducing unnecessary USART service overhead caused by the participation of non-addressed receivers.

Non-addressed devices can enable their mute function to enter mute mode. In mute mode:

- No receive status bits will be set.
- All receive interrupts are disabled.
- The RWU bit in the USARTx_CR1 register is set to 1. The RWU can be automatically controlled by hardware or written by software under certain conditions.

Depending on the state of the WAKE bit in the USARTx_CR1 register, the USARTx can enter or exit silent mode in two ways.

- If the WAKE bit is reset: idle bus detection is performed.
- If the WAKE bit is set: address mark detection is performed.

32.3.7.1. Idle Bus Detection (WAKE=0)

When the RWU bit is written as 1, the USART enters silent mode. When an idle frame is detected, it is awakened. Then RWU is cleared by hardware, but the IDLE bit in the USART_SR register is not set. RWU can also be cleared by software. The following example illustrates wake-up and silent mode entry using idle bus detection.

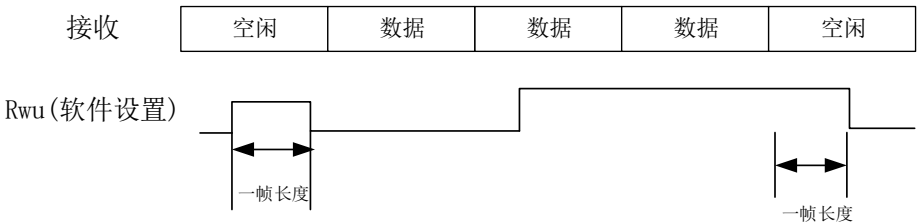


Figure 32-7 Silent Mode Using Idle Bus Detection

32.3.7.2. Address Mark Detection (WAKE=1)

In this mode, if the MSB is 1, the byte is considered an address; otherwise, it is considered data. In an address byte, the target receiver's address is placed in the 4 LSBs. This 4-bit address is compared by the receiver with its own address, which is programmed in the USART_CR2 register's ADD field.

If the received byte does not match its programmed address, the USART enters silent mode. At this point, the hardware sets the RWU bit.

Receiving this byte neither sets the RXNE flag nor generates an interrupt or DMA request because the USART is already in silent mode.

When the received byte matches the programmed address in the receiver, the USART exits silent mode. Then the RWU bit is cleared, and subsequent bytes are received normally. When this matching address byte is received, the RXNE bit will be set because the RWU bit has been cleared.

When the receive buffer contains no data (USART_SR's RXNE=0), the RWU bit can be written as 0 or 1. Otherwise, the write operation is ignored. The following example illustrates wake-up and silent mode entry using address mark detection.

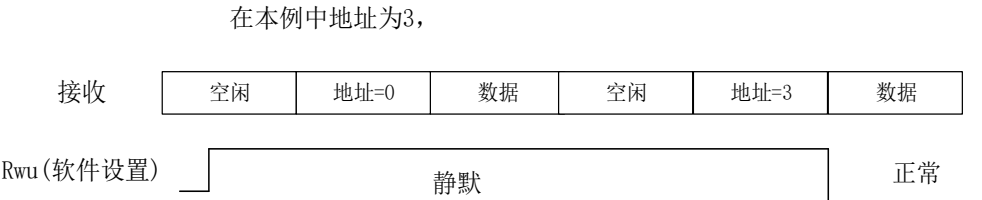


Figure 32-8 Silent Mode Using Address Mark Detection

32.3.7.3. Validation Control

Setting the PCE bit in the USART_CR1 register enables parity control (generates a parity bit during transmission and performs parity check during reception). Possible USART frame formats based on the frame length defined by the M bit are listed in the table below.

Table 32-4 Frame format

M bit	PCE bit	USART frame
0	0	SB—8 bit data—STB
0	1	SB—7 bit data—PB—STB
1	0	SB—9 bit data—STB
1	1	SB—8 bit data—PB—STB

When waking devices using address marking, address matching only considers the MSB of the data, regardless of the parity bit. *(The MSB is the last transmitted data bit, followed immediately by the parity bit or stop bit)*

32.3.7.4. Even parity

The parity bit makes the number of '1's in the 7 or 8 LSB data bits plus the parity bit even within a frame. For example: Data=00110101, which has 4 '1's. If even parity is selected (PS=0 in USARTx_CR1), the parity bit will be '0'.

32.3.7.5. Odd parity

This parity bit makes the number of '1's in the 7 or 8 LSB data bits plus the parity bit odd within a frame. For example: data=00110101, with 4 '1's. If odd parity is selected (PS=1 in USARTx_CR1), the parity bit will be '1'.

32.3.7.6. Transmission mode

If the PCE bit in USARTx_CR1 is set, the MSB of the data written to the data register is replaced with a parity bit before transmission (even parity for an even number of '1's, odd parity for an odd number of '1's). If parity check fails, the PE flag in the USART_SR register is set to '1', and if USART_CR1's PEIE is pre-configured, an interrupt is generated.

32.3.8. LIN (Local Interconnect Network) mode

Configure USART_CR2.LINEN=1 to select LIN mode. In LIN mode, the following bits must remain '0':

- USART_CR2 register's CLKEN;
- USART_CR3 register's STOP/SCEN/HDSEL/IREN.

32.3.8.1. Transmit

Compared to general USART transmission, LIN transmission has the following differences:

M=0, data length is 8 bits;

USART_CR2.LINEN=1 must be configured. Setting SBK will send a 13-bit '0' as the break frame. Then, a '1' bit is sent to allow detection of the next start bit.

32.3.8.2. Receive

When LIN mode is enabled, the break character detection circuit is activated. This detection is completely independent of the USART receiver. The break character can be detected as soon as it appears, whether during bus idle or when a data frame is being transmitted but not yet completed and a break character is inserted.

When the receiver is enabled (USART_CR1's RE=1), the circuit monitors the start signal on RX. The method for monitoring the start bit is the same as detecting a break character or data. After the start bit is detected, the circuit samples each subsequent bit at the 8th, 9th, and 10th oversampling clock points. If 10 consecutive bits (when USART_CR2's LBDL = 0) or 11 consecutive bits (when USART_CR2's LBDL = 1) are all '0' and followed by a delimiter, the LBD flag in USART_SR is set. If the LBDIE bit = 1, an interrupt is generated. Before confirming the break character, check the delimiter as it indicates the RX line has returned to a high level.

If a '1' is sampled before the 10th or 11th sampling point, the detection circuit cancels the current detection and restarts the search for a start bit. If LIN mode is disabled, the receiver continues to operate as a normal USART without considering break symbol detection.

If LIN mode is not activated (LINEN=0), the receiver continues to operate normally in USART mode without performing break detection.

If LIN mode is activated (LINEN=1), as soon as a frame error occurs (i.e., a '0' is detected at the stop bit, which happens in a break frame), the receiver stops until the break symbol detection circuit receives a 1 (this occurs when the break symbol is not fully transmitted) or a delimiter (this occurs when a complete break symbol has been detected).

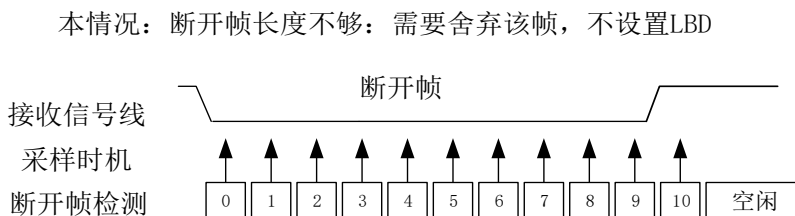


Figure 32-9 Case where the break frame length is insufficient in LIN mode

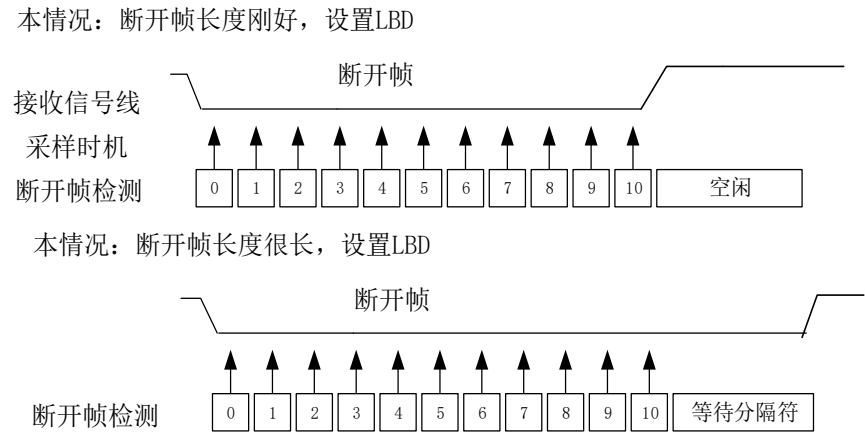


Figure 32-10 Case where the break frame length is sufficient in LIN mode

断开发生在空闲后

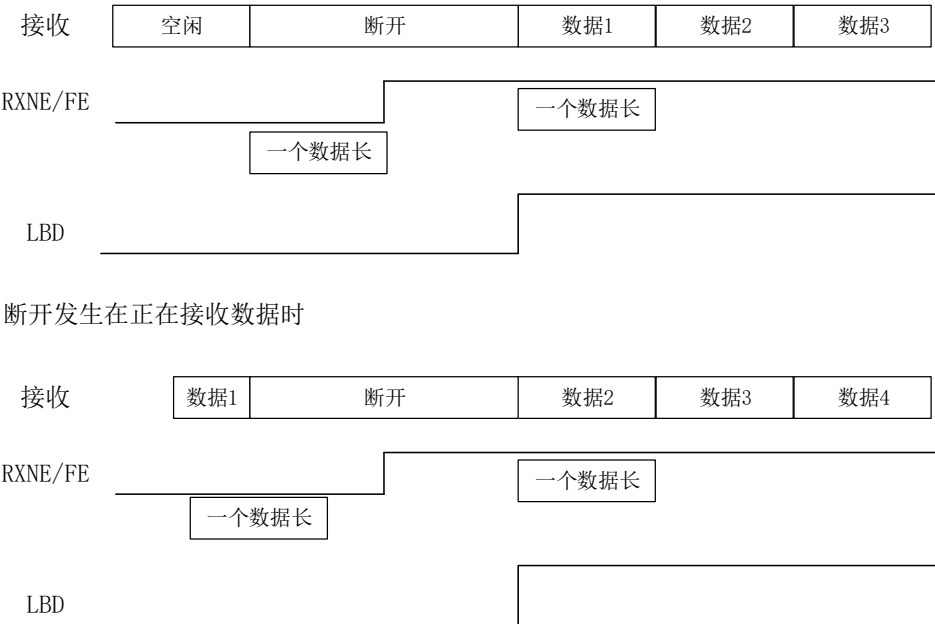


Figure 32-11 Break detection and frame error detection in LIN mode

32.3.9. USART Synchronous Mode

Synchronous mode is selected by writing the CLKEN bit in the USART_CR2 register to 1. In synchronous mode, the following bits must remain cleared:

The LINEN bit in the USART_CR2 register

The SCEN, HDSEL, and IREN bits in the USART_CR3 register

The USART allows the user to control bidirectional synchronous serial communication in master mode.

The CK pin is the output of the USART transmitter clock. No clock pulses are present on the CK pin during the start and stop bits. Depending on the state of the LBCL bit in the USART_CR2 register, a clock pulse is either generated or not generated during the last valid data bit. The CPOL bit in the USART_CR2 register allows the user to select the clock polarity, and the CPHA bit in the USART_CR2 register allows the user to select the phase of the external clock.

During bus idle periods, before actual data arrives, and when sending break symbols, the external CK clock is not activated.

In synchronous mode, the USART transmitter operates exactly the same as in asynchronous mode. However, since CK is synchronized with TX (according to CPOL and CPHA), the data on TX is transmitted synchronously with CK.

The operation of the USART receiver in synchronous mode differs from that in asynchronous mode. If RE=1, data is sampled on CK (determined by CPOL and CPHA to be on the rising or falling edge), without requiring any oversampling. However, setup time and duration (depending on the baud rate, 1/16 bit time) must be considered.

Note:

The CK pin works in conjunction with the TX pin. Thus, the clock is only provided when the transmitter is enabled (TE=1) and data is being transmitted (data is written to the USART_DR register). This means it is impossible to receive synchronous data when no data is being transmitted.

The correct configuration of LBCL, CPOL, and CPHA bits should be done when both the transmitter and receiver are disabled; these bits cannot be changed when the transmitter or receiver is enabled.

It is recommended to set TE and RE in the same instruction to reduce the receiver's setup and hold times.

The USART only supports master mode: it cannot receive or transmit data using an input clock from another device (CK is always an output).

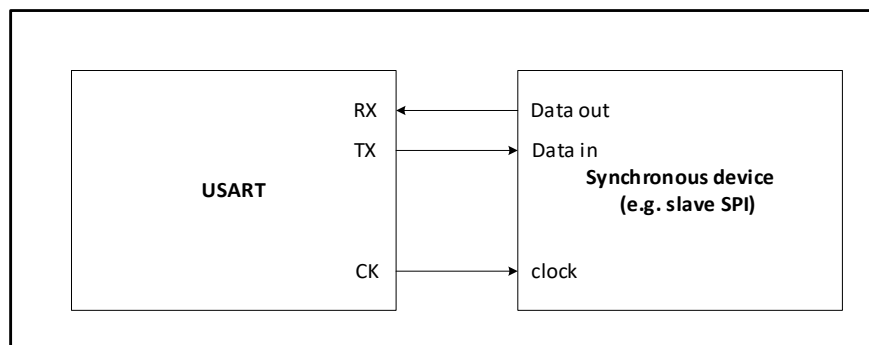


Figure 32-12 Example of USART synchronous transmission

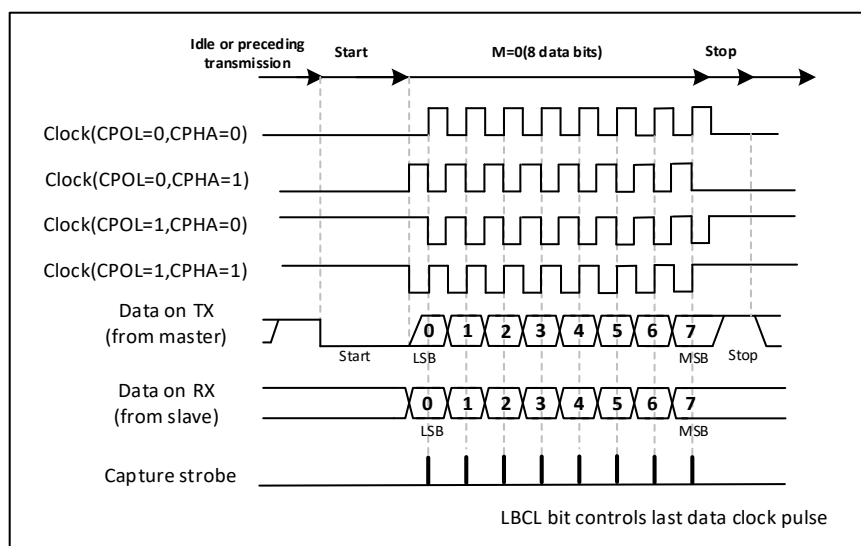


Figure 32-13 Example of USART data clock timing (M=0)

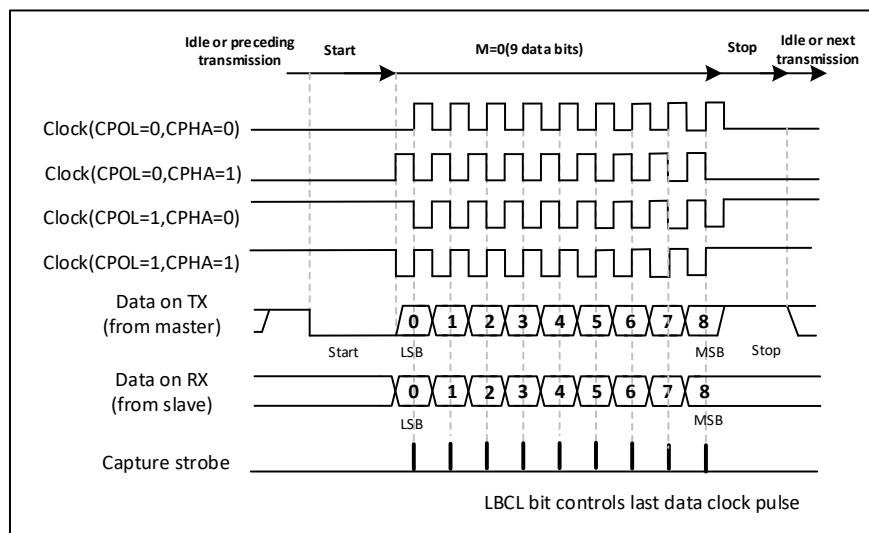


Figure 32-14 Example of USART data clock timing (M=1)

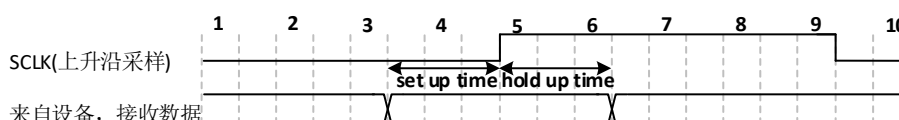


Figure 32-15 RX data sampling/hold time

32.3.10. Single-wire half-duplex communication

Single-wire half-duplex mode is selected by setting the HDSEL bit in the USARTx_CR3 register. In this mode, the following bits must remain cleared:

- CLKEN bit of the USARTx_CR2 register
- SCEN and IREN bits of the USART_CR3 register

The USART can be configured to follow the single-wire half-duplex protocol. In single-wire half-duplex mode, the TX and RX pins are internally interconnected on the chip. The control bit "HALF DUPLEX SEL" (HDSEL bit in USARTx_CR3) is used to select between half-duplex and full-duplex communication.

When HDSEL is '1'

- RX is no longer used
- When there is no data transmission, TX is always released. Therefore, it behaves as a standard I/O port in idle or receive states. This means the I/O must be configured as a floating input (or open-drain output high) when not driven by the USART.

Otherwise, communication is similar to normal USART mode. Line conflicts are managed by software (e.g., by using a central arbiter). In particular, transmission is never blocked by hardware. When the TE bit is set, transmission continues as soon as data is written to the data register.

32.3.11. Smart Card

Set the SCEN bit in the USART_CR3 register to select smart card mode. In smart card mode, the following bits must remain cleared:

- The LINEN bit in the USART_CR2 register.
- The HDSEL and IREN bits in the USART_CR3 register.

Additionally, the CLKEN bit can be set to provide a clock to the smart card.

This interface complies with the ISO7816-3 standard and supports the smart card asynchronous protocol.

The USART should be configured as follows:

1. 8 data bits plus parity bit: in this case, M=1 and PCE=1 in the USART_CR1 register.
2. 1.5 stop bits for both transmission and reception: i.e., STOP=11 in the USART_CR2 register.

Note: It is also possible to select 0.5 stop bits during reception, but to avoid switching between two configurations, it is recommended to use 1.5 stop bits for both transmission and reception.

The example below illustrates the signals on the data line in both cases: with and without a parity error.

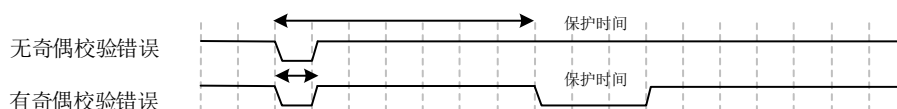


Figure 32-16 ISO7816-3 Asynchronous Protocol

When connected to a smart card, the USART's TX drives a bidirectional line that is also driven by the smart card. To achieve this, SW_RX must be connected to the same I/O port as TX. During the transmission of the start bit and data bytes, the transmitter's output enable bit TX_EN is set, and it is released (weak pull-up) during the stop bit transmission, allowing the receiver to pull the data line low in case of a parity error. If TX_EN is not used, TX is pulled high during the stop bit: in this case, as long as TX is configured as open-drain, the receiver can also drive this line.

The smart card is a single-wire half-duplex communication protocol.

- 1) Data transmission from the transmit shift register must be delayed by at least 1/2 baud clock. During normal operation, a full transmit shift register will start shifting data out on the next baud clock edge. In smart card mode, this transmission is delayed by 1/2 baud clock.
- 2) If a parity error is detected while receiving a data frame configured for 0.5 or 1.5 stop bits, after completing the reception of the frame (i.e., at the end of the stop bit), the transmit line is pulled low for one baud clock cycle. This informs the smart card that the data sent to the USART was not correctly

received. This NACK signal (pulling the transmit line low for one baud clock cycle) will generate a frame error at the transmitter side (the transmitter is configured for 1.5 stop bits). The application can handle retransmitting data according to the protocol. If the NACK control bit is set, the receiver will issue a NACK signal upon a parity error; otherwise, no NACK will be sent.

- 3) The setting of the TC flag can be delayed by programming the guard time register. During normal operation, TC is set when the transmit shift register becomes empty and no new transmit request occurs. In smart card mode, an empty transmit shift register will trigger the guard time counter to start counting up until the value in the guard time register is reached. TC is forced low during this period. When the guard time counter reaches the value in the guard time register, TC is set high.
- 4) The clearing of the TC flag is not affected by smart card mode.
- 5) If the transmitter detects a frame error (receiving a NACK signal from the receiver), the transmitter's receiver function module will not detect the NACK as a start bit. According to the ISO protocol, the duration of the received NACK can be 1 or 2 baud clock cycles.
- 6) On the receiver side, if a parity error is detected and a NACK is sent, the receiver will not detect the NACK as a start bit.

Note:

- 1) Break characters have no meaning in smart card mode. A 00h data with a frame error will be treated as data rather than a break character.
- 2) When toggling the TE bit back and forth, no IDLE frame is transmitted. The ISO protocol does not define IDLE frames.

The following figure details how the USART samples NACK signals. In this example, the USART is transmitting data and is configured for 1.5 stop bits. To check data integrity and NACK signals, the USART receiver function block is activated.

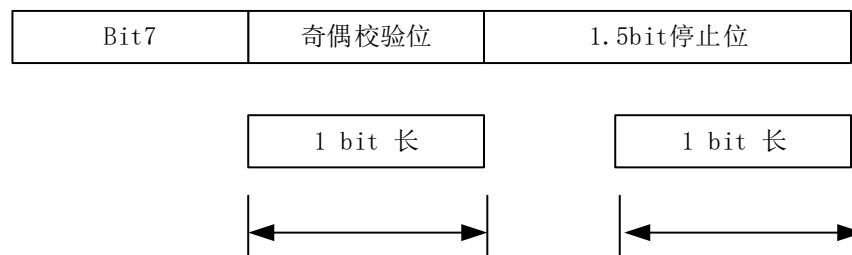


Figure 32-17 Detecting Parity Errors Using 1.5 Stop Bits

The USART can provide a clock to the smartcard via the CK output. In smartcard mode, CK is not directly related to communication but is simply driven by the internal peripheral input clock through a 5-bit prescaler to provide the smartcard clock. The division factor is configured in the prescaler register USART_GTPR. The CK frequency can range from $f_{CK}/2$ to $f_{CK}/62$, where f_{CK} is the peripheral input clock.

32.3.12. IrDA SIR ENDEC Functional Block

IrDA mode is selected by setting the IREN bit in the USART_CR3 register. In IRDA mode, the following bits must remain cleared:

- LINEN, STOP, and CLKEN bits in the USART_CR2 register.
- SCEN and HDSEL bits in the USART_CR3 register.

32.3.12.1. IrDA Normal Mode

The IrDA SIR physical layer specifies the use of an inverted return-to-zero modulation scheme (RZI), where an infrared light pulse represents logic '0'. The SIR transmit encoder modulates the NRZ (Non-Return-to-Zero) bitstream output from the USART. The output pulse stream is transmitted to an external output driver and infrared LED. The USART supports SIR ENDEC at rates up to 115.2Kbps. In normal mode, the pulse width is defined as 3/16 of a bit period.

The SIR receive decoder demodulates the return-to-zero bitstream from the infrared receiver and outputs the received NRZ serial bitstream to the USART. In the idle state, the decoder input is typically high (marking state). The polarity of the encoder output is opposite 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 will be ignored by the IrDA decoder. If the receiver is busy (i.e., the USART is receiving decoded data from the IrDA decoder), data from the USART to the IrDA TX will not be encoded by IrDA. When receiving data, transmission should be avoided as the data to be sent may be corrupted.
- The SIR transmission logic sends '0' as a high pulse and '1' as a low level. The pulse width is defined as 3/16 of the bit period in normal mode.
- The SIR receive logic interprets a high-level state as '1' and a low pulse as '0'.
- The transmitter encoder output has the opposite polarity of the decoder input. When idle, the SIR output remains in a low state.
- The SIR decoder converts IrDA-compatible received signals into a bit stream for the USART.
- The IrDA specification requires pulses to be wider than 1.41 μ s. The pulse width is programmable. The spike detection logic on the receiver side filters out pulses with widths less than 2 PSC cycles (PSC is the prescaler value programmed in the IrDA low-power baud rate register USART_GTPR). Pulses with widths less than 1 PSC cycle must be filtered out, but those with widths greater than 1 but less than 2 PSC cycles may be received or filtered, while those with widths greater than 2 cycles will be considered valid pulses. When PSC=0, the IrDA encoder/decoder does not work.
- The receiver can communicate with a low-power transmitter.
- In IrDA mode, the STOP bit in the USART_CR2 register must be configured to 1 stop bit.

32.3.12.2. IrDA Low-Power Mode

Transmitter:

In low-power mode, the pulse width no longer lasts for 3/16 of the bit period. Instead, the pulse width is 3 times the low-power baud rate, which can be as low as 1.42 MHz. Typically, this value is 1.8432 MHz ($1.42 \text{ MHz} < \text{PSC} < 2.12 \text{ MHz}$). A low-power mode programmable divider divides the system clock to achieve this value.

Receiver:

Reception in low-power mode is similar to reception in normal mode. To filter out spike interference pulses, the USART should filter out pulses shorter than 1 PSC. Only low-level signals with durations longer than 2 cycles of the IrDA low-power baud rate clock (PSC in USART_GTPR) are accepted as valid signals.

Note:

- 1. Pulses with widths less than 2 but greater than 1 PSC cycle may or may not be filtered out.
- 2. The receiver setup time should be managed by software. The IrDA physical layer specification requires a minimum delay of 10ms between transmission and reception (IrDA is a half-duplex protocol).

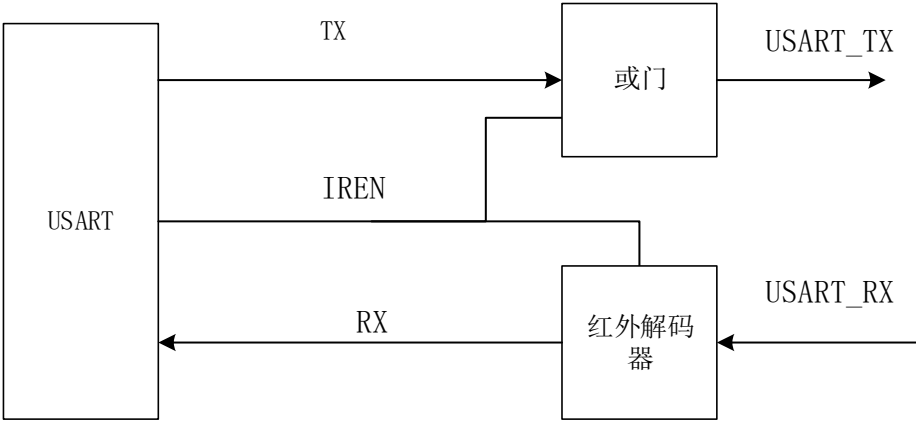


Figure 32-18 IrDA SIR ENDEC block diagram

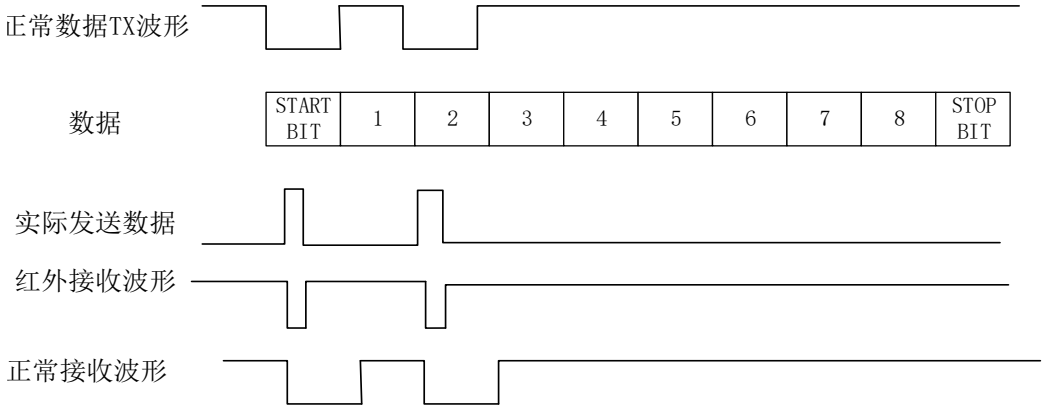


Figure 32-19 IrDA data modulation (3/16) - Normal mode

32.3.13. Continuous communication using DMA

USART can utilize DMA for continuous communication. DMA requests for the Rx buffer and Tx buffer are generated separately.

32.3.13.1. Using DMA for transmission

DMA transmission can be activated by setting the DMAT bit in the USART_CR3 register. When the TXE bit is set to '1', the DMA transfers data from the specified SRAM area to the USART_DR register. The steps to allocate a DMA channel for USART transmission are as follows:

- 1) Configure the USART_DR register address as the destination address for DMA transfer in the DMA control register. After each TXE event, data will be transferred to this address.

2) Configure the memory address as the source address for DMA transfer in the DMA control register. After each TXE event, data will be read from this memory area and transferred to the USART_DR register.

3) Configure the total number of bytes to be transmitted in the DMA control register.

4) Configure the channel priority in the DMA register.

5) Configure the DMA interrupt to be generated either at half or full completion of the transfer, according to the application requirements.

6) Clear the TC bit by writing 0.

7) Activate the channel in the DMA registers.

When the DMA controller completes transferring the specified amount of data, it generates an interrupt on the DMA channel's interrupt vector.

In transmission mode, when the DMA completes transferring all the data to be sent, the DMA controller sets the TCIF flag in the DMA_ISR register. Monitoring the TC flag in the USARTx_SR register can confirm whether the USART communication has ended, thus avoiding corruption of the last transmitted data before disabling the USART or entering stop mode. The software must first wait for TXE=1, then wait for TC=1.

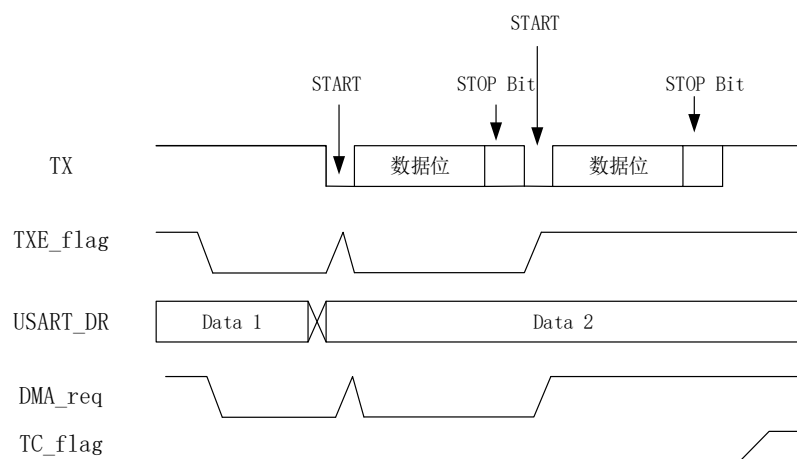


Figure 32-20 Using DMA for transmission

32.3.13.2. Using DMA for reception

DMA reception can be activated by setting the DMAR bit in the USART_CR3 register. Each time a byte is received, the DMA controller transfers the data from the USART_DR register to the specified SRAM area (refer to the DMA-related instructions). The steps to allocate a DMA channel for USART reception are as follows:

1) Configure the USART_DR register address as the source address for the transfer via the DMA control register. After each RXNE event, data will be read from this address and transferred to memory.

2) Configure the memory address as the destination address for the transfer via the DMA control register. After each RXNE event, data will be transferred from USART_DR to this memory area.

3) Configure the total number of bytes to be transmitted in the DMA control register.

4) Configure the channel priority in the DMA register.

5) Configure the DMA interrupt to be generated either at half or full completion of the transfer, as required by the application.

6) Activate the channel in the DMA control register.

When the DMA controller completes the specified transfer amount, it generates an interrupt on the DMA channel's interrupt vector.

32.3.13.3. Error flags and interrupt generation in multi-buffer communication

In the case of multi-buffer communication, if any error occurs during communication, the error flag will be set after the current byte transmission. If the interrupt enable bit is set, an interrupt will be generated. In the case of single-byte reception, frame error, overrun error, and noise flags set along with RXNE have separate error flag interrupt enable bits; if set, an interrupt will be generated after the current byte transmission ends.

32.3.14. Hardware flow control

The nCTS input and nRTS output can be used to control the serial data flow between two devices. The figure below shows how to connect two devices in this mode.

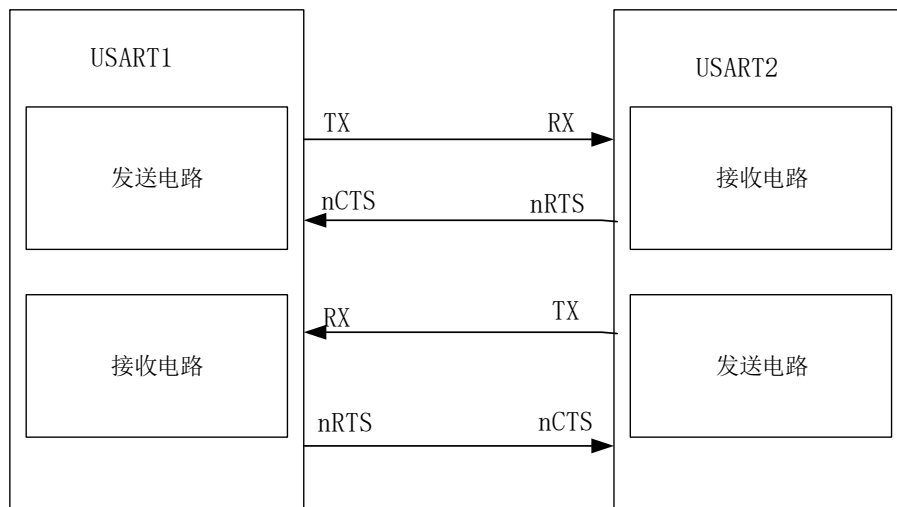


Figure 32-21 Hardware flow control between two USARTs

32.3.14.1. RTS flow control

If RTS flow control is enabled (RTSE=1), nRTS becomes active (pulled low) whenever the USART receiver is ready to receive new data. When data arrives in the receive register, nRTS is released, indicating the desire to stop data transmission at the end of the current frame.

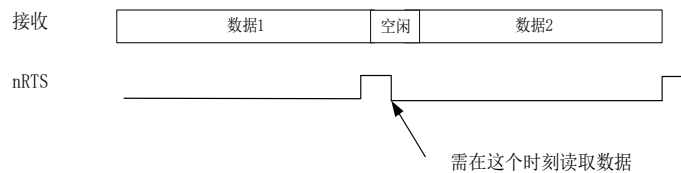


Figure 32-22 RTS flow control

32.3.14.2. CTS flow control

If CTS flow control is enabled (CTSE=1), the transmitter checks the nCTS input before sending the next frame. If nCTS is active (pulled low), the next data is transmitted (assuming the data is ready to send, i.e., TXE=0); otherwise, the next frame of data is not sent out. If nCTS becomes invalid during transmission, the current transmission stops after completion.

When CTSE=1, the hardware automatically sets the CTSIF status bit whenever the nCTS input changes state. It indicates whether the receiver is ready to communicate. An interrupt is generated if the CTSIE bit in the USART_CT3 register is set.

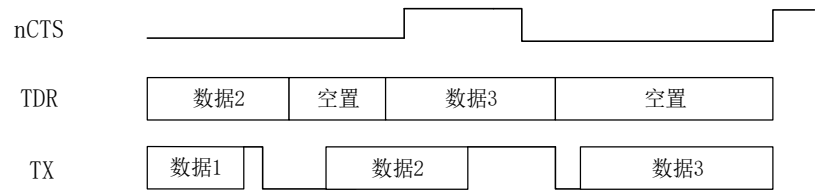


Figure 32-23 CTS flow control

32.4. USART interrupt request

Se- quence num- ber	Interrupt event	Event flag	Enable bit	Trans- mit/Receive
1	Transmit data register empty	TXE	TXEIE	Transmit
2	CTS (Clear to Send) interrupt	CTSIF	CTSIE	Transmit
3	Transmission complete	TC	TCIE	Transmit
4	Receive register not empty (read data ready)	RXNE	RXNEIE	Receive
5	Overrun error	ORE		Receive
6	Idle frame	IDLE	IDLEIE	Receive
7	Parity error	PE	PEIE	Receive
8	Break flag	LBD	LBDIE	Receive
9	Noise, overrun, and framing errors during multi-processor communication	NE/ORE/FE	EIE	Receive

Note: All USART interrupts share the same interrupt vector.

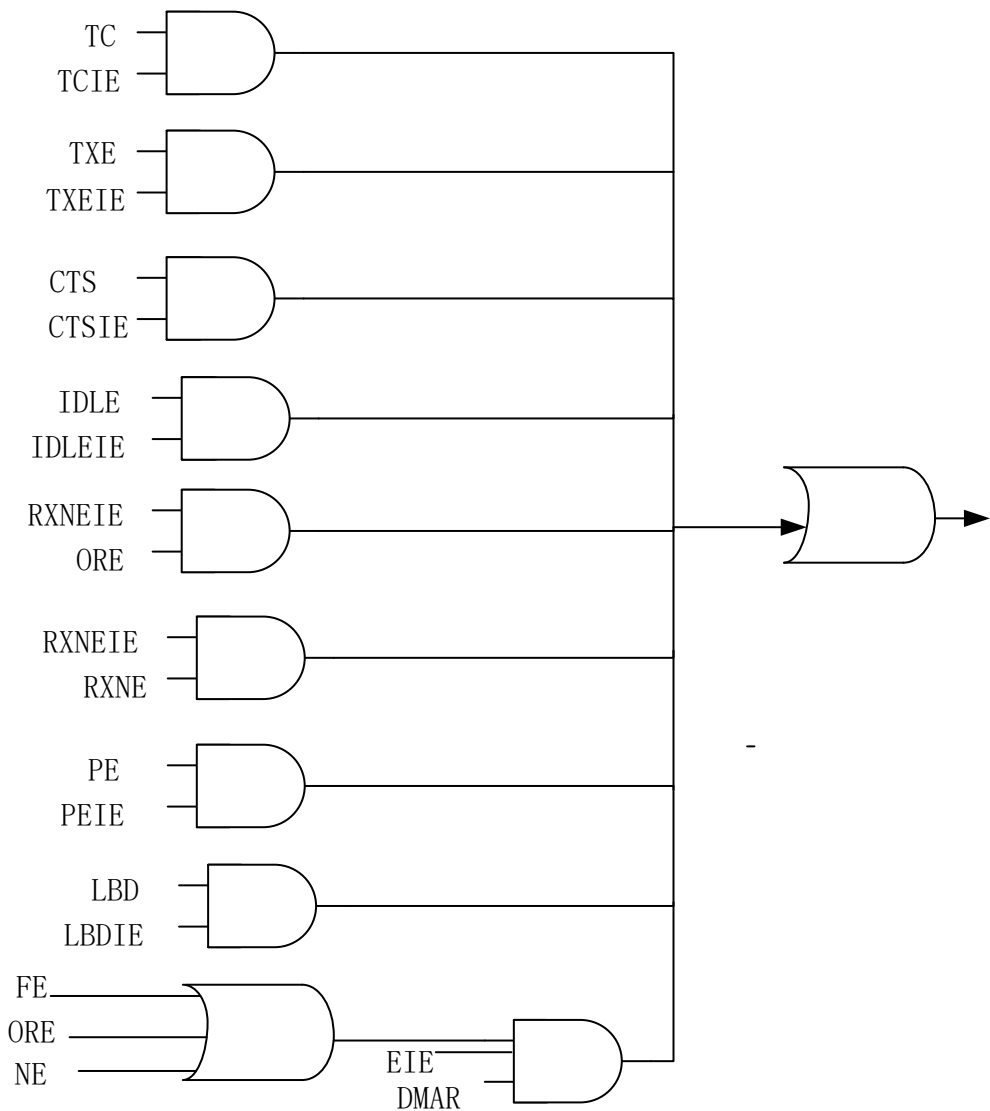


Figure 32-24 USART Interrupt Mapping Diagram

32.5. USART Registers

32.5.1. Status Register (USART_SR)

Address offset: 0x00

Reset value: 0x0000 00C0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res			ABRRQ	ABRE	ABRF	CTS	LBD	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
-			W	R		RC_W0		R	RC_W0		R				

Bit	Name	R/W	Reset Value	Function
31: 13	Reserved	-	-	Reserved

Bit	Name	R/W	Reset Value	Function
12	ABRRQ	W	0	Auto baud rate request. Writing 1 to this bit resets the ABRF flag and requests auto baud rate detection for the next frame.
11	ABRE	R	0	Auto baud rate error flag. This register is set by hardware when an auto baud rate detection error occurs (baud rate out of range or character comparison error). The software clears this bit by writing 1 to the ABRRQ register.
10	ABRF	R	0	Auto baud rate detection flag. This bit is set by hardware when auto baud rate setting is enabled (with RXNE=1, generating an interrupt if interrupts are enabled), or when an error occurs during auto baud rate detection (ABRE=1, RXNE=1, FE=1). The software clears this bit by writing 1 to the ABRRQ bit in the USART_RQR register.
9	CTS	RC_W0	0	CTS flag. This bit is set to 1 when CTS input toggles and CTSE=1. Cleared by writing 0. A CTS interrupt is generated when CTSIE=1. 0: CTS line value unchanged 1: CTS line value changed
8	LBD	RC_W0	0	LBD: LIN break detection flag This bit is set by hardware when a LIN break is detected and cleared by software (writing 0 to this bit). An interrupt is generated if LBDIE=1 in USART_CR3. 0: No LIN break detected; 1: LIN break detected. Note: If LBDIE=1, an interrupt is generated when LBD is '1'
7	TXE	R	1	Transmit data register empty flag. This bit is set by hardware when data from USART_DR register is transferred to the shift register. An interrupt is generated when TXEIE=1. Writing to USART_DR register clears this bit.

Bit	Name	R/W	Reset Value	Function
				0: Data not transferred to shift register 1: Data transferred to shift register
6	TC	RC_W0	1	Transmission complete flag. This register is set by hardware when the transmission of the data frame is complete and TXE=1. An interrupt is generated when TCIE=1. This bit is cleared by software reading the USART_SR register and then writing the USART_DR register (for multiprocessor communication). Software can also clear it by writing 0. 0: Transmission not complete 1: Transmission complete
5	RXNE	RC_W0	0	Read data register not empty flag. This register is set by hardware when the shift register value is transferred to the USART_DR register. This bit is cleared by software reading the USART_DR register. An interrupt is generated when RXNEIE=1. 0: No data received 1: Receive data ready
4	IDLE	R	0	Idle flag. The hardware sets this register when an IDLE line is detected. An interrupt is generated when IDLEIE=1. The software can clear this bit by first reading the USART_SR register and then reading the USART_DR register. 0: No IDLE line detected 1: IDLE line detected
3	ORE	R	0	Overrun error flag. When RXNE=1, the hardware sets this bit when the data received in the shift register is ready to be transferred to the RDR register. The software can clear this bit by first reading the USART_SR register and then reading the USART_DR register. An interrupt is generated when RXNEIE=1. 0: No Overrun error occurred

Bit	Name	R/W	Reset Value	Function
				<p>1: Overrun error occurred</p> <p>Note: When this register is set, the content of the RDR register is not lost, but the shift register content is overwritten.</p> <p>An ORE interrupt is generated when EIE=1.</p>
2	NE	R	0	<p>Noise error flag.</p> <p>The hardware sets this register when noise is detected in the data frame.</p> <p>The software can clear this bit by first reading the USART_SR register and then reading the USART_DR register.</p> <p>0: No noise error detected</p> <p>1: Noise error detected</p> <p>Note: When RXNE and NE occur simultaneously, no interrupt is generated when NE=1, but an interrupt is generated when the RXNE flag is set. In multi-buffer communication mode, an interrupt is generated when NE=1 while EIE=1.</p>
1	FE	R	0	<p>Frame error flag.</p> <p>This bit is set by hardware when desynchronization, excessive noise, or a break character is detected.</p> <p>The software can clear this bit by first reading the USART_SR register and then reading the USART_DR register.</p> <p>0: No frame error detected</p> <p>1: Frame error or break character detected</p> <p>Note: When RXNE and FE occur simultaneously, no interrupt is generated when FE=1, but an interrupt is generated when the RXNE flag is set. If the current data transmission generates both a frame error and an overrun error, the hardware will still continue the transmission and only set the ORE flag. In multi-buffer communication mode, an interrupt is generated when FE=1 while EIE=1.</p>
0	PE	R	0	<p>Parity error.</p> <p>The hardware sets this register when a parity error occurs during reception.</p>

Bit	Name	R/W	Reset Value	Function
				<p>The software can clear this bit by first reading the USART_SR register and then reading the USART_DR register. However, the software must wait for RXNE=1 before clearing this bit. An interrupt is generated when PEIE occurs.</p> <p>0: No parity error generated 1: Parity error generated</p>

32.5.2. Data Register (USART_DR)

Address offset: 0x04

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 9	Reserved	-	-	Reserved
8: 0	DR[8:0]	RW	0x0	<p>Receive/Transmit Data Register.</p> <p>Depending on whether it is a read or write operation, the former represents received data, and the latter represents transmitted data.</p> <p>The DR register physically consists of two registers (one for transmit TDR and one for receive RDR), enabling the DR register to perform both read and write functions.</p> <p>The TDR register provides a parallel interface between the internal bus and the output shift register, while the RDR register provides a parallel interface between the input shift register and the internal bus.</p> <p>When parity is enabled during transmit operations, writing to the MSB bit (bit7 or bit8) is invalid as it is replaced by the parity bit.</p> <p>When parity is enabled during receive operations, the read MSB bit is the received parity bit.</p>

32.5.3. Baud rate register (USART_BRR)

Address offset: 0x08

Reset value: 0x0000 0000

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

In auto baud rate detection mode, the hardware updates this register.

Bit	Name	R/W	Reset Value	Function
31: 16	Reserved	-	-	Reserved
15: 4	DIV_Mantissa[11:0]	RW	0	12-bit integer
3: 0	DIV_Fraction[3:0]	RW	0	4-bit fraction

32.5.4. Control register 1 (USART_CR1)

Address offset: 0x0C

Reset value: 0x0000_0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	UE	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	RWU	SBK
-	-	RW													

Bit	Name	R/W	Reset Value	Function
31: 14	Reserved	-	-	Reserved
13	UE	RW	0	<p>USART enable. When this bit is cleared, the USART module will immediately stop current operations. This bit is set and cleared by software.</p> <p>0: USART prescaler and output disabled, low-power mode</p> <p>1: USART enabled</p> <p>The software must wait for USART_SR.TC to be set before clearing the UE bit and entering low-power mode;</p>

Bit	Name	R/W	Reset Value	Function
				Meanwhile, the DMA channel needs to be disabled before clearing the UE bit.
12	M	RW	0	0: 1-bit start bit, 8-bit data bits, n-bit stop bits 1: 1-bit start bit, 9-bit data bits, n-bit stop bits
11	WAKE	RW	0	Receiver wake-up method. Wake-up method from mute mode. Set or cleared by software. 0: Idle line wake-up 1: Address wake-up
10	PCE	RW	0	Parity control. 0: Parity disable 1: Parity enable Parity bit: the 9th bit for 9-bit; the 8th bit for 8-bit.
9	PS	RW	0	Parity selection. Set and cleared by software. 0: Even parity 1: Odd parity
8	PEIE	RW	0	PE interrupt enabled. Set and cleared by software. 0: Disabled 1: PE interrupt enabled
7	TXEIE	RW	0	TXE interrupt enabled. Set and cleared by software. 0: Disabled 1: TXE interrupt enabled
6	TCIE	RW	0	Transmission Complete Interrupt Enable. Set and cleared by software. 0: Disabled 1: TC interrupt enable
5	RXNEIE	RW	0	RXNE interrupt enable; Set and cleared by software. 0: Disabled 1: ORE or RXNE interrupt enable
4	IDLEIE	RW	0	IDLE interrupt enable. Set and cleared by software. 0: Disabled 1: IDLE interrupt enable
3	TE	RW	0	Transmission enable. 0: Transmission disable 1: Transmission enable

Bit	Name	R/W	Reset Value	Function
2	RE	RW	0	Receiver enable. 0: Receiver disabled 1: Receiver enabled, start bit detection begins
1	RWU	RW	0	Receive wake-up. This bit indicates whether the USART is in mute mode. This register is set when the mute mode sequence is received; it is cleared when the wake-up sequence is received. The specific wake-up sequence (address or IDLE) is controlled by the USART_CR1.WAKE bit. 0: Receiver is in active mode 1: Receiver is in mute mode Note 1: Before setting this bit to enter mute mode, the USART must have already received a data byte; otherwise, it cannot be woken up by idle bus detection in mute mode. Note 2: When configured for address-mark detection wake-up (WAKE=1), the RWU bit cannot be modified by software when RXNE is set.
0	SBK	RW	0	Send break frame. Software sets this register to send a break byte. The hardware clears this register after sending the stop bit of the Break frame. 0: Do not send break byte 1: Send break character

32.5.5. Control register 2 (USART_CR2)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	LINEN	STOP[1:0]]		CLKEN	CPOL	CPHA	LBCL	Res	LBDIE	LBDL	Res	ADD[3:0]			
-	RW							-	RW	RW	-	RW			

Bit	Name	R/W	Reset Value	Function
31: 15	Reserved	-	-	Reserved
14	LINEN	RW	0	<p>LIN mode enabled. Software set and clear.</p> <p>0: LIN mode; 1: Enable LIN mode;</p> <p>In LIN mode, send LIN sync breaks by enabling SBK bit.</p> <p>Breaks (13 lower bits), and detects LIN sync break character.</p>
13: 12	STOP[1:0]	RW	0	<p>Stop bit configuration.</p> <p>00: 1 stop bit; 01: 0.5stop; 10: 2 stop bits; 11: 1.5stop</p>
11	CLKEN	RW	0	<p>CK pin enabled.</p> <p>0: Disabled; 1: CK pin enabled;</p> <p>When synchronous mode is not supported, this bit is reserved.</p>
10	CPOL	RW	0	<p>Clock polarity.</p> <p>Synchronous mode, CK pin output clock polarity.</p> <p>0: Outside the transmission window, the CK pin maintains a stable low value; 1: Outside the transmission window, the CK pin maintains a stable high value;</p>
9	CPHA	RW	0	<p>This bit is used in synchronous mode to select the phase of the clock output on the CK pin. It works with the CPOL bit to generate the desired clock/data relationship.</p> <p>0: The first clock transition is the first data capture edge; 1: The second clock transition is the first data capture edge;</p>
8	LBCL	RW	0	<p>Determines whether the clock pulse of the last data bit is output on the CK pin.</p> <p>0: The clock pulse of the last data bit is not output on the CK pin; 1: The clock pulse of the last data bit is output on the CK pin;</p>

Bit	Name	R/W	Reset Value	Function
7	Reserved	-	-	Reserved
6	LBDIE	RW	0	LIN break interrupt enable. 0: Disabled; 1: Interrupt generated; Controls the LBD in the USART_SR register, enabling Sets LBD to 1 and generates an interrupt.
5	LBDL	RW	0	LIN break detection length. 0: 10-bit break detection; 1: 11-bit break detection;
4	Reserved	-	-	Reserved
3: 0	ADD[3:0]	RW	4'b0	USART address. This register is used for multiprocessor mute mode and serves as the address for 4-bit address wake-up.

32.5.6. Control Register 3 (USART_CR3)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Re	ABR-		ABR	OVE	CTSI	CTS	RTS	DMA	DMA	SC	NAC	HDSE	IRL	IRE	EI
s	MOD[1:0]		EN	R8	E	E	E	T	R	EN	K	L	P	N	E
-	RW														

Bit	Name	R/W	Reset Value	Function
31: 15	Reserved	-	-	Reserved
14: 13	ABRMOD[1:0]	RW	2'b0	Auto baud rate detection mode. 00: Measure baud rate starting from the start bit 01: Measure from falling edge to falling edge 10: Reserved 11: Reserved When ABREN=0 or UE=0, this register is write-only.
12	ABREN	RW	0	Auto baud rate enabled. 0: Disabled 1: Auto baud rate enabled

Bit	Name	R/W	Reset Value	Function
11	OVER8	RW	0	Oversampling mode. 0: 16x oversampling 1: 8x oversampling This bit is writable only when UE=0.
10	CTSIE	RW	0	CTS interrupt enabled. 0: Disabled; 1: CTSIF interrupt enabled;
9	CTSE	RW	0	CTS enabled. 0: CTS hardware flow control disabled; 1: CTS mode enabled. Data is transmitted only when the CTS input is 0. At this time, after data is written to the data register, transmission will not start until CTS is active.
8	RTSE	RW	0	RTS enabled. 0: RTS hardware flow control disabled; 1: RTS output enabled, the next data is requested only when the receive buffer is not full. After the current data transmission is completed, the sending operation is paused. If data can be received, set RTS to active (0).
7	DMAT	RW	0	Enable DMA when sending. 0: Disabled; 1: Enable DMA during transmission;
6	DMAR	RW	0	Enable DMA during reception. 0: Disabled; 1: Enable DMA during reception;
5	SCEN	RW	0	Smart card mode enable. 0: Disabled; 1: Enable;
4	NACK	RW	0	Smart card NACK enable. 0: NACK transmission disabled when parity error occurs; 1: NACK transmission enabled when parity error occurs;
3	HDSEL	RW	0	Half-duplex selection. 0: Non-half-duplex mode; 1: Half-duplex mode selection;
2	IRLP	RW	0	IrDA low power. 0: Normal mode;

Bit	Name	R/W	Reset Value	Function
				1: IrDA low-power mode;
1	IREN	RW	0	IrDA mode enable. Software enables and clears this register. 0: IrDA disabled; 1: IrDA enabled;
0	EIE	RW	0	Error interrupt enable. 0: Disabled; 1: Frame error FE, overrun error ORE, and noise NE interrupts enabled.

32.5.7. Guard time and prescaler (USART_GTPR)

Address offset: 0x18

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 16	Reserved	-	0	Reserved
15: 8	GT[7:0]	RW	0	Guard time value. This field defines the guard time in baud clock units. This feature is required in smartcard mode. The transmission complete flag is set only after the guard time has elapsed.
7: 0	PSC[7:0]	RW	0	Prescaler value. ■ In infrared (IrDA) low-power mode: PSC[7:0] = Infrared low-power baud rate. Divide system clock to obtain frequency in low-power mode: Source clock is divided by the value in the register (only 8 bits are valid) 00000000: Reserved – Do not write this value; 00000001: Divide source clock by 1; 00000010: Divide source clock by 2;

Bit	Name	R/W	Reset Value	Function
				<ul style="list-style-type: none"> ■ In normal infrared (IrDA) mode: PSC can only be set to 00000001. ■ In smartcard mode: PSC[4:0]: Prescaler value Divide the system clock to provide a clock for the smart card. The value given in the register (lower 5 bits valid) multiplied by 2 serves 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: Bits [7:5] are meaningless in smart card mode.

33. CAN2.0 controller

33.1. Introduction

CAN (Controller Area Network) bus is a bus standard that enables mutual communication between microprocessors or devices without a host.

The CAN controller complies with the CAN bus CAN2.0B protocol.

The CAN bus controller can handle data transmission and reception on the bus. In this product, the CAN controller has 12 sets of filters. Filters are used to select the messages the application wants to receive.

In the CAN controller, the application can send data to the bus through 1 high-priority Primary Transmit Buffer (PTB) and 3 Secondary Transmit Buffers (STB), with the transmission scheduler determining the mailbox sending order. Bus data is obtained through 3 receive buffers (Receive Buffer, referred to as RB). The 3 STBs and 3 RBs can be understood as two 3-level FIFOs, which are entirely hardware-controlled.

The CAN bus controller can also support time-triggered CAN communication.

33.2. CAN main features

- Fully supports CAN2.0B protocol.
- CAN2.0 supports a maximum communication baud rate of 1 Mbit/s
- Supports baud rate prescaler from 1 to 1/32, allowing flexible baud rate configuration.
- 3 receive buffers
 - FIFO mode
 - Error or non-received data will not overwrite stored messages
- 1 high-priority primary transmit buffer (PTB)
- 3 secondary transmit buffers (STB)
 - FIFO method
 - Priority arbitration method
- 12 independent filter groups
 - Supports 11-bit standard ID and 29-bit extended ID
 - Programmable ID CODE bits and MASK bits
- Both PTB/STB support single transmission mode
- Supports silent mode
- Supports loopback mode
- Supports capturing transmission error types and locating arbitration failure positions
- Programmable error warning value
- Supports time-triggered CAN and reception timestamp as specified in ISO11898-4

33.3. CAN functional description

33.3.1. Block diagram

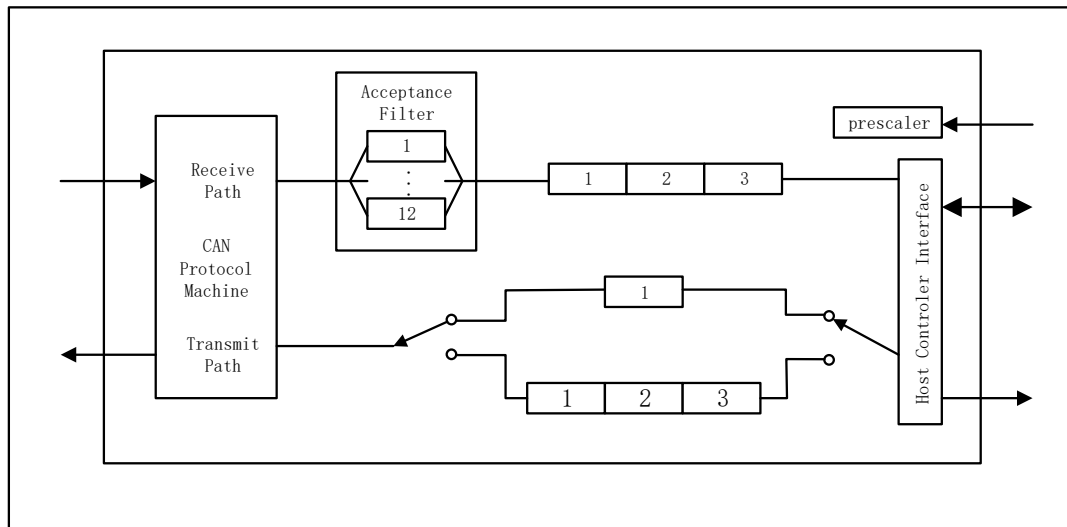


Figure 33-1 CAN module block diagram

33.3.2. Active mode

The CAN controller has two operating modes: reset mode (`CAN_MCR.RESET=1`) and active mode (`CAN_MCR.RESET=0`). During module initialization, registers that can only be operated in reset mode should first be configured in reset mode (see the software reset function section for details), then exit reset mode and operate the remaining registers in active mode.

33.3.3. Baud rate configuration

The clock source for CAN communication, `CAN_clk`, is the external high-speed oscillator HSE or PLL. Before using the CAN module, the CAN communication clock must be configured in the RCC section.

The figure below shows the CAN bit timing definition. The part above the dotted line represents the bit timing specified by the CAN protocol, while the part below the dotted line represents the bit timing defined by this CAN controller `CAN-CTRL`. Among them, segment1 and segment2 can be configured via the `CAN_ACBTR` register. The `CAN_ACBTR` register can only be configured when `CAN_MCR.RESET=1`, i.e., during CAN software reset.

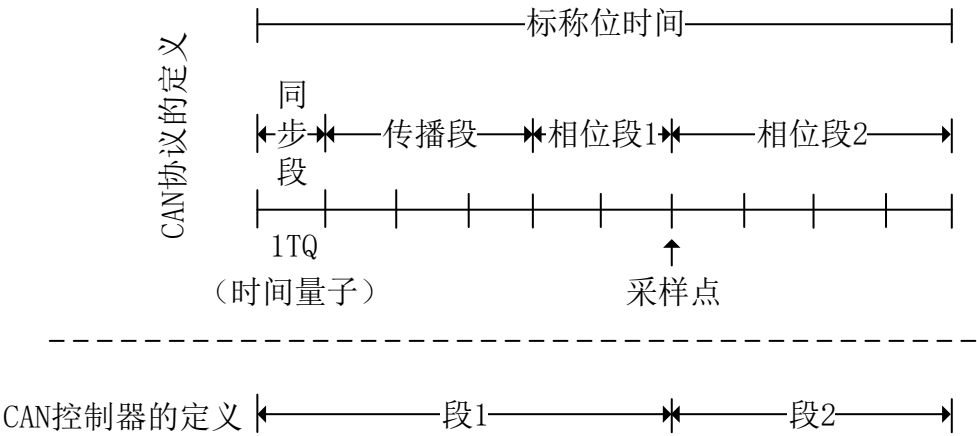


Figure 33-2 CAN Bit Time Definition

Refer to the following formula for TQ calculation method, where PRESC is configured through the PRESC bit of the CAN_RLSSP register. fcan_clk is the CAN communication clock frequency.

$$TQ = \frac{S_PRESC + 1}{f_{can_clk}}$$

Refer to the following formula for bit time calculation method, where segment1 and segment2 are configured through the AC_SEG_1 bit and AC_SEG_2 bit of the CAN_ACBTR register.

$$BT = tS_segment1 + tS_segment2 = ((AC_SEG_1 + 2) + (AC_SEG_2 + 1)) \times TQ$$

Table 33-1 CAN Timing Configuration Rules

Bit	Setting range			Rule
AC_SEG_1 bit of CAN_ACBTR register	[0..511]	CAN2.0 bits	(slow)	SEG_1 ≥ SEG_2 + 1
AC_SEG_2 bit of CAN_ACBTR register	[0..127]	CAN2.0 bits	(slow)	-
AC_SJW bit of CAN_ACBTR register	[0..127]	CAN2.0 bits	(slow)	-

Table 33-2 Recommended Bit Rate Settings for 20 MHz Communication Clock

Bit Rate [Mbps]	PSP [%]	Prescaler	Bit Time [TQ]	Seg 1 [TQ]	Seg 2 [TQ]	SJW [TQ]	TDC [CAN Communication Clock]
0.25 (Arbitration)	80	1	80	64	16	16	-
0.5 (Arbitration)	80	1	40	32	8	8	-
0.5	80	1	40	32	8	8	-
1	80	1	20	16	4	4	16

Table 33-3 Recommended Bit Rate Settings for 40 MHz Communication Clock

Bit Rate [Mbit/s]	PSP [%]	Prescaler	Bit Time [TQ]	Seg 1 [TQ]	Seg 2 [TQ]	SJW [TQ]	TDC [CAN Communication Clock]
0.25 (Arbitration)	80	2	80	64	16	16	-
0.5 (Arbitration)	80	1	80	64	16	16	-
0.5	80	2	40	32	8	8	-
1	80	1	40	32	8	8	32

33.3.4. Transmit Buffer

The CAN controller provides two types of transmit buffers for sending data: the Primary Transmit Buffer (PTB) and the Secondary Transmit Buffer (STB). PTB has the highest priority but can only buffer one frame of data. STB has lower priority than PTB but can buffer 3 frames of data. The 3-frame data within STB can operate in either FIFO mode or priority arbitration mode. The 3-frame data in STB can be transmitted all at once by setting the TSALL bit of the TCMD register to 1. In FIFO mode, the first written data is sent first. In priority mode, data with smaller IDs is sent first. Data in PTB has the highest priority, so PTB transmission can defer STB transmission. However, STB transmissions that have already won arbitration and begun cannot be deferred by PTB transmissions.

PTB and STB can be accessed via the TBUF register. PTB or STB is selected via the TBSEL bit in the TCMD register. TBSEL=0 selects PTB, while TBSEL=1 selects STB. The next SLOT in STB is selected via the TSNEXT bit in the TCTRL register. The correspondence is as shown in the following diagram:

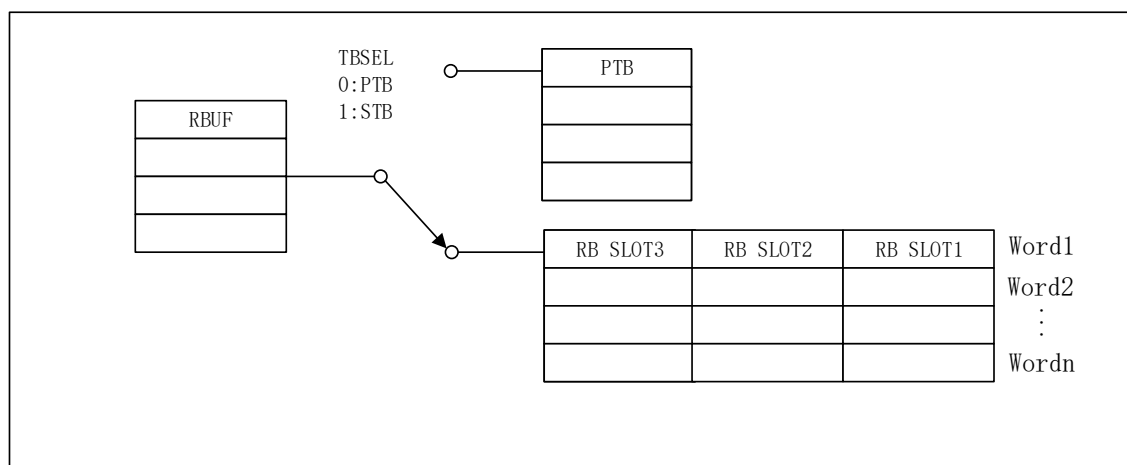


Figure 33-3 Diagram of CAN TBUF Register Writing to Transmit Buffer

33.3.5. Receive Buffer

The CAN controller provides a receive buffer with 3 SLOTS to store received data. These 3 SLOTS operate in FIFO mode. RB SLOT reads received data via the RBUF register, always fetching the earliest received data first. The RREL bit in the RCTRL register is set to 1 to release the read RB SLOT and point to the next RB SLOT.

The diagram for reading RB SLOT via RBUF is as follows.

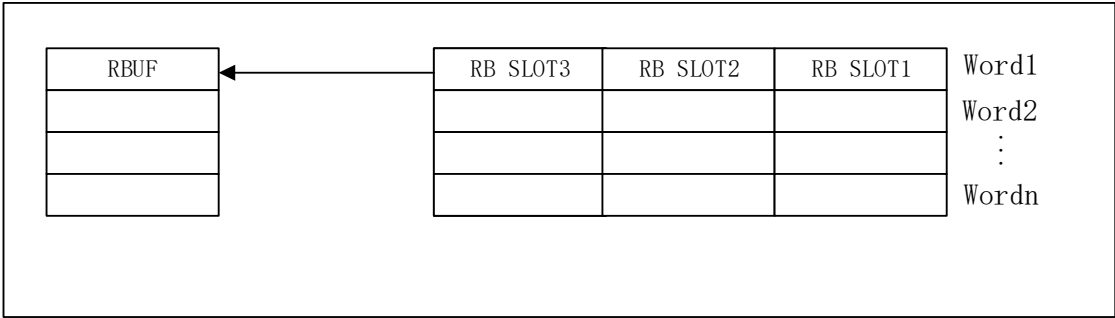


Figure 33-4 Diagram of CAN RBUF Register Reading Receive Buffer

33.3.6. Receive Filter Register Group

The CAN controller provides 12 sets of 32-bit filters to filter received data, thereby reducing CPU load. The filters support either standard 11-bit ID format or extended 29-bit ID format. Each filter group consists of 3 CODE registers (refer to Filter Group CODE Registers) and 3 MASK registers (refer to Filter Group MASK Registers). The CODE registers are used to compare the received CAN frame (refer to LLC Frame Format Definition for frame format), while the MASK registers serve as masks for the CODE registers. When a corresponding MASK bit is 1, the corresponding bit in the CODE register is ignored.

Received data is accepted and stored in the RB if it passes any of the 12 filter groups; otherwise, the data is neither accepted nor stored.

Each filter group is enabled or disabled via the AE_n bit in the CAN_ACFCR register. ID CODE and ID MASK are configured via the CAN_ACFC and CAN_ACFM registers. The filter is selected via the ACFADR bit in the CAN_ACFCR register. ID CODE and ID MASK are accessed via the ACF register and can only be configured when CAN_MCR.RESET=1, i.e., during CAN software reset. Refer to the following diagram for the method of accessing the ACF register filter group.

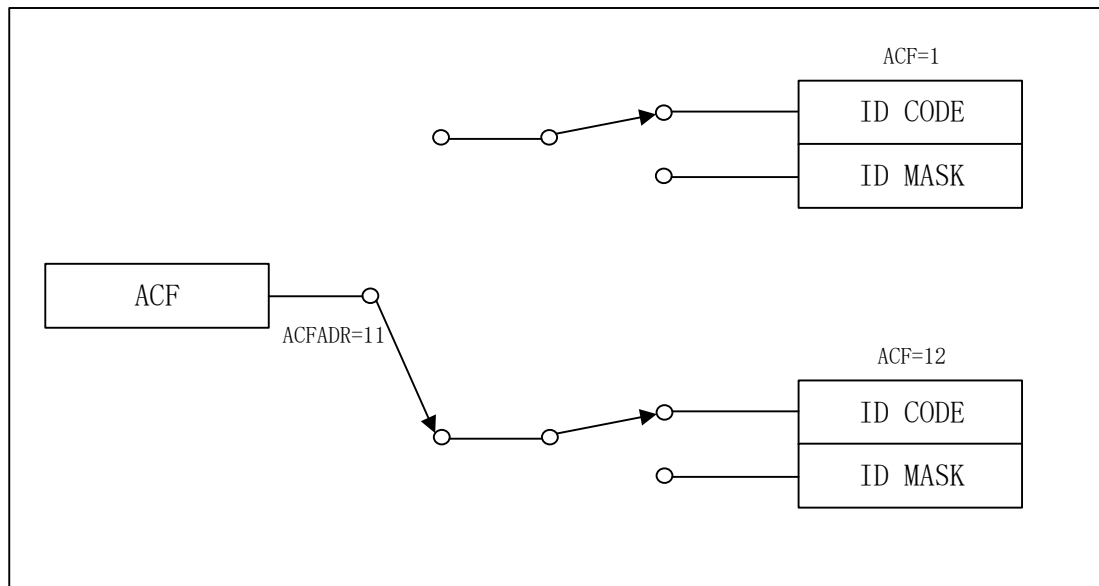


Figure 33-5 Diagram of CAN ACF Register Access Filter Group

33.3.7. LLC Frame Format Definition

The following table shows the definition of the Logical Link Control frame (LLC) containing a timestamp. This unified definition is used for transmitting frames (stored in TBUF), receiving frames (read from RBUF), and configuring receive filters (ACFC and ACFM).

Offset address	Register																
0x00	CAN_ID	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		Res	Res	Res	ID[28:16]												
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		ID[15:0]															
0x04	CAN_FORMAT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		Res	Res	Res	LBF	Res	KOER[2:0]				Res	Res	Res	RMF.	Res	Res	FDF.
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		Res	Res	Res	Res	Res								DLC[3:0]			
0x08	CAN_TYPE	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		HANDLE[7:0]								Res	Res	Res	Res	Res	Res	Res	Res
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		Res															
0x18	CAN_TTCAN	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res

Offset address	Register																
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		CYCLE_TIME[15:0]															
0x1c	CAN_DATA1	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		Data[31:16]															
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		Data[15:0]															
0x20	CAN_DATA2	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		Data[31:16]															
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		Data[15:0]															

Bit	Description
ID	Frame Identifier
DLC	<p>Data Length Code</p> <p>DLC defines the number of payload bytes within the frame.</p> <p>The DLC length of a classic CAN 2.0B frame is 4 bits (DLC(3:0)).</p> <p>For classic CAN 2.0B, it is possible to select remote frames where the DLC has no significance. The bit RMF interpretation provides more details.</p>
IDE	<p>ID format</p> <p>0 – Standard format: ID (28:18)</p> <p>1 – Extended format: ID (28:0)</p>
FDF	This bit should be set to 0 to support CAN 2.0 frame
RMF	<p>Remote frame</p> <p>0 – Data frame</p> <p>1 – Remote frame</p> <p>Remote frames carry zero bytes of payload. The DLC value is transmitted as-is but has no effect on the frame size. Therefore, the DLC value of a remote frame may carry some encoded information.</p> <p>Only classic CAN 2.0B frames can be remote frames.</p>
KOER	<p>Error type</p> <p>The KOER of the received frame has the same meaning as the KOER bit in the EALCAP register. If RBALL=1, the KOER of the received frame becomes meaningful.</p> <p>Note that if RBALL=1, filter groups are generally disabled.</p>
LBF	Loopback frame

Bit	Description
	If loopback mode is activated and the CAN controller has received its own transmitted frame, the LBF of the received frame is set to 1. This is useful if LBME=1 and other nodes in the network also transmit.
HANDLE	<p>Frame identification HANDLE</p> <p>The purpose of the handle is to identify frames using TSTAT. HANDLE is not used in MAC frames. It is recommended that the host application writes the software counter's value to HANDLE. Such a software counter can increment with each new frame to be transmitted and should roll over.</p> <p>Note: To avoid uninitialized memory issues during simulation, the host application should always define HANDLE for each frame to be transmitted.</p>
CYCLE_TIME	<p>Cycle time (timestamp for TTCAN)</p> <p>CYCLE_TIME will only store received frames. This is the cycle time at the frame's SOF. The cycle time of reference messages is always 0.</p>
Data	<p>Payload data of the frame.</p> <p>Classic CAN 2.0B supports up to 8 bytes. Unused payload data words in RBUF contain garbage data that should be ignored.</p>

33.3.8. Data transmission

Before starting transmission, ensure at least one frame of data is loaded in either PTB or STB. TPE is locked during PTB transmission. STB loading status can be confirmed via the TSSTAT bit. The steps for setting transmission data are as follows:

1. Set TBSEL to choose the transmission BUF between PTB and STB
2. Write the data to be transmitted via the TBUF register.
3. If STB is selected, set TSNEXT=1 to complete STB SLOT loading.
4. Transmission enable
 - PTB transmission uses TPE
 - STB transmission uses TSALL or TSONE
5. Transmission completion status confirmation
 - PTB uses TPIF for transmission completion. TPIE is used to enable TPIF.
 - When STB uses TSONE for transmission completion, TSIF is employed. TSIE is used to enable TSIF.
 - When STB uses TSALL for transmission completion, TSIF is employed. TSIF is set only after all STB SLOT data transmission is complete. TSIE is used to enable TSIF.

33.3.9. Cancel Data Transmission

Data transmission that has been requested but not yet executed can be canceled via TPA or TSA.

Canceling data transmission may result in the following scenarios:

1. During arbitration

If the node loses arbitration, data transmission is canceled.

- If the node wins arbitration, transmission continues.

2. During data transmission

- If data is successfully transmitted and an ACK is received, the corresponding flags and statuses are set normally. Data transmission is not canceled.
- Data transmission is canceled if successfully sent but no ACK is received, and the error counter is incremented.
- For data transmission configured with TSALL=1, the STB SLOT data currently being transmitted is sent normally, while STB SLOTS that have not started transmission are canceled.

Canceling data transmission results in the following two scenarios.

1. TPA releases PTB and sets TPE=0.

2. TSA releases one STB SLOT or all STB SLOTS, depending on whether the transmission is enabled by TSONE or TSALL.

33.3.10. Data Reception

The receive filter group can filter out unwanted received data, reducing interrupt occurrences and RB reads, thereby lowering CPU load. The steps for setting up data reception are as follows:

1. Configure the filter group.
2. Set RFIE, RAFIE, and AFWL.
3. Wait for RFIF or RAFIF.
4. Read the earliest received data from the RB FIFO via RBUF.
5. Set RREL=1 to select the next RB SLOT.
6. Repeat steps 4 and 5 until RSTAT confirms that RB is empty.

33.3.11. Error Handling

The CAN controller can automatically handle certain errors, such as retransmitting data or discarding received frames containing errors. On the other hand, it reports errors to the CPU via interrupts.

A CAN node has the following three error states:

- Error-active: When an error is detected, the node automatically sends an active error flag.
- Error-passive: When an error is detected, the node automatically sends a passive error flag.
- Bus-off: In this state, the node no longer affects the entire CAN network.

The CAN controller provides two counters, TECNT and RECNT, for counting errors. The TECNT and RECNT counters increment and decrement according to the rules specified by the CAN protocol. Additionally, a programmable CAN error warning LIMIT register is provided to generate an error interrupt to notify the CPU.

There are five types of errors during CAN communication, and the error type can be identified through the KOER bit in the EALCAP register.

- Bit error
- Form error
- Stuff error
- Acknowledgment error

- CRC error

33.3.12. Bus-off

When the transmit error count exceeds 255, the CAN node automatically enters the bus-off state and does not participate in CAN communication until it returns to the error-active state. The CAN bus-off state can be confirmed through the BUSOFF bit in the CAN_MCR register. BUSOFF is set while the EIF interrupt is generated.

There are two methods for a CAN slave node to recover from the bus-off state to the error-active state:

- Power-on reset
- Receiving a continuous sequence of 128 recessive bits (recovery sequence)

In the bus-off state, the TECNT value remains unchanged, and RECNT is used to count the recovery sequence. After the slave node recovers from the bus-off state, TECNT and RECNT are reset to 0.

33.3.13. Arbitration failure position capture

The CAN controller can accurately capture the position of the arbitration failure bit and reflect it in the WECR. In the ALC register. The WECR.ALC register stores the position of the last arbitration failure bit. If the node wins arbitration, the WECR.ALC bit is not updated. The ALC value is defined as follows:

After the SOF bit, the first ID data bit has ALC=0, the second ID data bit has ALC=1, and so on. Since arbitration only occurs within the arbitration field, the maximum value of ALC is 31. For example, in arbitration between a standard format remote frame and an extended frame, if the extended frame fails at the IDE bit, then ALC=12.

33.3.14. Loopback mode

The CAN controller supports the following two loopback modes:

- Internal loopback
- External loopback

Both loopback modes can receive their own transmitted data frames and are primarily used for testing purposes.

In internal loopback mode, the module internally connects the receive data line to the transmit data line, and transmitted data is not output. In internal loopback mode, the node generates a self-acknowledgment signal to avoid ACK errors.

External loopback mode maintains a connection to the transceiver, so transmitted data still appears on the CAN bus. With the transceiver's assistance, the CAN can receive its own transmitted data. External loopback mode can determine whether to generate a self-acknowledgment signal via the SACK bit in the RCTRL register. When SACK=0, no self-acknowledgment is generated; when SACK=1, a self-acknowledgment is generated.

In external loopback mode with SACK=0, the following two scenarios occur:

- Other nodes also receive the data frame sent by this node and send an acknowledgment signal, allowing successful data transmission and reception for this node.

- If no other node returns an acknowledgment signal, an acknowledgment error occurs, causing data to be resent and the error counter to increment. In this case, single-shot transmission mode is recommended.

When returning from loopback mode to normal mode, in addition to clearing the mode bits, a software reset of the CAN controller is required.

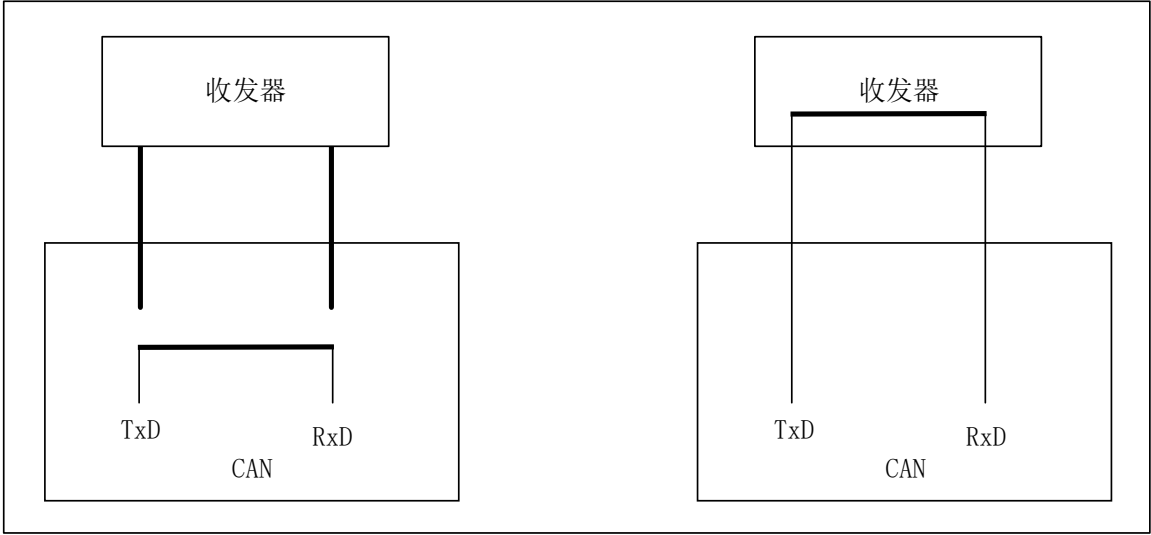


Figure 33-6Diagram of CAN internal loopback LBMI and external loopback LBME

33.3.15. Silent mode

Silent mode can be used to monitor CAN network data. In silent mode, data can be received from the CAN bus, but no data is transmitted to the bus. Set the LOM bit in the CAN_MCR register to 1 to enter silent mode for the CAN bus controller. Clearing it to 0 exits silent mode.

External loopback mode can be combined with silent mode to form external loopback silent mode. In this mode, CAN can be considered a quiet receiver but can send data when necessary. In external loopback silent mode, frames containing self-acknowledgment signals are allowed to be sent, but the node will not generate error flags or overload frames.

33.3.16. Software Reset Function

By setting the RESET bit of the CAN_MCR register to 1, the software reset function is activated. The reset scope of this function is shown in the table below.

Table 33-4 Software Reset Scope Table

Register Bit Name	Software Reset	Note	Register Bit Name	Software Reset	Note
ACFADR	No	-	F_SEG_1	Yes	Writable only during software reset
ACODE	No	Writable only during software reset	F_SEG_2	Yes	Writable only during software reset

Register Bit Name	Software Reset	Note	Register Bit Name	Software Reset	Note
AE_x	No	-	F_SJW	Yes	Writable only during software reset
AFWL	No	-	KOER	Yes	-
AIF	Yes	-	LBME	Yes	-
ALC	Yes	-	LBMI	Yes	-
ALIE	No	-	RACTIVE	Yes	Reception stops immediately and no ACK is generated
ALIF	Yes	-	RAFIE	No	-
AMASK	No	Writable only during software reset	RAFIF	Yes	-
BEIE	No	-	RBALL	Yes	-
BEIF	Yes	-	RBUF	Yes	RB is marked as empty, value indeterminate
BUSOFF	No	Clear by writing 1	RECNT	No	Clear by writing 1 to BUSOFF
EIE	No	-	REF_ID	No	-
EIF	No	-	REF_IDE	No	-
EPASS	No	-	RFIE	No	-
EPIE	No	-	RFIF	Yes	-
EPIF	Yes	-	RIE	No	-
EWARN	No	-	RIF	Yes	-
EWL	Yes	-	ROIE	No	-
			ROIF	Yes	-
F_FRESC	No	Writable only during software reset	ROM	No	
ROV	Yes	-	TSMODE	No	
RREL	Yes	-	TSNEXT	Yes	-
RSTAT	Yes		TSONE	Yes	-
SACK	Yes	-	TPIE	No	-
SELMASK	No	-	TPIF	Yes	-
S_PRESC	No	Writable only during software reset	TPSS	Yes	-
S_SEG_1	No	Writable only during software reset	TSFF	Yes	All STB SLOTS are marked as empty
S_SEG_2	No	Writable only during software reset	TSIE	No	-

Register Bit Name	Software Reset	Note	Register Bit Name	Software Reset	Note
SJW	No	Writable only during software reset	TSIF	Yes	-
SSPOFF	Yes	-	TSSS	Yes	-
TACTIVE	Yes	Send immediate stop	TSSTAT	Yes	All STB SLOTS are marked as empty
TBE	Yes	-	TTEN	Yes	-
TBF	No	-	TTIF	Yes	-
TBPTR	No	-	TTIE	No	-
TBSEL	Yes	-	TTPTR	No	-
TBUF	Yes	The STB is marked as empty, pointing to PTB	TTTBM	No	-
TDCEN	Yes	-	TTYPE	No	-
TECNT	No	Can be cleared by BUSOFF=1	TT_TRIG	No	-
TEIF	Yes	-	TT_WTRIG	No	-
TPA	Yes	-	T_PRESC	No	-
TPE	Yes	-	WTIE	No	-
TSA	Yes	-	WTIF	Yes	
TSALL	Yes	-			

33.3.17. Time-Triggered TTCAN

CAN-CTRL provides partial (level 1) hardware support for the time-triggered communication method specified in ISO11898-4. This section introduces the TTCAN function from the following five parts.

33.3.17.1. TBUF behavior in TTCAN mode

TTTBM=1

When TTTBM=1, PTB and STB SLOT together form the TB SLOT. The transmit buffer is specified by the TBPTR register, where TBPTR=0 points to PTB, TBPTR=1 points to STB SLOT1, and so on. The host can mark the transmit buffer SLOT using the TBE and TBF registers. At this time, the TBSEL and TSNEXT registers are meaningless and can be ignored.

When TTTBM=1, PTB does not have any special attributes. Like STB SLOT, the transmission completion flag also uses TSIF.

In TTCAN mode, the transmit buffer does not have FIFO mode or priority arbitration mode, and only one selected SLOT can transmit data.

In TTCAN mode, transmission initiation requires a time-triggered approach. TPE, TSONE, TSALL, and TPA are fixed to 0 and ignored.

TTTBM=0

When TTTBM=0, event-driven communication and reception timestamp functions are used in combination. In this mode, the functions of PTB and STB are consistent with when TTEN=0, so PTB always has the highest priority, while STB can operate in FIFO mode or arbitration mode.

33.3.17.2. TTCAN Function

After power-on, the Time Master needs to be initialized according to the ISO 11898-4 protocol. A CAN network can have up to 8 potential Time Masters. Each Time Master has its own reference message ID (last 3 bits of the ID). These potential Time Masters transmit their respective reference messages based on their priority. After TTEN=1, the 16-bit counter starts. When a reference message is successfully received or the Time Master successfully transmits a reference message, the CAN controller copies Sync_Mark to Ref_Mark, and Ref_Mark sets the cycle time to 0. Successful reception of a reference message sets the RIF flag, while successful transmission of a reference message sets the TPIF or TSIF flag. At this point, the host needs to prepare the trigger condition for the next action.

The trigger condition can be a reception trigger. This trigger only triggers an interrupt to detect whether the expected message has been received. The trigger condition can also be a transmission trigger. This trigger initiates the transmission of data in the TBUF SLOT specified by the TTPTR register. If the selected TBUF SLOT is marked as empty, transmission does not start, but the interrupt flag is set.

33.3.17.3. TTCAN Timing

The CAN controller supports ISO11898-4 level 1. Includes a 16-bit counter operating under the bit time defined by AC_PRESC, AC_SEG_1, and AC_SEG_2. If TTEN=1, there is an additional prescaler T_PRESC.

At the SOF of a frame, the counter value is Sync_Mark. If the frame data is a reference message, copy Sync_Mark to Ref_Mark. Cycle time equals the counter value minus Ref_Mark. This time serves as the timestamp for receiving messages or the trigger time reference for sending messages.

33.3.17.4. TTCAN trigger modes

The TTYPE register defines the TTCAN trigger mode, the TTPTR register specifies the transmit SLOT, and TT_TRIG specifies the trigger's cycle time.

Includes the following five trigger modes:

- Immediate trigger
- Time trigger
- Single-shot trigger
- Transmission start trigger
- Transmit stop trigger

All triggers except the immediate trigger mode use the TTIF flag. When TTTBM=1, only the time-triggered mode is supported.

Immediate trigger

The trigger is activated by writing to the high byte of TT_TRIG (the written value is irrelevant). In this mode, the data in the TBUF SLOT selected by TTPTR is transmitted immediately. TTIF is not set.

Time trigger

The time-triggered mode only generates an interrupt by setting the TTIF flag and has no other functionality. If a node expects to receive specific data within a particular time window, the time-triggered mode can be used. If the TT_TRIG value is less than the actual cycle time, the TEIF is set with no further action.

Single-shot trigger

Single-shot trigger mode is used to transmit data within the execution time window.

Set the TEW bit to configure the maximum 16 cycle time ticks as defined by ISO11898-4, with a configurable range of 1 to 16. If data transmission does not start within the specified transmission enable time window, the frame is discarded. The corresponding transmit BUF SLOT is marked as empty, and the AIF is set. The data in the corresponding transmit BUF will not be overwritten because it can be retransmitted by setting the TPF.

If the TT_TRIG value is smaller than the actual cycle time, TEIF is set with no further action.

Transmission start trigger

The transmit start trigger is used to participate in arbitration during the arbitration time window. If the message specified by the TTPTR register is not successfully transmitted, the transmit stop trigger can be used to stop the transmission.

If the TT_TRIG value is smaller than the actual cycle time, TEIF is set with no further action.

Transmit stop trigger

The transmit stop trigger is used to stop a transmission that has already started via the transmit start trigger. If transmission is stopped, the transmit frame is discarded, AIF is set, and the selected TBUF SLOT is marked as empty. However, the data in the TBUF SLOT is not overwritten and can be retransmitted by setting TBF.

If the TT_TRIG value is smaller than the actual cycle time, TEIF is set and execution stops.

33.3.17.5. TTCAN watch trigger time

The TTCAN watch trigger time function is similar to a watchdog function and is used when TTTBM=1. Used to monitor the time elapsed since the last successful reception of a reference message. Reference messages can be received during the cycle time or after an event. The application should set an appropriate watch time based on the specific situation.

If the cycle count equals TT_WTRIG, WTIF is set. Disable watch trigger by writing 0 to WTIE. If TT_WTRIG is smaller than the actual cycle time, TEIF is set.

33.3.18. Interrupt

Interrupt flag	Description
RIF	Receive interrupt
ROIF	Receive overflow interrupt
RFIF	Receive buffer full interrupt
RAFIF	Receive buffer almost full interrupt
TPIF	PTB transmission interrupt
TSIF	STB transmission interrupt
EIF	Error interrupt

AIF	Transmission abort interrupt
EPIE	Error passive interrupt
ALIF	Arbitration lost interrupt
BEIF	Bus error interrupt
WTIF	Trigger watchdog interrupt
TEIF	Trigger error interrupt
TTIF	Time-triggered interrupt

33.4. Register description

33.4.1. Node configuration register (CAN_TSNCR)

Address offset: 0x00

Reset value: 0x0201 0801

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	ROP	CES
-														RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VERSION[15:0]															
R															

Bit	Name	R/W	Reset Value	Function
31: 18	Reserved	-	-	Reserved
17	ROP	RW	0	<p>Restricted Operation</p> <p>0: Restricted Operation Disabled</p> <p>1: Restricted Operation Enabled</p> <p>ROP cannot be changed while transmission is active. If ROP is enabled, data frame transmission cannot be initiated.</p>
16	CES	RW	1	<p>CAN error signaling</p> <p>0: Disable error signaling</p> <p>1: Enable error signaling</p> <p>If CES=1, errors are signaled with an error flag and the error counters are incremented. This behavior is equivalent to</p> <p>Definition in ISO 11898-1:2015 (Classic CAN).</p> <p>Otherwise, if CES=0, the error will cause a protocol exception event and the error counters will not be modified.</p>
15: 0	Version [15: 0]	R	0x2050	CAN-CTRL version. VER_1 holds the major version, VER_0 holds the minor version. For example: version

				5x15N00S00 is represented by VER_1=5 and VER_0=15=0x0f.
--	--	--	--	---

33.4.2. Bit timing configuration register (CAN_ACBTR)

Address offset: 0x04

Reset value: 0x0505 0008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	AC_SJW[6:0]							Res	AC_SEG_2[6: 0]						
-	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
Res	Res	Res	Res	Res	Res	Res	AC_SEG_1[8: 0]								
-	-	-	-	-	-	-	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31	Reserved	-	-	Reserved
30: 24	AC_SJW	RW	0x05	Synchronization jump width Synchronization jump width $t_{SJW} = (AC_SJW + 1) \cdot TQ$ is the maximum time to shorten or lengthen the resynchronization bit time.
23	Reserved	-	-	Reserved
22: 16	AC_SEG_2	RW	0x05	Bit timing segment 2 Time $t_{SEG_2} = (AC_SEG_2 + 1) \cdot TQ$ from sample point to bit end
15: 9	Reserved	-	-	Reserved
8: 0	AC_SEG_1	RW	0x08	Bit Timing Segment 1 The sampling point is set to $\square\square\square\square_1 = (AC_1 + 2) \cdot \square\square$ after the start of the bit time.

33.4.3. Limitation and Prescaler Configuration Register (CAN_RLSSP)

Address offset: 0x10

Reset value: 0x7700 0000

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

Bit	Name	R/W	Reset Value	Function
31	Reserved	-	-	Reserved
30: 28	RETLIM	RW	0x07	<p>Automatic retransmission limit</p> <p>111: Unlimited (only limited by the transmit error counter TECNT)</p> <p>110: 7 attempts</p> <p>...</p> <p>000: 1 attempt (no retransmission)</p> <p>If any errors occur during transmission, the CAN node can automatically retry when the bus is idle. RETLIM can be used to limit the number of retransmission attempts.</p> <p>RETLIM can be updated at any time, but after an update, the register contents require several clocks to synchronize with the CAN protocol engine. During this period, REALIM cannot be written again.</p>
27	Reserved	-	-	Reserved
26: 24	REALIM	RW	0x07	<p>Re-arbitration attempt limit</p> <p>111: Unlimited</p> <p>110: 7 attempts</p> <p>...</p> <p>000: 1 attempt (no auto-arbitration)</p> <p>If two or more CAN nodes attempt to transmit frames simultaneously, the lower-priority frame loses arbitration and silently backs off without interrupting the higher-priority frame. A CAN node can automatically retry when the bus is idle. Using REALIM can limit the number of re-arbitration attempts.</p> <p>REALIM can be updated at any time, but after an update, it takes several clock cycles for the register content to synchronize with the CAN protocol engine. During this period, REALIM cannot be written again.</p>
23: 5	Reserved	-	-	Reserved
4: 0	PRESC	RW	0	<p>Prescaler</p> <p>The module input clock is divided by (PRESC+1) to obtain TQ.</p>

33.4.4. Status Register (CAN_IFR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EWARN	EPASS	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
R	R	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	WTIF	TEIF	TTIF	EPIF	ALIF	BEIF	RIF	ROIF	RFIF	RAFIF	TPIF	TSIF	EIF	AIF
-	-	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31	EWARN	R	0	Error Warning limit reached 0: RECNT or TECNT is below the EWL set value 1: RECNT or TECNT reaches or exceeds the EWL set value
30	EPASS	R	0	Error Passive (Error Passive mode active) 0: Node is in error active mode 1: Node is in error passive mode
29: 14	Reserved	-	-	Reserved
13	WTIF	RW	0	TTCAN: Watch trigger interrupt flag If the cycle count reaches the limit defined by TT_WTRIG and WTIE is set, WTIF will be set.
12	TEIF	RW	0	TTCAN: Trigger error interrupt flag The condition for setting TEIF is described in the TTCAN section; there is no bit to enable or disable TEIF processing
11	TTIF	RW	0	Time-triggered interrupt flag When CYCLECOUNT value=TT_TRIG setting and TTIE=1, TTIF is set. If TT_TRIG is not updated, TTIF will only be set once. In the next counting cycle, even if CYCLECOUNT value=TT_TRIG, it will not be set. The application clears this flag by writing 1.
10	EPIF	RW	0	Error passive interrupt flag. If EPASS changes and EPIE=1, EPIF will be set. The application clears this flag by writing 1.
9	ALIF	RW	0	Arbitration Lost Interrupt Flag 0: Arbitration successful 1: Arbitration failed Writing 1 through application clears this flag.
8	BEIF	RW	0	Bus Error Interrupt Flag 0: No bus error

Bit	Name	R/W	Reset Value	Function
				1: Bus error occurred The application clears this flag by writing 1.
7	RIF	RW	0	Receive Interrupt Flag 0: No data frame received 1: Valid data frame or remote frame received The application clears this flag by writing 1.
6	ROIF	RW	0	Receive Overrun Interrupt Flag 0: No RB has been overwritten (overWrite) 1: At least one RB has been overwritten ROIF and RFIF are both set to 1 when overflow occurs. The application clears this flag by writing 1.
5	RFIF	RW	0	Receive Buffer Full Interrupt Flag (RB Full Interrupt Flag) 0: RB FIFO is not full 1: RB FIFO is full The application clears this flag by writing 1.
4	RAFIF	RW	0	Receive Buffer Almost Full Interrupt Flag (RB Almost Full Interrupt Flag) 0: Number of filled RB SLOTS is less than the AFWL set value 1: Number of filled RB SLOTS is greater than or equal to the AFWL set value The application clears this flag by writing 1.
3	TPIF	RW	0	Transmission Primary Interrupt Flag 0: No PTB transmission is completed. 1: The requested PTB transmission is successfully completed. This flag is cleared by writing 1 through the application. Note: In TTCAN mode, TPIF is invalid, and only the TSIF flag is applicable.
2	TSIF	RW	0	Transmission Secondary Interrupt Flag 0: No STB transmission is completed. 1: The requested STB transmission is successfully completed. This flag is cleared by writing 1 through the application. Note: In TTCAN mode, TPIF is invalid, and only the TSIF flag is used.
1	EIF	RW	0	Error Interrupt Flag

Bit	Name	R/W	Reset Value	Function
				<p>0: The BUSOFF bit does not change, or the relative relationship between the error counter value and the ERROR warning limit set value does not change.</p> <p>1: The BUSOFF bit changes, or the relative relationship between the error counter value and the ERROR warning limit set value changes. For example, the value of the error counter changes from less than the set value to greater than the set value, or from greater than the set value to less than the set value.</p> <p>The application clears this flag by writing 1.</p>
0	AIF	RW	0	<p>Abort Interrupt Flag</p> <p>0: Transmission data not aborted</p> <p>1: The transmission requested by TPA or TSA was successfully aborted.</p> <p>The application clears this flag by writing 1.</p>

33.4.5. Interrupt Enable Register (CAN_IER)

Address offset: 0x18

Reset value: 0x0004 68FE

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	WTIE	Res	TTIE	EPIE	ALIE	BEIE	RIE	ROIE	RFIE	RAFIE	TPIE	TSIE	EIE	Res
-	-	RW	-	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 14	Reserved	-	-	Reserved
13	WTIE	RW	1	<p>Watchdog Trigger Interrupt Enable</p> <p>0: Disabled</p> <p>1: Enabled</p>
12	Reserved	-	-	Reserved
11	TTIE	RW	1	<p>Time Trigger Interrupt Enable</p> <p>0: Disabled</p> <p>1: Enabled</p>
10	EPIE	RW	0	<p>Error Warning limit reached</p> <p>0: RECNT or TECNT is below the EWL set value</p> <p>1: RECNT or TECNT reaches or exceeds the EWL set value</p>

Bit	Name	R/W	Reset Value	Function
9	ALIE	RW	0	Arbitration Lost Interrupt Enable 0: Disabled 1: Enabled
8	BEIE	RW	0	Bus Error Interrupt Enable 0: Disabled 1: Enabled
7	RIE	RW	1	Receive Interrupt Enable (Receive Interrupt Enable) 0: Disabled 1: Enabled
6	ROIE	RW	1	Receive Overrun Interrupt Enable (Receive Overrun Interrupt Enable) 0: Disabled 1: Enabled
5	RFIE	RW	1	Receive Buffer Full Interrupt Enable (RB Full Interrupt Enable) 0: Disabled 1: Enabled
4	RAFIE	RW	1	Receive Buffer Almost Full Interrupt Enable (RB Almost Full Interrupt Enable) 0: Disabled 1: Enabled
3	TPIE	RW	1	PTB Transmission Interrupt Enable (Transmission Primary Interrupt Enable) 0: Disabled 1: Enabled
2	TSIE	RW	1	Transmission Secondary Interrupt Enable (STB) 0: Disabled 1: Enabled
1	EIE	RW	1	Error Interrupt Enable 0: Disabled 1: Enabled
0	Reserved	-	-	Reserved

33.4.6. Transmission Status Register (CAN_TSR)

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	TSTAT_H[2:0]			HANDLE_H[7:0]							
-	-	-	-	-	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
Res	Res	Res	Res	Res	TSTAT_L[2:0]			HANDLE_L[7:0]							
-	-	-	-	-	R	R	R	R	R	R	R	R	R	R	R

Bit	Name	R/W	Reset Value	Function
31: 27	Reserved	-	-	Reserved
26: 24	TSTAT_H	R	0	Status code of the completed transmission send frame
23: 16	HANDLE_H	R	0	Handle value of the completed transmission send frame
15: 11	Reserved	-	-	Reserved
10: 8	TSTAT_L	R	0	Status code of the currently transmitting send frame
7: 0	HANDLE_L	R	0	Handle value of the transmit frame currently being transmitted

33.4.7. Global Configuration Register (CAN_MCR)

Address offset: 0x28

Reset value: 0x0090 0080

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SACK	ROM	ROV	RREL	RBALL	Res	RSTAT[1:0]		Res	TSNEXT	TSMODE	TTTBM	Res	TSFF	TSSTAT[1:0]	
RW	RW	R	RW	RW	-	R	R	RW				-	R	R	R
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TBSEL	LOM	STBY	TPE	TPA	TSONE	TSALL	TSA	RE-SET	LBME	LBMI	Res	Res	Res	Res	BUSOFF
RW											-	-	-	-	RW

Bit	Name	R/W	Reset Value	Function
31	SACK	RW	0	Self-ACKnowledge (SACK) 0: No self-acknowledgment 1: When LBME=1, enables self-answer function
30	ROM	RW	0	Receive buffer overflow mode setting bit (Receive buffer Overflow Mode) 0: The earliest received data is overwritten 1: Newly received data is not stored
29	ROV	R	0	Receive buffer overflow flag (Receive buffer OVerflow) 0: No overflow 1: Overflow, at least one data lost Cleared by writing RREL as 1.
28	RREL	RW	0	Release receive buffer (Receive buffer RElease) 0: Do not release 1: Indicates the receive buffer has been read, and the RBUF register points to the next RB SLOT.
27	RBALL	RW	0	Receive BUF data storage stores ALL data frames (Receive Buffer stores ALL data frames) 0: Normal mode 1: Store all data including erroneous data.
25: 24	RSTAT	R	0	Receive buffer status 00 - Empty 01 - > Empty and < Almost full (AFWL) 10 - Almost full (threshold programmable via AFWL) but not full and no overflow 11 - Full (remains set on overflow)
23	Reserved	-	-	Reserved
22	TSNEXT	RW	0	Next STB (Transmit buffer Secondary NEXT) 0: No action 1: Current STB SLOT is filled, point to the next SLOT

Bit	Name	R/W	Reset Value	Function
				<p>After the application writes data into TBUF, it sets the TSNEXT bit to indicate the current STB SLOT is filled, and the hardware then points TBUF to the next STB SLOT.</p> <p>The data in the STB SLOT marked by the TSNEXT bit can be sent via the TSONE or TSALL bit. This bit is set to 1 by the application and cleared by hardware.</p> <p>After all STB SLOTS are filled, TSNEXT remains 1 until an STB SLOT is released.</p> <p>Note: In TTCAN mode, this bit is fixed at 0.</p>
21	TSMODE	RW	0	<p>STB transmit mode (Transmit buffer Secondary operation MODE)</p> <p>0: FIFO mode</p> <p>1: Priority mode</p> <p>FIFO mode sends data frames in the order they are written.</p> <p>Priority mode is automatically determined by ID. The smaller the ID, the higher the priority. In any mode, PTB has the highest priority.</p> <p>Note: The TSMODE bit can only be set when STB is empty.</p>
20	TTTBM	RW	1	<p>TTCAN BUF mode (TTCAN Transmit Buffer Mode). When TTEN=0, TTTBM is ignored.</p> <p>0: Determined by TSMODE, PTB and STB</p> <p>1: Set via TBPTR and TTPTR</p> <p>In TTCAN mode, when only the timestamp reception function is needed, this bit can be set to 0, and TSMODE determines whether PTB or STB is used.</p> <p>Note: The TSMODE bit can only be set when STB is empty.</p>
19	Reserved	-	-	Reserved
18	TSFF	R	0	<p>TTEN=0 or TTTBM=0: STB full flag (Transmit Secondary buffer Full Flag)</p> <p>0: STB SLOT is not completely filled</p> <p>1: STB SLOT is completely filled</p> <p>TTEN=1 and TTTBM=1: TB full flag (Transmit buffer Full Flag)</p> <p>0: The transmit BUF selected by TBPTR is not completely filled</p>

Bit	Name	R/W	Reset Value	Function
				1: The transmit BUF selected by TBPTR is completely filled
17: 16	TSSTAT	R	0	<p>STB status (Transmission Secondary Status bits)</p> <p>TTEN=0 or TTEN=1 & TTTBM=0</p> <p>00: STB empty</p> <p>01: STB less than or equal to half full</p> <p>10: STB more than half full</p> <p>11: STB full</p> <p>TTEN=1 and TTTBM=1</p> <p>00: PTB and STB empty</p> <p>01: PTB and STB not full</p> <p>10: Reserved</p> <p>11: PTB and STB full</p>
15	TBSEL	RW	0	<p>Transmit BUF select bit (Transmit Buffer Select)</p> <p>0: PTB</p> <p>1: STB</p> <p>When TTEN=1&TTTBM=1, TBSEL is reset to its default value.</p> <p>Note: When writing to the TBUF register or TSNEXT bit, this bit must remain constant.</p>
14	LOM	RW	0	<p>Silent mode enable bit (Listen Only Mode)</p> <p>0: Disable silent mode</p> <p>1: Enable silent mode. When LOM=1&LBME=0, transmission is prohibited.</p> <p>When LOM=1&LBME=1, responding to received frames and error frames is prohibited, but data transmission is allowed. Note: This bit is prohibited from being set during communication.</p>
13	STBY	RW	0	<p>Transceiver standby mode</p> <p>0 - Disable</p> <p>1 - Enable</p> <p>This register bit is connected to the output signal stby and can be used to control the transceiver's standby mode.</p> <p>If TPE=1, TSONE=1, or TSALL=1, STBY cannot be set to 1.</p> <p>If the host sets STBY to 0, then before requesting a new transmission, the host needs to</p> <p>Time required to wait for the transceiver to start.</p>
12	TPE	RW	0	PTB Transmission Enable bit (Transmit Primary Enable)

Bit	Name	R/W	Reset Value	Function
				<p>0: Disable PTB transmission 1: Enable PTB transmission</p> <p>When this bit is enabled, the Mailbox in PTB will be transmitted at the next available position. The ongoing STB transmission will continue, but the next pending STB transmission will be delayed until PTB transmission is completed.</p> <p>When this bit is written as 1, it will remain as 1 until PTB transmission is completed or canceled via TPA. Software cannot clear this bit by writing 0.</p> <p>TPE is hardware reset to its default value under the following conditions:</p> <p>RESET=1 BUSOFF=1 LOM=1&LBME=0 TTEN=1&TTTBM=1</p>
11	TPA	RW	0	<p>PTB transmission abort bit (Transmit Primary Abort).</p> <p>0: Do not cancel 1: Cancel a PTB transmission that has been requested by setting TPE to 1 but has not yet started.</p> <p>This bit is written as 1 by software but cleared by hardware. Writing 1 can clear the TPE bit, so write 1 simultaneously with TPE. TPE is hardware reset to its default value under the following conditions:</p> <p>RESET=1 BUSOFF=1 TTEN=1&TTTBM=1</p>
10	TSONE	RW	0	<p>Transmit one frame of STB data set bit (Transmit Secondary ONE frame).</p> <p>0: Do not transmit 1: Transmit one frame of STB data.</p> <p>In FIFO mode, the earliest written data is transmitted; in priority mode, the highest priority data is transmitted.</p> <p>This bit remains set to 1 after being written until the STB transmission is completed or canceled via TSA. Software cannot clear this bit by writing 0.</p> <p>TSONE is hardware reset to its default value under the following conditions:</p> <p>RESET=1</p>

Bit	Name	R/W	Reset Value	Function
				BUFOFF=1 LOM=1&LBME=0 TTEN=1&TTTBM=1
9	TSALL	RW	0	Transmit all STB data setting bit (Transmit Secondary ALL frame) 0: Do not transmit 1: Transmit all data in STB This bit remains set to 1 after being written until the STB transmission is completed or canceled via TSA. Software cannot clear this bit by writing 0. TSALL is reset to its default value by hardware under the following conditions: RESET=1 BUSOFF=1 LOM=1&LBME=0 TTEN=1&TTTBM=1
8	TSA	RW	0	STB Transmission Abort bit (Transmit Secondary Abort) 0: Do not cancel 1: Cancel STB transmissions that have been requested by setting TSONE or TSALL to 1 but have not yet started This bit is set to 1 by software but cleared by hardware. Writing 1 can clear the TSONE or TSALL bit. TSA is reset to its default value by hardware under the following conditions: RESET=1 BUSOFF=1
7	RESET	RW	1	Reset request bit 0: Do not request local reset 1: Request local reset Some registers can only be written when RESET=1. For details, refer to the software reset function. When the node enters BUS OFF state, hardware automatically sets RESET to 1. Note that after RESET=0, it takes 11 CAN bit times for the node to participate in communication.
6	LBME	RW	0	External loopback mode enable bit 0: Disable external loopback mode 1: Enable external loopback mode Note: Setting this bit is prohibited during communication.
5	LBMI	RW	0	Internal loopback mode enable bit

Bit	Name	R/W	Reset Value	Function
				0: Disable internal loopback mode 1: Enable internal loopback mode Note: Setting this bit is prohibited during communication.
4: 1	Reserved	-	-	Reserved
0	BUSOFF	RW	0	Bus-off state 0: Bus active state 1: Bus-off state Note: Writing 1 can clear TECNT and RECNT registers, generally used for debugging purposes only.

33.4.8. Error Warning Register (CAN_WECR)

Address offset: 0x2C

Reset value: 0x0000 001B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TECNT[7:0]								RECNT[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
KOER[2:0]			ALC[4:0]					AFWL[3:0]				EWL[3:0]			
R	R	R	R	R	R	R	R	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 24	TECNT	R	0	Transmit Error Count (TEC) The transmit error counter increments or decrements according to the error counting rules defined by the CAN protocol. This counter does not overflow; 255 is the maximum value.
23: 16	RECNT	R	0	Receive Error Count (RECNT) The receive error counter increments or decrements according to the error counting rules defined by the CAN protocol. This counter does not overflow; 255 is the maximum value.
15: 13	KOER	RW	0	Kind Of Error (KOER) 000: No error 001: Bit error 010: Form error 011: Stuffing error 100: Acknowledgment error 101: CRC error 110: Other errors

Bit	Name	R/W	Reset Value	Function
				111: Reserved The KOER bit updates when there is an error and remains unchanged during normal transmission and reception.
12: 8	ALC	R	0	Arbitration Lost Capture When arbitration fails, ALC records the position where arbitration failed in a frame of data.
7: 4	AFWL	RW	0x1	Receive buffer Almost Full Warning Limit. The setting range is 1~3. AFWL=0 is meaningless and will be treated as AFWL=1.
3: 0	EWL	RW	0xB	Programmable Error Warning Limit value. Error Warning Limit = (EWL + 1) * 8. The value set in this register affects the EIF flag.

33.4.9. Reference ID Register (CAN_REFMSG)

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REF_IDE	Res	Res	REF_ID[28:16]												
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
REF_ID[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31	REF_IDE	RW	0	IDE bit of reference message (REFerence message IDE bit) 0: Standard format 1: Extended format
30: 29	Reserved	-	-	Reserved
28: 0	REF_ID	RW	0	Reference message ID bits (REFerence message IDentifier) REF_IDE=0: REF_ID[10:0] is valid REF_IDE=1: REF_ID[28:0] is valid REF_ID is used to detect the reference message, applicable for both transmission and reception. After detecting the reference message, the Sync_Mark of the current frame becomes Ref_Mark. REF_ID[2:0] is fixed to 0 and its value is not checked, allowing support for up to 8 potential time masters.

				After the highest byte of REF_MSG is written, it is necessary to wait for 6 CAN clock cycles to complete the transfer of REF_MSG to the CAN clock domain.
--	--	--	--	---

33.4.10. TTCAN Configuration Register (CAN_TTCR)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	T_PRESC[1:0]		TTEN	TBE	TBF	TBPTR[5:0]					
-	-	-	-	-	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
TEW[3:0]				Res	TTYPE[2:0]			Res	Res	TTPTR[5:0]					
RW	RW	RW	RW	-	RW	RW	RW	-	-	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 27	Reserved	-	-	Reserved
26: 25	T_PRESC	RW	0	<p>TTCAN counter prescaler</p> <p>00b: 1/1 division of the bit time set by the ACBTR register</p> <p>01b: 1/2 division of the bit time set by the ACBTR register</p> <p>10b: 1/4 division of the bit time set by the ACBTR register</p> <p>11b: 1/8 division of the bit time set by the ACBTR register</p> <p>The TTCAN time base is the CAN bit time defined by PRESC, AC_SEG_1, and AC_SEG_2.</p> <p>T_PRESC defines an additional prescaler factor of 1, 2, 4, or 8.</p> <p>T_PRESC can only be modified when TTEN=0. Alternatively, T_PRESC can be modified while simultaneously setting TTEN with a single write access.</p>
24	TTEN	RW	0	<p>TTCAN Enable (Time Trigger Enable)</p> <p>0: Disabled</p> <p>1: Enable TTCAN, and the counter starts counting.</p>
23	TBE	RW	0	<p>Set TB slot to "empty"</p> <p>0: No operation</p> <p>1: The SLOT selected by TBPTR is marked as empty</p> <p>When a SLOT is marked as empty and TSFF=0, TBE is automatically reset to 0.</p> <p>If this bit is set to 1 and there is data in the selected SLOT being transmitted while TBE=1, TBE will be reset to 0 after the transmission completes, encounters an error, or is canceled.</p>

Bit	Name	R/W	Reset Value	Function
				TBE has higher priority than TBF.
22	TBF	RW	0	Set TB slot to "Filled" 0: No operation 1: The SLOT selected by TBPTR is marked as filled When a SLOT is marked as filled and TSFF=1, TBE is automatically reset to 0.
21: 16	TBPTR	RW	0	TB SLOT pointer (Pointer to a TB message slot) 000: Points to PTB 001: Points to STB SLOT1 010: Points to STB SLOT2 011: Points to STB SLOT3 Others: Invalid setting The pointed TB SLOT can be read and written via TBUF, and its fill status can be marked using TBE and TBF. In TTCAN mode, the TBSEL and TSNEXT registers are invalid Note: This bit can only be written when TSFF=0
15: 12	TEW	RW	0	Transmit Enable Window Single Shot Transmit Trigger mode for TTCAN, which allows setting a window of TEW+1 cycle times where transmission is permitted only within this window.
11	Reserved	-	-	Reserved
10: 8	TTYPE	RW	0	Trigger Type 000: Immediate Trigger for immediate transmission 001: Time Trigger for receive triggers 010: Single Shot Transmit Trigger for exclusive time windows 011: Transmit Start Trigger for merged arbitrating time windows 100: Transmit Stop Trigger for merged arbitrating time windows Others: Reserved The trigger time is set via the TT_TRIG register, and the TB Slot is selected via TTPTR.
7: 6	Reserved	-	-	Reserved
5: 0	TTPTR	RW	0	Transmit Trigger TB slot Pointer 000: Points to PTB 001: Points to STB SLOT1 010: Points to STB SLOT2

Bit	Name	R/W	Reset Value	Function
				011: Points to STB SLOT3 Others: Setting prohibited If the pointed TB SLOT is marked as empty, TEIF is set when the trigger time is reached.

33.4.11. TTCAN Trigger Register (CAN_TTTR)

Address offset: 0x38

Reset value: 0x0202 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TT_WTRIG[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
TT_TRIG[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 16	TT_WTRIG	RW	0xffff	Cycle time of the watchdog trigger After the highest byte operation of TT_WTRIG, the TT_WTRIG value starts transferring to the CAN clock domain. Therefore, for BYTE operations, the low byte must be written first, followed by the high byte.
15: 0	TT_TRIG	RW	0	Trigger Time Used to specify the cycle time of the trigger. For the transmit trigger, the SOF transmission time is approximately TT_TRIG set value +1. After the highest byte operation of TT_WTRIG, the TT_WTRIG value starts transferring to the CAN clock domain. Therefore, for BYTE operations, the low byte must be written first, followed by the high byte.

33.4.12. Memory Status Register (CAN_SCMS)

Address offset: 0x3C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	HELOC[1:0]		TXB	TXS	ACFA	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	R		R	R	RW	-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-	-	-	-	-	-	-	-	-	-	-	-	-			-

Bit	Name	R/W	Reset Value	Function
31: 29	Reserved	-	-	Reserved
28: 27	HELOC	R	0x0	<p>Host-side memory error location</p> <p>00 – No error in host-side access</p> <p>01 – Error in host-side access to TBUF</p> <p>10 – Error in host-side access to RBUF</p> <p>11 – Error in host-side access to ACF</p> <p>During read access from the host side, HELOC will be updated for each new error.</p> <p>This is sufficient because read errors during read access from the CAN side will be signaled by ACFA, TXS, and TXB.</p> <p>HELOC is only updated in case of an error, but not in case of a warning caused by a corrected unit error.</p>
26	TXB	R	0x0	<p>Transmission block</p> <p>0 – Normal operation</p> <p>1 – Transmission blocked</p> <p>If MDEIF or MAEIF is set due to an error while the CAN protocol machine is reading data for transmission, the transmission is immediately blocked.</p> <p>If SEIF is set, the transmission is also immediately blocked.</p> <p>If TXB=1, the error counter is frozen.</p> <p>If RESET=1, then TXB is reset.</p>
25	TXS	R	0x0	<p>Transmission stopped</p> <p>0 – Normal operation</p> <p>1 – Transmission stopped</p> <p>If MDEIF or MAEIF is set due to an error while the priority-sorting machine accesses memory, any new transmission is stopped. If there is an active transmission, it will complete before stopping, but if an error occurs during this transmission, no retransmission will be initiated.</p> <p>If RESET=1, then TXS is reset.</p>
24	ACFA	RW	0x0	<p>Filter enable</p> <p>0 – ACF operates normally</p> <p>1 – ACF disabled: All received frames are accepted</p> <p>ACFA is set if MDEIF or MAEIF is set due to an address range error in ACF. Then acceptance filtering is disabled, and all frames will be accepted.</p>

				ACFA can be reset by writing 1 to it, similar to an interrupt flag. However, since ACFA will be set while reception is still active, it needs to be reset again after reception is completed, for example when RIF is set. If RESET=1, ACFA will also be reset.
23: 0	Reserved	-	-	Reserved

33.4.13. Filter group control register (CAN_ACFCR)

Address offset: 0x44

Reset value: 0x0001 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	AE_11	AE_10	AE_9	AE_8	AE_7	AE_6	AE_5	AE_4	AE_3	AE_2	AE_1	AE_0
-	-	-	-	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
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	ACFADR			
-	-	-	-	-	-	-	-	-	-	-	-	RW			

Bit	Name	R/W	Reset Value	Function
31: 28	Reserved	-	-	Reserved
27: 16	AE_x	RW	0x1	ACF Enable 1 – Enable 0 – Disabled Each acceptance filter (ACFC/ACFM) can be individually enabled or disabled. After hardware reset, only filter number 0 is enabled by default.
15: 4	Reserved	-	-	Reserved
3: 0	ACFADR	RW	0x0	Filter address ACFADR points to a specific acceptance filter. The selected filter can be accessed using registers ACFC and ACFM. Values of ACFADR>ACF_NUMBER-1 are meaningless and are automatically treated as value ACF_NUMBER-1

33.4.14. Filter group code register (CAN_ACFC)

Address offset: 0x48

Reset value: 0xXXXX XXXX

0x00	CAN_ID
0x04	CAN_FORMAT
0x08	CAN_TYPE

Note: For the control definitions of the above registers, refer to Section 33.3.7 LLC Frame Format Definition.

33.4.15. Filter Group Mask Register (CAN_ACFM)

Address offset: 0x58

Reset value: 0xFFFF XXXX

0x00	CAN_ID
0x04	CAN_FORMAT
0x08	CAN_TYPE

Note: For the control definitions of the above registers, refer to Section 33.3.7 LLC Frame Format Definition.

33.4.16. CAN Receive Buffer Register (CAN_RBUF)

Address offset: 0x70

Reset value: 0xFFFF XXXX

Note: For the control definitions of the above registers, refer to Section 33.3.7 LLC Frame Format Definition.

33.4.17. CAN Transmit Buffer Register (CAN_TBUF)

Address offset: 0x94

Reset value: 0xFFFF XXXX

Note: For the control definitions of the above registers, refer to Section 33.3.7 LLC Frame Format Definition.

34. USB Full-Speed Device Interface (USBD)

34.1. Introduction

The USBD peripheral implements the interface between the USB 2.0 Full-Speed bus and the APB bus.

The USBD peripheral supports USB suspend/resume operations, enabling low power consumption by stopping the device clock.

34.2. Key Features of USB

- Compliant with USB 2.0 Full-Speed Device specifications
- Configurable 1 to 6 USB endpoints (Endpoint 0 to Endpoint 5)
- Dedicated 1024-byte packet buffer storage
- CRC (Cyclic Redundancy Check) generation/verification, Non-Return-to-Zero Inverted (NRZI) encoding/decoding, and bit stuffing
- Supports control/isochronous/bulk/interrupt transfers
- Supports dual-buffer mechanism for bulk/isochronous endpoints
- Supports USB suspend/resume operations
- Frame-locked clock pulse generation

34.3. USB Device Block Diagram

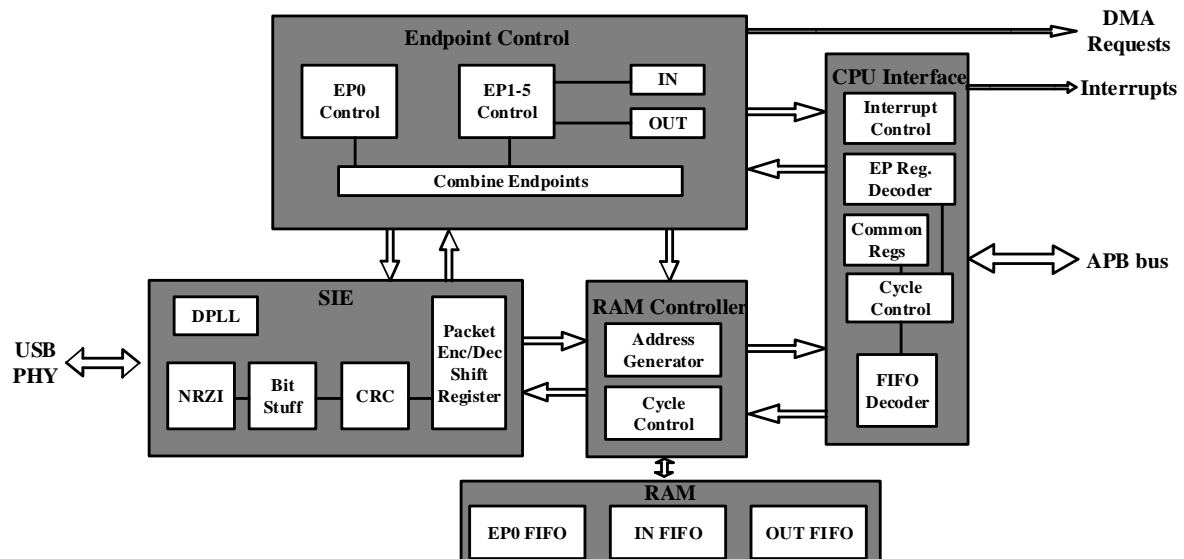


Figure 34-1 USB Device Block Diagram

34.4. Functional Description

The USB module provides a USB specification-compliant communication link between the PC host and microcontroller. Data transfer between the PC host and microcontroller is accomplished through a dedicated data buffer directly accessible by the USB module. The size of this dedicated data buffer

is fixed: Endpoint 0 uses a maximum 64-byte buffer, EP1 uses a 512-byte buffer, EP2/EP3/EP4 each use 128-byte buffers, and EP5 uses a 64-byte buffer. IN and OUT buffers for the same endpoint are shared. The USB module communicates with the PC host, performing token packet detection, data transmission/reception processing, and handshake packet handling in accordance with the USB specification.

When a valid token packet is recognized by the USB module, the corresponding data transfer occurs. The USB module exchanges data between the port and dedicated buffers via an internal register. After all data transfers are completed, it sends or receives the appropriate handshake packet based on the transfer direction if required.

The USB module operates using a fixed clock frequency defined by the USB standard at 48 MHz.

34.4.1. Functional Module Description

The USB module implements all features of a standard USB interface and consists of the following components:

- Serial Interface Engine (SIE): The SIE handles NRZI encoding/decoding, bit stuffing/destuffing, and CRC generation/checking. It generates a 12 MHz USB clock from the input 48 MHz clock, synchronizing data streams with this clock when receiving data from the USB host. It generates headers for transmitted packets and decodes them upon reception.
- Endpoint controller: Uses two state controllers, one for control transfers on endpoint 0 and another for bulk, interrupt, or isochronous transfers on endpoints 1-5.
- CPU interface: The CPU interface allows access to the control/status registers and FIFOs of each endpoint. When a packet is successfully transmitted or received, or when the host enters or resumes from SUSPEND mode, it generates an interrupt to the CPU.
- RAM controller: Used to control the synchronous single-port RAM that buffers data packets between the CPU and USB.

34.4.2. System reset and power-on reset

When a system reset or power-on reset occurs, the first thing the application needs to do is provide the clock signal required by the USB module, then clear the reset signal to allow access to the USB module's registers.

34.4.3. USB reset state

When a USB reset occurs, the USB module enters the state described below:

- Set the ADD[6:0] bits of the USB_CR register to 0;
- Set the INDEX bit of the USB_FRAME register to 0;
- Clear all data in FIFOs;
- Clear all control/status registers;
- Enable all interrupts except SUSPEND;
- Generate a USB reset interrupt.

34.4.4. USB suspend/wakeup mode

When the USB bus remains idle for 3 ms and the Enable_Suspend bit in the USB_CR register is set to 1, the USB module will enter SUSPEND mode. If the USB_INTRE.EN_Suspend bit is set, a SUSPEND interrupt will be generated.

When USB enters SUSPEND mode, the internal 12 MHz data clock will be stopped to reduce power consumption. Meanwhile, the input 48MHz system clock needs to remain active so that the USB module can detect signals on the USB bus. If the system clock stops, the USB module will not be able to detect signals on the USB bus, and the user will have to restart the system clock.

When the USB module is in SUSPEND mode, detecting a K state on the USB bus can wake up the device, causing it to exit SUSPEND mode and generate a RESUME interrupt. Of course, the application can also set the USB_CR.Resume bit to force the device to leave SUSPEND mode. In this case, the application should clear this bit after 10 ms (15 ms maximum). At this point the USB module completely exits SUSPEND mode and will not generate a RESUME interrupt.

34.4.5. IN packet (for data transmission)

The FIFO sizes for IN endpoints 1-5 are 512 Bytes, 128 Bytes, 128 Bytes, 128 Bytes, and 64 Bytes respectively. However, the maximum packet size for IN transfers is configurable and determined by writing to the InMaxP (USB_INEPxCSR) register for each endpoint.

Since each packet to be sent is loaded into the IN FIFO, the InPktRdy bit in the USB_INEPxCSR register must be set. If the AutoSet bit in the USB_INEPxCSR register is set, the InPktRdy bit will be automatically set when the FIFO is loaded with the maximum packet size. For packets smaller than the maximum size, the InPktRdy must be set manually (by the application).

When the InPktRdy bit is set, the FIFONotEmpty bit in the USB_INEPxCSR register will also be set, and the packet is then ready to be sent.

When the packet is successfully sent, the USB_INEPxCSR.InPktRdy bit and USB_INEPxCSR.FIFONotEmpty bit are cleared. If the corresponding interrupt is enabled, the corresponding IN endpoint interrupt will be generated. The next packet can then be loaded into the FIFO.

34.4.5.1. IN packet double buffering feature

If the FIFO size of the IN endpoint is at least twice the maximum packet size of this endpoint (depending on the size of USB_INEPxCSR.InMaxP), then two packets can be cached in the FIFO.

After the first packet is loaded into the FIFO, the USB_INEPxCSR.InPktRdy will be set and then immediately cleared, generating the corresponding IN endpoint interrupt. The second packet can then be loaded into the FIFO, and the USB_INEPxCSR.InPktRdy will be set again, allowing both packets to be sent.

When the first packet is successfully sent, the clearing of the USB_INEPxCSR.InPktRdy bit and the generation of the corresponding IN endpoint interrupt indicate that another packet can now be loaded into the FIFO. At this point, the state of the USB_INEPxCSR.FIFONotEmpty bit indicates how many packets can be loaded. If the USB_INEPxCSR.FIFONotEmpty bit is set, there will be another

packet in the FIFO, and only one packet can be loaded. If the USB_INEPxCSR.FIFONotEmpty bit is in the reset state, it indicates there is no packet in the FIFO, and two packets can be loaded.

34.4.6. OUT packets (used for data reception)

The FIFO sizes for OUT endpoints 1-5 are 512 Bytes, 128 Bytes, 128 Bytes, 128 Bytes, and 64 Bytes respectively. However, the maximum packet size for OUT transfers is programmable and determined by writing to the OutMaxP (USB_OUTEPxCSR) register for each endpoint.

When a data packet is received and placed in the OUT FIFO, the OutPktRdy and FIFOFull bits in USB_OUTEPxCSR will be set, and the appropriate OUT endpoint interrupt will be generated to indicate that a packet can now be read from the FIFO. After the data packet is read, the USB_OUTEPxCSR.OutPktRdy bit must be cleared to receive more packets. If the AutoClear bit in USB_OUTEPxCSR is set and the maximum-sized data packet is read from the FIFO, the OutPktRdy bit will be cleared automatically. The FIFOFull bit is also cleared. For packets smaller than the maximum size, USB_OUTEPxCSR.OutPktRdy must be cleared manually (i.e., by the application).

34.4.6.1. OUT Packet Double Buffering Feature

If the size of the OUT endpoint FIFO is at least twice the maximum packet size of that endpoint (set in the USB_OUTEPxCSR.OutMaxP register), the OUT FIFO can buffer two data packets.

When the first data packet to be received is loaded into the OUT FIFO, the OutPktRdy bit in USB_OUTEPxCSR is set, and the corresponding OUT endpoint interrupt is generated to indicate that the packet can now be read from the FIFO.

Note: The FIFOFull bit in USB_OUTEPxCSR is not set at this point; it is only set when the second data packet is received and loaded into the OUT FIFO.

After reading the first data packet, the USB_OUTEPxCSR.OutPktRdy bit must be cleared to receive more packets. If the AutoClear bit in USB_OUTEPxCSR is set and a maximum-sized data packet is read from the FIFO, the OutPktRdy bit will be cleared automatically. For packets smaller than the maximum size, USB_OUTEPxCSR.OutPktRdy must be cleared manually (i.e., by the application).

If USB_OUTEPxCSR.FIFOFull is set to 1 when clearing USB_OUTEPxCSR.OutPktRdy, the USB_OUTEPxCSR.FIFOFull bit will be cleared first. Then, USB_OUTEPxCSR.OutPktRdy is set again to indicate that another data packet is waiting in the FIFO to be read.

34.4.7. Control Transfer

Endpoint 0 is the control endpoint of USB. Therefore, the routines required to drive Endpoint 0 are more complex than those needed for other endpoints.

Applications can receive and process all standard device requests through Endpoint 0. These are described in the USB Bus Specification. Standard device requests can be divided into three categories: zero-data requests, write requests, and read requests. This section describes the sequence of events that an application must perform when handling different types of device requests.

Note: The Setup token packet associated with any standard device request should include an 8-byte command. Any Setup packet containing a command longer than 8 bytes will be automatically rejected.

34.4.7.1. Zero-data request

All information for zero-data requests is contained in the 8-byte command, with no additional data transfer required. Examples of standard device requests with zero data are: SET_FEATURE, CLEAR_FEATURE, SET_ADDRESS, SET_CONFIGURATION, SET_INTERFACE (refer to the "USB Bus Specification").

As with all requests, the sequence of events will begin when the application receives an endpoint 0 interrupt. The USB_EP0CSR.OutPktRdy bit will also be set. The 8-byte command should be read from endpoint 0 FIFO, decoded, and appropriate action taken. For example, if the command is SET_ADDRESS, the 7-bit address value included in the command should be written to the USB_CR.ADD register.

The application should then set the USB_EP0CSR.ServicedOutPktRdy bit (indicating that the command has been read from the FIFO) and set the USB_EP0CSR.DataEnd bit (indicating that no further data is required for this request).

When the host moves to the status phase of the request as defined in the USB protocol, the USB module will generate a second endpoint 0 interrupt to indicate that the request has been completed. No further action is required by the application; the second interrupt simply confirms the successful completion of the request.

If the command is unrecognized or cannot be executed for some other reason, the application should set the USB_EP0CSR.ServicedOutPktRdy bit and the USB_EP0CSR.SendStall bit after it is decoded. When the host moves to the status phase of the request, the device will send a STALL to inform the host that the request was not executed. The USB module will generate a second endpoint 0 interrupt and set the USB_EP0CSR.SentStall bit.

If the host sends more data after setting the USB_EP0CSR.DataEnd bit, the device will send a STALL handshake packet. The USB device will generate an endpoint 0 interrupt and set the USB_EP0CSR.SentStall bit.

34.4.7.2. Write request

Write requests include one (or more) additional data packets that are sent from the host after the 8-byte command. An example of a standard device write request is: SET_DESCRIPTOR (refer to the "USB Bus Specification").

As with all requests, the sequence of events will begin when the application receives an endpoint 0 interrupt. The USB_EP0CSR.OutPktRdy bit will also be set. The 8-byte command should be read from endpoint 0 FIFO and decoded.

As with a zero-data request, the application should set the USB_EP0CSR.ServicedOutPktRdy bit (indicating the command has been read from the FIFO), but in this case, the USB_EP0CSR.DataEnd bit should not be set (indicating more data is needed).

When the second endpoint 0 interrupt is received, the USB_EP0CSR register should be read to check the endpoint status. The USB_EP0CSR.OutPktRdy bit should be set to indicate that a data packet has been received. The USB_EP0CSR.COUNT0 register should then be read to determine the size of this data packet. Data packets can be read from the endpoint 0 FIFO.

If the data length associated with the request is greater than the maximum packet size of endpoint 0, additional data packets will be sent. In this case, the USB_EP0CSR.ServicedOutPktRdy bit should be set, but the USB_EP0CSR.DataEnd bit should not be set.

When all expected data packets are received, the USB_EP0CSR.ServicedOutPktRdy bit and USB_EP0CSR.DataEnd bit should be set (indicating no more data is needed).

When the host moves to the status phase of the request, the USB device will generate another endpoint 0 interrupt to indicate that the request has been completed. No further action is required from the application; the interrupt is merely confirmation of successful completion.

If the command is unrecognized or cannot be executed for some other reason, the application should set the USB_EP0CSR.ServicedOutPktRdy bit and the USB_EP0CSR.SendStall bit after it is decoded. When the host sends more data, the USB device will send a STALL handshake packet to inform the host that the request was not executed. The USB device will generate an endpoint 0 interrupt, and the USB_EP0CSR.SentStall bit will be set.

If the host sends more data after setting USB_EP0CSR.DataEnd, the USB device will send a STALL handshake packet. An endpoint 0 interrupt will be generated, and the USB_EP0CSR.SentStall bit will be set.

34.4.7.3. Read request

After the 8-byte command, a read request has one (or more) data packets sent from the device to the host. Examples of standard device requests include: GET_CONFIGURATION, GET_INTERFACE, GET_DESCRIPTOR, GET_STATUS, SYNCH_FRAME (refer to the "USB Bus Specification").

As with all requests, the sequence of events will begin when the application receives an endpoint 0 interrupt. The USB_EP0CSR.OutPktRdy bit will also be set. The 8-byte command should be read from endpoint 0 FIFO and decoded. Then the ServicedOutPktRdy bit in the USB_EP0CSR register should be set (indicating the command has been read from the FIFO).

Data to be sent to the host should be written to the FIFO of endpoint 0. If the data to be sent is larger than the maximum packet size of endpoint 0, only the maximum packet size should be written to the FIFO. Then the InPktRdy bit in the USB_EP0CSR register is set (indicating there is a data packet in the FIFO ready to be sent). When the data packet is sent to the host, another endpoint 0 interrupt will be generated, and the next data packet can be written to the FIFO.

When the last data packet is written to the FIFO, the InPktRdy bit and DataEnd bit in the USB_EP0CSR register should be set (indicating no more data after this packet).

When the host moves to the status phase of the request, the USB device will generate another endpoint 0 interrupt to indicate that the request has been completed. No further action is required by the application; the interrupt is merely a confirmation of successful completion.

If the command is unrecognized or cannot be executed for some other reason, the USB_EP0CSR.ServicedOutPktRdy bit and USB_EP0CSR.SendStall bit should be set after it is decoded. When the host requests data, the USB device will send a STALL handshake packet to inform the host that the request was not executed. Then an endpoint 0 interrupt will be generated, and the USB_EP0CSR.SentStall bit will be set.

If the host requests more data after setting USB_EP0CSR.DataEnd, the USB device will send a STALL handshake packet. An endpoint 0 interrupt will be generated, and the USB_EP0CSR.SentStall bit will be set.

34.4.7.4. Error Handling

The USB device automatically detects protocol errors under the following conditions and sends a STALL handshake packet to the host.

- The host sends more data than specified in the command during the OUT DATA phase of a write request. This condition is detected when the host sends an OUT token after the USB_EP0CSR.DataEnd bit is set.
- The host requests more data than specified in the command during the IN DATA phase of a read request. This condition is detected when an IN token is sent after the USB_EP0CSR.DataEnd bit is set.
- The OUT packet sent by the host contains more data bytes than specified in the command.
- The host sends a non-zero-length DATA1 packet during the status phase of a read request.

After the USB device sends a STALL handshake packet, it sets the USB_EP0CSR.SentStall bit and generates an interrupt. When the application receives an endpoint 0 interrupt with the USB_EP0CSR.SentStall bit set, it should abort the current transfer, clear the USB_EP0CSR.SentStall bit, and return to the initial state.

If the host enters the status phase before all data transfers are completed, or sends a new SETUP token packet before the current transfer is finished, prematurely ending the transfer, the USB_EP0CSR.SetupEnd bit will be set, and an endpoint 0 interrupt will be generated. When the application receives an endpoint 0 interrupt with the USB_EP0CSR.SetupEnd bit set, it should abort the current transfer, set the USB_EP0CSR.ServicedSetupEnd bit, and return to the initial state. If the USB_EP0CSR.OutPktRdy bit is set, this indicates that the host has sent another SEPUP token packet, and the application should then process this command.

If the application wishes to terminate the current transfer because it cannot process the command or has other internal errors, it should set the USB_EP0CSR.SendStall bit. The USB device will then send a STALL handshake packet to the host, set the USB_EP0CSR.SentStall bit, and generate an endpoint 0 interrupt.

34.4.8. Isochronous transfer

The USB standard defines a full-speed transfer mode that requires maintaining a fixed and precise data transfer rate: isochronous transfer. Isochronous transfers are typically used for data with strict rate requirements, such as audio streams or compressed video streams. If an endpoint is defined

as an "isochronous endpoint" during enumeration, the USB host allocates fixed bandwidth for each frame and ensures that exactly one IN or OUT transaction (determined by the endpoint direction) is transmitted per frame. To meet bandwidth requirements, there is no error retransmission in isochronous transfers; this means that after sending or receiving a packet, no handshake protocol (such as an ACK handshake packet) is used. Similarly, isochronous transfers only transmit DATA0 PID packets and do not use the data toggle mechanism.

34.4.8.1. Isochronous read transfer

Isochronous read transfers are used to transmit periodic data from the USB module to the host.

Three optional features are available for isochronous IN endpoints:

■ Double-Packet Buffering Function

Double packet buffering is automatically enabled when the value written to the USB_INEPxCSR.InMaxP register is less than or equal to half of the FIFO size allocated to the endpoint. When enabled, up to two packets can be stored in the FIFO awaiting transmission to the host.

■ DMA Function

If DMA is enabled for an endpoint, a DMA request will be generated whenever the endpoint can receive another packet in its FIFO. This feature allows packets to be loaded into the FIFO without processor intervention. However, this feature is not particularly useful for isochronous endpoints because transmitted packets are typically not the maximum size, and the USB_INEPxCSR register needs to be accessed after each packet to check for Underrun errors.

■ AutoSet feature

When the AutoSet feature is enabled, the USB_INEPxCSR.InPktRdy bit will be automatically set when a packet of USB_INEPxCSR.InMaxP bytes is loaded into the FIFO. However, this feature is not particularly useful for isochronous endpoints because transmitted packets are typically not the maximum size, and the USB_INEPxCSR register needs to be accessed after each packet to check for Underrun errors.

Before using an isochronous IN endpoint, the USB_INEPxCSR.InMaxP register must be written with the maximum packet size (in bytes) for that endpoint. This value should be the same as the field in the standard endpoint descriptor of the endpoint. In addition, the relevant interrupt enable bits in the USB_INTRE register should be set to 1, and certain bits in the USB_INEPxCSR register should be configured as follows:

Table 34-1 USB_INEPxCSR Register

AutoSet	ISO	Mode	DMAEnab	FrcDataTog
0/1	1	1	0/1	0

A synchronous endpoint does not support data retransmission, so to avoid data errors, the data sent to the host must be loaded into the FIFO before the IN token is received. The host will send an IN token every frame, but the timing within the frame can vary. If an IN token is received at the end of one frame and then transmitted at the start of the next frame, there will be very little time to reload the FIFO. For this reason, double buffering is typically required in synchronous IN endpoints.

The AutoSet feature can be used for synchronous IN endpoints, but unless the data from the source arrives at an absolutely consistent rate and is synchronized with the host's frame clock, the size of the packets sent to the host will have to be increased or decreased frame by frame to match the source data rate. This means the actual packet size is not always the USB_INEPxCSR.InMaxP size, rendering the AutoSet feature useless.

An interrupt is generated when a data packet is sent to the host, and the application can use this interrupt to load the next packet into the FIFO and set the InPktRdy bit in the USB_INEPxCSR register. Since interrupts can occur at almost any time within a frame, depending on when the host schedules the routine, this may result in irregular timing for FIFO load requests. If the endpoint's data source comes from some external hardware, it may be more convenient to wait for the end of each frame before loading the FIFO, as this minimizes the need for additional buffering. The above operation can be achieved by using the SOF interrupt or the SOF_PULSE signal from the USB device to trigger the loading of the next data packet. When a SOF packet is received, the USB device generates a SOF_PULSE signal (the USB also maintains an external frame counter, so it can still generate SOF_PULSE if the SOF packet is lost) and generates a SOF interrupt. This interrupt/signal can be used to set the InPktRdy bit in USB_INEPxCSR and check for data overflow/missing.

If the endpoint receives an IN token packet and there is no data in the FIFO, it will send an empty data packet to the host and set the UnderRun bit in the USB_INEPxCSR register. This indicates that the application is not providing data to the host fast enough. The application decides how to handle this error condition.

If the application loads one packet per frame and finds that the InPktRdy bit in the USB_INEPxCSR register is set when it attempts to load the next packet, this indicates the packet has not been sent (perhaps due to a corrupted IN token packet from the host). The application can choose to flush un-sent packets by setting the FlushFIFO bit in the USB_INEPxCSR register or skip the current packet.

34.4.8.2. Synchronous Write Transfer

Synchronous OUT endpoints are used to transfer periodic data from the function controller to the host. Three optional features are available for synchronous OUT endpoints:

1. Double Packet Buffering Feature

Double packet buffering is automatically enabled when the value written to the USB_OUTEPxCSR.OutMaxP register is less than or equal to half the size of the FIFO allocated to the endpoint. When enabled, up to two packets can be stored in the FIFO.

2. DMA Feature

If DMA is enabled for the endpoint, a DMA request will be generated whenever there is a packet in the endpoint's FIFO. This feature allows data to be read from the FIFO without processor intervention. However, this feature is not particularly useful for synchronous endpoints because the transmitted packets are usually not the maximum packet size, and the USB_OUTEPxCSR register needs to be accessed after each packet to check for overflow or CRC errors.

3. AutoClear Feature

When the AutoClear feature is enabled, the USB_OTEPxCSR.OutPktRdy bit will be automatically cleared when USB_OTEPxCSR.OutMaxP bytes of data are read from the FIFO. However, this feature is not particularly useful for synchronous endpoints because the transmitted packets are usually not the maximum packet size, and the USB_OTEPxCSR register needs to be accessed after each packet to check for overflow or CRC errors.

Before using a synchronous OUT endpoint, the USB_OTEPxCSR.OutMaxP register must be written with the maximum packet size of the endpoint in bytes. This value should be the same as the field in the standard endpoint descriptor of the endpoint. In addition, the relevant interrupt enable bits in the USB_INTRE register should be set to 1, and certain bits in the USB_OTEPxCSR register should be configured as shown below.

Table 34-2 USB_OTEPxCSR Register

AutoClear	ISO	DMAEnab
0/1	1	0/1

A synchronous endpoint does not support data retransmission, so to avoid data overflow, there must be space in the FIFO to receive packets. The host will send one packet per frame, but the timing within the frame can vary. If one packet is received at the end of a frame and another arrives at the start of the next frame, there is little time to read the data from the FIFO. Therefore, for synchronous OUT endpoints, double packet buffering is usually required.

The AutoClear feature can be used with synchronous OUT endpoints. However, unless the data receiver receives data at an absolutely consistent rate and is synchronized with the host's frame clock, the size of the packets sent by the host will have to increase or decrease frame by frame to match the required data rate. This means the actual packet size is not always the USB_OTEPxCSR.OutMaxP size, making the AutoClear feature useless.

When a data packet is received from the host, an interrupt is generated, which the application can use to read data from the FIFO and clear the OutPktRdy bit in the USB_OTEPxCSR register. Since interrupts can occur at almost any time within a frame, depending on when the host scheduled the transaction, the timing of FIFO read requests may be irregular. If the endpoint's data receiver is to be connected to some external hardware, it is better to wait for the end of each frame before reading the FIFO, thereby reducing the need for additional buffering. This can be done by using the SOF interrupt or the SOF_PULSE signal from the USB device to trigger the reading of the packet. When an SOF packet is received, the USB device generates an SOF_PULSE signal (the USB device also maintains an external frame counter, so it can still generate SOF_PULSE when SOF packets are lost) and generates an SOF interrupt. This interrupt/signal can be used to clear the OutPktRdy bit in USB_OTEPxCSR and check for data overflow/missing.

If there is no space in the FIFO to store the packet received from the host, the OverRun bit in the USB_OTEPxCSR register will be set. This indicates that the application is not reading data fast enough. The application determines how to handle this error condition.

If a USB device detects a received packet with a CRC error, it still stores the packet in the FIFO and sets the USB_OUTEPxCSR.OutPktRdy bit and the USB_OUTEPxCSR.DataError bit. How to handle this error condition is determined by the application.

34.4.9. Bulk transfer

34.4.9.1. Bulk read transfer

Bulk IN endpoints are used to transfer non-periodic data from the microcontroller to the host. There are three optional features available for bulk IN endpoints:

- Double-Packet Buffering Function

If the value written to the USB_INEPxCSR.InMaxP register is less than or equal to half the FIFO size allocated to the endpoint, the double packet buffering feature is automatically enabled. When enabled, up to two packets can be stored in the FIFO pending transmission to the host.

- DMA Function

If DMA is enabled for an endpoint, a DMA request is generated whenever the endpoint can accept another packet in its FIFO. This feature can be used to load packets into the FIFO without processor intervention.

The DMA startup process is as follows:

1. Configure the memory address as the source address for DMA transfer in the DMA control register, and the USB FIFO address as the peripheral address, with the transfer direction set to read from memory;
2. Configure the total number of bytes to be transferred in the DMA control register;
3. Configure the channel priority on the DMA register;
4. Configure whether to generate a DMA interrupt when half or all of the transfer is completed, based on application requirements;
5. Activate the channel on the DMA register;
6. Enable the USB_INEPxCSR.DMAEnab bit;
7. After the DMA finishes transferring data to the FIFO, the application sets the USB_INEPxCSR.InPktRdy bit, and the USB device then begins transferring data to the USB host.

Note: It is recommended to use the DMA feature together with the AutoSet feature described below, so that the application does not need to set the USB_INEPxCSR.InPktRdy bit except for the last packet.

- AutoSet feature

When the AutoSet feature is enabled, the USB_INEPxCSR.InPktRdy bit will be automatically set when a packet of USB_INEPxCSR.InMaxP bytes is loaded into the FIFO. This is particularly useful when using DMA to load the FIFO, as it avoids any processor intervention when loading individual packets during bulk transfers.

Before using bulk IN endpoints, the maximum packet size (in bytes) specified in wMaxPacketSize (refer to the USB Bus Specification) must be written into the USB_INEPxCSR.InMaxP register. Additionally, the relevant interrupt enable bit in the USB_INTRE register should be set to 1, and the USB_INEPxCSR register should be configured as shown below.

Table 34-3 USB_INEPxCSR Register

AutoSet	ISO	Mode	DMAEnab	FrcDataTog
0/1	0	1	0/1	0

When configuring a bulk IN endpoint for the first time, after executing the SET_CONFIGURATION or SET_INTERFACE command on endpoint 0, the USB_INEPxCSR register should be written to set the ClrDataTog bit. This ensures the data toggle starts in the correct state. Additionally, if there are any data packets in the FIFO (indicated by the set USB_INEPxCSR.FIFONotEmpty bit), they should be flushed by setting the USB_INEPxCSR.FlushFIFO bit.

When data is to be transmitted through a bulk IN endpoint, the data packet must be loaded into the FIFO and the USB_INEPxCSR.InPktRdy bit must be set. After data transmission is completed, the USB_INEPxCSR.InPktRdy bit is cleared, and an interrupt is generated so that the next packet can be loaded into the FIFO. If double packet buffering is enabled, the USB_INEPxCSR.InPktRdy bit will be cleared immediately after the first packet is loaded and the USB_INEPxCSR.InPktRdy bit is set, generating an interrupt so that the second packet can be loaded into the FIFO. The application should operate in the same way, loading a data packet when it receives an interrupt, regardless of whether double packet buffering is enabled.

The size of the data packet must not exceed the value specified in the USB_INEPxCSR.InMaxP register. When transmitting a data block larger than USB_INEPxCSR.InMaxP, it must be sent as multiple data packets. The size of these data packets should all match the value specified in the USB_INEPxCSR.InMaxP register, except for the last packet. The host can determine that all data has been transmitted by knowing the expected total amount of data. Alternatively, when it receives a packet smaller than the USB_INEPxCSR.InMaxP size, it can infer that all data has been transmitted. In the latter case, if the total size of the data block is an integer multiple of the value defined in the USB_INEPxCSR.InMaxP register, an empty data packet needs to be sent after all data transmission is completed. This is done by setting USB_INEPxCSR.InPktRdy and, upon receiving the next interrupt, not loading any data into the FIFO.

If transferring large amounts of bulk data, using DMA can avoid invoking the interrupt service routine to load each packet.

If the application wants to disable a bulk IN endpoint, it should set the USB_INEPxCSR.SendStall bit. When the USB device receives the next IN token, it will send a STALL handshake packet to the host, and the USB_INEPxCSR.SentStall bit will be set, generating an interrupt.

When the application receives an interrupt with the USB_INEPxCSR.SentStall bit set, it should clear the USB_INEPxCSR.SentStall bit. Of course, it should also retain the USB_INEPxCSR.SendStall bit setting until ready to re-enable bulk IN transfers.

Note: If the host cannot receive the STALL handshake packet for any reason, it will send another IN token packet. Therefore, it is recommended to keep the USB_INEPxCSR.SendStall bit set until the application is ready to re-enable bulk IN transfers. When a bulk IN transfer is re-enabled, the data toggle sequence should be restarted by setting the ClrDataTog bit in the USB_INEPxCSR register.

34.4.9.2. Bulk Write Transfer

Bulk OUT endpoints are used to transfer non-periodic data from the host to the USB module. Three optional features are available for bulk OUT endpoints:

1) Double-Packet Buffering Function

If the value written to the USB_OUTEPxCSR.OutMaxP register is less than or equal to half of the FIFO size allocated to the endpoint, double-packet buffering is automatically enabled. When enabled, the FIFO can store up to two packets.

2) DMA Function

If DMA is enabled for an endpoint, a DMA request is generated whenever there is a packet in the endpoint's FIFO. This feature allows reading packets from the FIFO without processor intervention.

The DMA startup process is as follows:

1. Configure the memory address as the source address for DMA transfer and the USB FIFO address as the peripheral address in the DMA control register, with the transfer direction set to read from the peripheral;
2. Configure the total number of bytes to be transferred in the DMA control register;
3. Configure the channel priority on the DMA register;
4. Configure whether to generate a DMA interrupt when half or all of the transfer is completed, based on application requirements;
5. Activate the channel on the DMA register;
6. Enable the USB_OUTEPxCSR.DMAEnab bit;
7. After the DMA completes transferring data from the FIFO, the application clears the USB_OUTEPxCSR.OutPktRdy bit.

Note: It is recommended to use the DMA feature together with the AutoClear feature described below, so that the application does not need to clear the USB_OUTEPxCSR.OutPktRdy bit for all packets except the last one.

3) AutoClear Feature

When the AutoClear feature is enabled, the USB_OUTEPxCSR.OutPktRdy bit is automatically cleared after a packet of USB_OUTEPxCSR.OutMaxP bytes is unloaded from the FIFO. This is particularly useful when using DMA to read data from the FIFO. This avoids processor intervention when reading individual packets during large bulk transfers.

Before using a bulk OUT endpoint, the maximum packet size (in bytes) specified in wMaxPacketSize (refer to the USB Bus Specification) must be written to the USB_OUTEPxCSR.OutMaxP register. Additionally, the relevant interrupt enable bit in the USB_INTRE register should be set to 1 (if interrupts are required for this endpoint), and the corresponding bits in the USB_OUTEPxCSR register should be configured as shown below.

Table 34-4 USB_OUTEPxCSR Register

AutoClear	ISO	DMAEnab
0/1	0	0/1

When configuring a bulk OUT endpoint for the first time, after the SET_CONFIGURATION or SET_INTERFACE command on endpoint 0, the USB_OUTEPxCSR.ClrDataTog bit should be set.

This ensures the data toggle starts in the correct state. Similarly, if there are any packets in the FIFO (OutPktRdy bit is set), they should be flushed by setting the USB_OTEPxCSR.FlushFIFO bit.

When a packet is received by a bulk OUT endpoint, the USB_OTEPxCSR.OutPktRdy bit is set and an interrupt is generated. The application should read the USB_OUTCOUNT.OUTCOUNT register of the endpoint to determine the packet size. The packet is then read from the FIFO, and the USB_OTEPxCSR.OutPktRdy bit is cleared.

The packet size should not exceed the value specified in the USB_OTEPxCSR.OutMaxP register. When a data block larger than USB_OTEPxCSR.OutMaxP is to be sent to the USB device, it is transmitted as multiple packets. Except for the last data packet, all other packets have the size specified in the USB_OTEPxCSR.OutMaxP register, while the last packet contains the remaining portion. The application can use specific methods to determine the total size of the transfer, thereby identifying when the last packet is received. Alternatively, when it receives a packet smaller than the size specified by the USB_OTEPxCSR.OutMaxP register, it may infer that all data has been received (if the total size of the data block is an integer multiple of USB_OTEPxCSR.OutMaxP, a zero-length packet will be sent after the data to indicate completion of the transfer).

If transferring large amounts of bulk data, using DMA can avoid calling the interrupt service routine to read data for each packet.

If the application wants to disable the use of a bulk OUT endpoint, it should set the USB_OTEPxCSR.SendStall bit. When the USB device receives the next data packet, it will send a STALL handshake packet to the host, set the USB_OTEPxCSR.SentStall bit, and generate an interrupt.

When the application receives an interrupt with the USB_OTEPxCSR.SentStall bit set, it should clear the USB_OTEPxCSR.SentStall bit. The application should also retain the setting of the USB_OTEPxCSR.SendStall bit until it is ready to re-enable the bulk OUT endpoint.

Note: If the host fails to receive the STALL handshake packet for any reason, it will send another packet, so it is recommended to retain the USB_OTEPxCSR.SendStall bit until the application is ready to re-enable bulk OUT transfers. When re-enabling bulk OUT transfers, the USB_OTEPxCSR.ClrDataTog bit should be set to restart the data toggle sequence.

34.4.10. Interrupt transfer

34.4.10.1. Interrupt read transfer

Interrupt IN endpoints are used to transfer periodic data from the function controller to the host.

Interrupt IN endpoints use the same protocol as bulk IN endpoints and can be used in the same manner. Although DMA can be used, it provides little benefit since interrupt endpoints typically expect all data to be transferred in a single packet.

Interrupt IN endpoints also support a feature not available for bulk IN endpoints: continuous toggling of the data toggle bit. This feature is enabled by setting the FrcDataTog bit in the USB_INEPxCSR register. When this bit is set to 1, the USB device will consider the packet successfully sent and toggle the data bit for the endpoint, regardless of whether an ACK was received from the host.

34.4.10.2. Interrupt write transfer

Interrupt OUT endpoints are used to transfer periodic data from the host to the function controller.

Interrupt OUT endpoints use nearly the same protocol as bulk OUT endpoints and can be used in the same manner. Although DMA can be used with an interrupt OUT endpoint, it generally provides little benefit since interrupt endpoints typically expect all data to be transferred in a single packet.

34.5. USB registers

Note: When writing to USB registers, operations must be performed in byte or word units only. When reading from USB registers, operations must be performed in byte units only.

34.5.1. USB Control Register (USB_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISO Up- date	Res			Re- set	Re- sume	Sus- pend Mode	Enable Sus- pend	Up- date	ADD						
RW	-			R	RW	R	RW	R	RW						

Bit	Name	R/W	Reset Value	Function
31: 16	Reserved	-	0	Reserved
15	ISO_Update	RW	0	ISO_Update: After setting this bit to 1, the USB controller needs to wait for a SOF after setting InPktRdy to 1 before sending a data packet; if an IN Token is received before the SOF, the USB controller will send a zero-length packet. (This register is only used during ISO transfer.)
14: 12	Reserved	-	-	Reserved
11	Reset	R	0	Reset: This bit is set to 1 when there is a reset signal on the USB bus.
10	Resume	RW	0	Resume: When the USB device is in Suspend mode, setting this bit to 1 by software generates a Resume wake-up signal; the software should clear this bit after 10ms. (Maximum 15mS)
9	Suspend_Mode	R	0	Suspend_Mode: Set to 1 by USB device hardware when entering Suspend mode; Cleared

				when software reads the interrupt register or writes to the Resume register
8	Enable_Suspend	RW	0	Enable_Suspend: Suspend function enable
7	Update	R	0	Update: Set to 1 when writing to the ADD bit; Cleared when the address takes effect (at the end of transmission);
6: 0	ADD	RW	0	ADD: Function Address

34.5.2. USB Interrupt Status Register (USB_INTR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res										EP5IN	EP4IN	EP3IN	EP2IN	EP1IN	EP0
-										R	R	R	R	R	R
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res		EP5 OUT	EP4 OUT	EP3 OUT	EP2 OUT	EP1 OUT	Res					SOF	Reset	Re- sume	Sus- pend
-		R	R	R	R	R	-					R	R	R	R

Bit	Name	R/W	Reset Value	Function
31: 22	Reserved	-	-	Reserved
21	EP1IN	R	0	IN endpoint 1 interrupt
20	EP4IN	R	0	IN endpoint 4 interrupt
19	EP3IN	R	0	IN endpoint 3 interrupt
18	EP2IN	R	0	IN endpoint 2 interrupt
17	EP5IN	R	0	IN endpoint 5 interrupt
16	EP0	R	0	Endpoint 0 interrupt
15: 14	Reserved	-	-	Reserved
13	EP1OUT	R	0	OUT endpoint 1 interrupt
12	EP4OUT	R	0	OUT endpoint 4 interrupt
11	EP3OUT	R	0	OUT endpoint 3 interrupt
10	EP2OUT	R	0	OUT endpoint 2 interrupt
9	EP5OUT	R	0	OUT endpoint 5 interrupt
8: 4	Reserved	-	-	Reserved
3	SOF	R	0	SOF interrupt, set to 1 at the start of each frame.
2	Reset	R	0	Reset interrupt, set to 1 when a reset signal is detected on the USB bus.

Bit	Name	R/W	Reset Value	Function
1	Resume	R	0	Resume interrupt, set to 1 when a Resume signal is detected on the USB bus while the USB device is in Suspend mode.
0	Suspend	R	0	Suspend interrupt, set to 1 when a Suspend signal is detected on the USB bus.

34.5.3. USB Interrupt Enable Register (USB_INTRE)

Address offset: 0x08

Reset value: 0x003F 3E06

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res										EP5INE	EP4INE	EP3INE	EP2INE	EP1INE	EP0E
-										RW	RW	RW	RW	RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res		EP5 OUT	EP4 OUT	EP3 OUT	EP2 OUT	EP1 OUT	Res					SOFE	ResetE	ResumeE	Sus- pendE
-		RW	RW	RW	RW	RW	-					RW	RW	RW	RW

Bit	Name	R/W	Reset Value	Function
31: 22	Reserved	-	-	Reserved
21	EP1INE	RW	1	IN Endpoint 1 Interrupt Enable
20	EP4INE	RW	1	IN Endpoint 4 Interrupt Enable
19	EP3INE	RW	1	IN Endpoint 3 Interrupt Enable
18	EP2INE	RW	1	IN Endpoint 2 Interrupt Enable
17	EP5INE	RW	1	IN Endpoint 5 Interrupt Enable
16	EP0E	RW	1	Endpoint 0 Interrupt Enable
15: 14	Reserved	-	-	Reserved
13	EP1OUTE	RW	1	OUT Endpoint 1 Interrupt Enable
12	EP4OUTE	RW	1	OUT Endpoint 4 Interrupt Enable
11	EP3OUTE	RW	1	OUT Endpoint 3 Interrupt Enable
10	EP2OUTE	RW	1	OUT Endpoint 2 Interrupt Enable
9	EP5OUTE	RW	1	OUT Endpoint 5 Interrupt Enable
8: 4	Reserved	-	-	Reserved
3	SOFE	RW	0	SOF Interrupt Enable
2	ResetE	RW	1	Reset Interrupt Enable
1	ResumeE	RW	1	Resume Interrupt Enable
0	SuspendE	RW	0	Suspend Interrupt Enable

34.5.4. USB Frame Register (USB_FRAME)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res												INDEX			
-												RW			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res						FrameNum									
-						R									

Bit	Name	R/W	Reset Value	Function
31: 20	Reserved	-	-	Reserved
19: 16	INDEX	RW	0	Endpoint Selection
15: 11	Reserved	-	0	Reserved
10: 0	FRAMENUM	R	0	Last received frame number

34.5.5. USB Endpoint 0 Control Register (USB_EP0CSR)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res															
-															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	COUNT0							Serviced-SetupEnd	ServicedOut-PktRdy	Send-Stall	Set-upEnd	Da-taEnd	Sent-Stall	InPk-tRdy	Out-PktRdy
-	R							W	W	W	R	W	RW0	RS	R

Bit	Name	R/W	Reset Value	Function
31: 15	Reserved	-	-	Reserved
14: 8	COUNT0	R	0	Length of data received by EP0, valid when Out-PktRdy is set to 1
7	ServicedSetupEnd	W	0	Writing 1 by software clears SetupEnd, this bit is automatically cleared
6	ServicedOutPktRdy	W	0	Writing 1 by software clears OutPktRdy, this bit is automatically cleared

5	SendStall	W	0	Writing 1 by software terminates the current transfer; A STALL handshake will be sent, then this bit is automatically cleared;
4	SetupEnd	R	0	At the end of control transfer, setting DataEnd to 1 in advance will generate an interrupt and clear the FIFO;
3	DataEnd	W	0	Software writes 1 under the following conditions, this bit is automatically cleared; 1. After sending the last data packet and setting InPktRdy; 2. After the last data packet is read and OutPktRdy is cleared by software; 3. Send a zero-length packet, after setting InPktRdy;
2	SentStall	RC_W0	0	Set to 1 after sending a STALL handshake, cleared by software.
1	InPktRdy	RS	0	Software writes 1 after writing a data packet into the FIFO. Cleared after the data packet transmission is completed, generating an interrupt upon clearing.
0	OutPktRdy	R	0	This bit is set to 1 after receiving a data packet and generates an interrupt.

34.5.6. USB IN Endpoint Control Register (USB_INEPxCSR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res								InMaxP							
-								RW							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	ClrData Tog	Sent Stall	Send Stall	Flush FIFO	Un- der- Run	FIFONotE mpty	InPk- tRdy	Au- toSet	ISO	Mod e	DMAE- nab	FrcD ata- Tog	Res		
-	W	RC_ W0	RW	W	RC_W 0	RC_W0	RS	RW	RW	RW	RW	RW	-		

Bit	Name	R/W	Reset Value	Function
31: 24	Reserved	-	-	Reserved

23: 16	InMaxP	RW	0	Maximum packet length for IN endpoint, in units of 8 Bytes; each IN endpoint has one, except for EP0.
15	Reserved	-	-	Reserved
14	ClrDataTog	W	0	Software writes 1 to reset IN EP data toggle to 0.
13	SentStall	RC_W0	0	Set to 1 after sending a STALL handshake packet; at this time, the FIFO must be cleared, InPktRdy must also be cleared, and this bit is cleared by software.
12	SendStall	RW	0	Software writes 1, sends a STALL handshake after receiving an IN Token; Software clears this bit to terminate Stall transmission; this bit is invalid for ISO.
11	FlushFIFO	W	0	Software writes 1 to clear IN FIFO; only clears the next packet to be transmitted. If there are two packets in FIFO, it needs to be written twice. This bit is automatically cleared.
10	UnderRun	RC_W0	0	In ISO mode, this bit is set when a zero-length packet is sent after receiving an IN Token and InPktRdy is not set to 1; in Bulk and Int modes, this bit is set when the USB device responds with a NAK after receiving an IN Token. This bit is cleared by software.
9	FIFONotEmpty	RC_W0	0	IN FIFO not empty flag
8	InPktRdy	RS	0	After writing a data packet to FIFO, software writes 1; cleared after packet transmission completes, and an interrupt is generated upon clearing.
7	AutoSet	RW	0	After setting to 1, when data written to IN FIFO reaches the maximum packet size (InMaxP), InPktRdy is automatically set to 1.
6	ISO	RW	0	Set to 1 to enable ISO transfer, clear for Bulk or Interrupt transfer
5	Mode	RW	0	1: IN endpoint; 0: OUT endpoint;
4	DMAEnab	RW	0	IN endpoint DMA request enable
3	FrcDataTog	RW	0	When set to 1, regardless of whether ACK is received, forcibly toggle the data toggle signal and clear the packet in the FIFO
2: 0	Reserved	-	-	Reserved

34.5.7. USB OUT endpoint control register (USB_OUTEPxCSR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res								OutMaxP							
-								RW							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ClrData-Tog	Sent-Stall	Send-Stall	Flush-FIFO	DataError	Over-Run	FIFOFull	Out-PktRdy	Auto-Clear	ISO	DMA-Mode	Res				
W	RC_W0	RW	W	R	RC_W0	R	RC_W0	RW	RW	RW	RW	-			

Bit	Name	R/W	Reset Value	Function
31: 24	Reserved	-	-	Reserved
23: 16	OutMaxP	RW	0	Maximum packet size for OUT endpoint, each OUT endpoint has one, except EP0;
15	ClrDataTog	W	0	Writing 1 resets EP's data toggle to 0;
14	SentStall	RC_W0	0	Set to 1 after STALL handshake is sent; Cleared by software;
13	SendStall	RW	0	Writing 1 sends STALL handshake, clearing by software ends STALL; Invalid in ISO mode;
12	FlushFIFO	W	0	Clear OUT FIFO, writing once clears one packet of data;
11	DataError	R	0	After OutPktRdy is set to 1, the data packet has a CRC error or bit-stuff error; automatically cleared after OutPktRdy is cleared; only valid in ISO mode;
10	OverRun	RC_W0	0	OUT packets cannot be written to OUT FIFO, cleared by software; Only valid in ISO mode;
9	FIFOFull	R	0	OUT FIFO full flag
8	OutPktRdy	RC_W0	0	Set to 1 upon receiving a data packet; cleared by software after data is read from FIFO, which generates an interrupt;

Bit	Name	R/W	Reset Value	Function
7	AutoClear	RW	0	When set to 1, OutPktRdy automatically clears after data read from OUT FIFO reaches Out-MaxP value
6	ISO	RW	0	1: ISO mode; 0: Bulk or Interrupt mode;
5	DMAEnab	RW	0	DMA Enable
4	DMAMode	RW	0	0: All received packets generate DMA request and interrupt; 1: Generates DMA request upon receiving Out-MaxP data, no interrupt; For other packet sizes, generates interrupt but no DMA request;
3: 0	Reserved	-	-	Reserved

34.5.8. USB OUT Endpoint Count Register (USB_OUTCOUNT)

Address offset: 0x1C

Reset value: 0x0000 0000

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

Bit	Name	R/W	Reset Value	Function
31: 11	Reserved	-		Reserved
10: 0	OUTCOUNT	R	0	Received data length, valid when OutPktRdy is set to 1

34.5.9. USB FIFO Register (USB_FIFO)

Address offset: 0x20-0x33

Reset value: 0x0000 0000

Note: FIFO can only operate by word or byte

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

Bit	Name	R/W	Reset Value	Function
31: 0	FIFODATA	-	-	FIFODATA (0~5, each EP occupies 4 Byte addresses)

35. Debug Support

35.1. Overview

This chip is based on the Cortex-M0+ CPU, which includes advanced debug hardware extension features. The hardware debug module allows the core to stop during instruction fetch (instruction breakpoint) or data access (data breakpoint). When the core is stopped, both its internal state and the system's external state can be queried. After the query is completed, the core and peripherals can be restored, and the program will continue to execute.

Debug functions are used by the debug host when connecting and debugging the MCU, with the debug interface being serial wire. The debug function in the M0+ CPU Core is an ARM CoreSight Design kit.

M0+ provides integrated on-chip debug support, consisting of:

- SW-DP: Serial wire
- BPU: Break point unit
- DWT: Data watchpoint trigger

Debug support also includes the chip's integrated debug features:

- Flexible debug pin assignment, SWDIO@PA13, SWCLK@PA14
- MCU Debug Box (supports low-power mode, controls peripheral clocks, etc.)

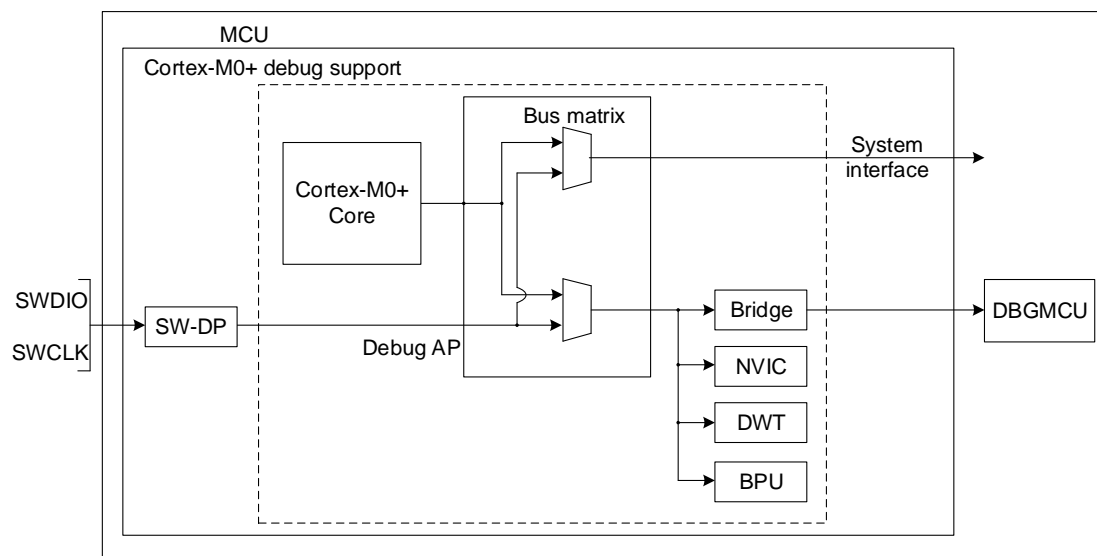


Figure 35-1 DBG Block Diagram

35.2. Pin distribution and debug port pins

35.2.1. SWD Debug Port

There are two debug function related ports, visible in all package forms.

Table 35-1 DBG Block Diagram

SW-DP Port pin name	SW debug interface		Pin assignment
	Type	Debug function	
SWDIO	Input/output	Serial data input/output	PA13
SWCLK	Input	Serial clock	PA14

35.2.2. Flexible SW-DP pin assignment

After chip reset (system reset or power-on reset), the ports used for SW-DP are assigned as dedicated pins for immediate use by the debug host.

However, the chip provides a method to disable the SWD port and release it for GPIO use.

35.2.3. Internal pull-up and pull-down on SWD pins

Once the SWD port is released by software, the GPIO controller takes over these two ports. The reset state of the GPIO control register sets the IOs to the same state:

- SWDIO: input pull-up
- SWCLK: input pull-down

The on-chip pull-up and pull-down resistors eliminate the need for external resistors.

35.3. ID code and locking mechanism

The chip stores the ID code. It is recommended that tools like Keil and IAR use this ID Code (located at address 0x4001 5800) to lock debugging.

After power-on, the hardware reads the reserved area of Flash at address 0x1FFF 33F0 and loads it into the DBG_IDCODE register.

35.4. SWD Debug Port

35.4.1. SWD Protocol Introduction

This is a synchronous serial communication protocol using the following two ports:

- SWCLK: Clock signal from host to chip
- SWDIO: Bidirectional data signal

This protocol allows two banks of registers (DPACC registers and APACC registers) to be read and written. Data bits are transmitted LSB-first on the line. For bidirectional management of SWDIO, the line must be pulled up on the board level (recommended 100k ohm resistor).

In the protocol, every time the SWDIO direction changes, a turnaround time is inserted on the line when neither the host nor the chip is driving it. By default, this turnaround time is 1 bit time, but it can be adjusted by configuring the SWCLK frequency.

35.4.2. SWD Protocol Sequence

Each sequence consists of the following phases:

- Host's packet request (8 bits)

- Chip's acknowledgment response (3 bits)
- Data transmission phase from host or chip (33 bits)

Table 35-2 Request Packet (8 bits)

Bit	Name	Description
0	Start	Must be "1"
1	ApnDP	0: DP access 1: AP access
2	RnW	0: Write request 1: Read request
4:3	A[3:2]	Address region of DP or AP register
5	Parity	Parity bit of previous bits
6	Stop	0
7	Park	Not driven by the host. Due to the pull-up property, it will be read as 1 by the chip.

The normal turnaround time (default is 1 bit) follows the packet request, during which neither the host nor the chip drives the signal line.

Table 35-3 ACK Response (3 bits)

Bit	Name	Description
[2:0]	ACK	001: FAULT 010: WAIT 100: OK

For a read operation or if a WAIT or FAULT response is received, the turnaround time must follow the ACK response.

Table 35-4 DATA transfer (33 bits)

Bit	Name	Description
[31:0]	WDATA or RDATA	Write or read data
32	Parity bit	Parity bit for [31:0]

For a read operation, the turnaround time must follow the data transfer.

35.4.3. SW-DP state machine (reset, idle states, ID code)

The SW-DP state machine has a defined internal ID code for the SW-DP. It complies with the JEP-106 standard. This ID code is the default ARM code and is set to 0x0BC11477 (for Cortex-M0+).

35.4.4. DP and AP read/write accesses

- Read DP operations are not posted: the chip response can be immediate (ACK=OK) or delayed (ACK=WAIT).

- Read AP operations are posted: this means the access result is returned in the next transfer. If the next access is not an AP access, the DP-RDBUFF register must be explicitly read to obtain the result.

The READOK flag in the DP-CTRL/STAT register is updated during each AP read access or RDBUFF read request (to determine whether the AP read access was successful).

- The SW-DP implements a write buffer (for DP and AP writes), which can receive a write operation even when other operations are still pending. If the write buffer is full, the chip response is "WAIT". IDCODE read, CTRL/STAT read, or ABORT write are exceptions (even if the write buffer is full).
- Since SWCLK and HCLK are asynchronous clocks, two additional SWCLK cycles are required after a write operation (after the parity bit) to ensure internal write validity. These cycles should be applied when driving the signal line low.

This is especially important when writing CTRL/STAT for a power-up request. If the next operation (requiring power-up) occurs immediately, it will fail.

35.4.5. SW-DP registers

These registers can be accessed when ApnDP=0.

A[3:2]	R/W	CTRLSEL bit or SELECT register	Register
00	Read		IDCODE
00	Write		ABORT
01	Read/Write	0	DP-CTRL/STAT
01	Read/Write	1	WIRE CONTROL
10	Read		READ RESEND
10	Write		SELECT
11	Read/Write		READ BUFFER

35.4.6. SW-AP register

Address	A[3:2]	Description
0x0	00	Reserved
0x4	01	DP CTRL/STAT register, used as <ul style="list-style-type: none"> ■ Request a system or debug power-up ■ Configure transport operations for AP access ■ Control pushed comparison and pushed verification operations ■ Read some status flags (overflow, power-up acknowledgment)
0x8	10	DP SELECT Register: Used to select the current access port and the active 4-word register window. <ul style="list-style-type: none"> ■ Bit 31:24: APSEL: Selects the current AP

		<ul style="list-style-type: none"> ■ Bit 23:8: Reserved ■ Bit 7:4: APBANKSEL: Selects the active 4-word register window within the current AP ■ Bit 3:0: Reserved
0xC	11	DP RDBUFF Register: Used to provide the debugger with the final result after an operation sequence (without requesting a new JTAG-DP operation)

35.5. Core Debug

Core debug can be accessed through core debug registers. Debug accesses these registers through the debug access port. It consists of the following four registers

Table 35-5 Core Debug Registers

Register	Description
DHCSR	32-bit Debug Halt Control and Status Register
DCRSR	17-bit Debug Core Register Selector Register
DHCDR	32-bit Debug Core Register Data Register
DEMCR	32-bit Debug Exception and Monitor Control Register

These registers are not affected by system reset. They can only be reset by power-on reset. To halt (Hart) during reset, the following must be done:

- Enable bit 0 (VC_CORRESET) of the Debug and Exception Monitor Control Register.
- Enable bit 0 (C_DEBUGEN) of the Debug Halt Control and Status Register

35.6. BPU Breakpoint Unit (Break Point Unit)

The Cortex-M0+ BPU implementation provides 4 breakpoint registers. The BPU is a set of ARMv7-M Flash Patch and Breakpoint (FPB) Block (Cortex-M3 & Cortex-M4).

35.6.1. BPU Functionality

The processor breakpoint implementation is based on PC breakpoint functionality.

Refer to the ARMv6-M ARM and ARM Coresight Components Technical Reference Manual for more details about BPU Coresight identity registers, their addresses, and access types.

35.7. Data Watchpoint DWT (Data Watchpoint)

The Cortex-M0 DWT implementation provides 2 watchpoint registers.

35.7.1. DWT Functionality

The processor's breakpoint implementation is based on PC breakpoint functionality.

35.7.2. DWT Program Counter Sample Register

Processors implementing the data watchpoint unit also implement the ARMv6-M optional DWT Program Counter Sample register (DWT_PCSR). This register allows debuggers to periodically sample the PC without stopping the processor. This mechanism provides coarse-grained analysis.

CORTEX-M0+ DWT_PCSR records instructions that passed and failed the condition code.

35.8. MCU Debug Module (DBGMCU)

The MCU debug component provides the following support for debuggers:

- Low-power mode
- Clock control for timer, watchdog, I2C, CAN, and RTC during breakpoints
- Control over trace pin allocation

The MCUIDBG register also provides the chip ID code. This ID code can be accessed using JTAG or SW debug interfaces, or by user programs.

35.8.1. Debug Support for Low-Power Modes

To enter low-power mode, execute the WFI or WFE instruction. The MCU enters low-power mode either by stopping the CPU Clock or reducing CPU power consumption.

The CPU is not allowed to stop FCLK or HCLK during debugging. Since these are required for debugger connections, they must remain enabled during debugging. The MCU integrates special methods to allow users to debug software in low-power modes.

Therefore, the debugger host must first set the contents of certain debug configuration registers to alter low-power behavior:

- In Sleep mode: FCLK and HCLK remain active. Accordingly, this mode should not impose any limitations on standard debugging functions.
- In Stop mode: The DBG_STOP bit must be set by the debugger in advance.

35.8.2. Supports debugging for timers, watchdogs, CAN, and I2C.

During a breakpoint, it is necessary to select how the timer counter and watchdog should behave:

- They can continue counting inside a breakpoint. For example, this is often needed when a PWM is controlling a motor.
- They can stop counting inside a breakpoint. This is determined by the watchdog's characteristics.

For CAN, users can choose to block updates to the receive register during breakpoints.

For I2C, users can choose to block SMBUS timeout during breakpoints.

35.9. DBG Register

35.9.1. DBG Device ID Code Register (DBG_IDCODE)

Address offset: 0x00

Supports only 32-bit address access, read-only.

This register can be accessed via the software debug port (2-pin) or user software.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DBG_IDCODE[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
DBG_IDCODE[16:0]															
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Bit	Name	R/W	Reset Value	Function
31: 0	DBG_IDCODE[31:0]	R	0x0618 8061	Revision ID

35.9.2. Debug MCU Configuration Register (DBGMCU_CR)

This register configures the MCU low-power mode in debug state.

This register is asynchronously reset by power-on reset (not system reset). It can be written by the debugger during system reset.

If the debug host does not support this feature, it is still possible for software users to write to these registers.

Address offset: 0x04

Reset value: 0x0000 0000 (not reset by system reset)

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

Bit	Name	R/W	Reset Value	Function
31: 2	Reserved	-		Reserved
1	DBG_STOP	RW	0	Debug stop mode. 0: (FCLK=off, HCLK=off). In Debug stop mode, both HCLK and FCLK are turned off. When exiting Debug stop mode, the clock configuration is the same as after power-

				<p>on reset (system clock is HSI). Subsequently, the software must reconfigure the clock controller.</p> <p>1: (FCLK=on, HCLK=on). When entering Debug stop mode, HSI remains on, and FCLK and HCLK are generated by HSI. When exiting Debug stop mode, if clock control needs to be changed, the software must reconfigure it.</p>
0	DBG_SLEEP	RW	0	<p>Debug sleep mode.</p> <p>0: (FCLK on, HCLK off). In Debug sleep mode, FCLK is provided by the pre-configured system clock, while HCLK is turned off. Since Debug sleep mode does not reset the configured clock system, software does not need to reconfigure the clock after exiting Debug sleep mode.</p> <p>1: (FCLK on, HCLK on). In SLEEP mode, both FCLK and HCLK clocks are provided by the pre-configured system clock.</p>

35.9.3. DBG APB Freeze Register 1 (DBG_APB_FZ1)

This register is used to configure the clocks of timer, RTC, IWDG, and WWDG during debug. This register is asynchronously reset by power-on reset (not system reset). It can be written by the debugger during system reset.

Address offset: 0x08

Power on Reset value: 0x0000 0000

31	3	2	28	27	26	2	2	2	22	21	20	19	1	17	16
0	9					5	4	3					8		
DBG_LPTIM_STOP	Res								DBG_I2C2_SMBUS_TIMEOUT	DBG_I2C1_SMBUS_TIMEOUT	Res	DBG_CAN_STOP	Res		
RW	-								R	RW	-	RW	-		
15	1	1	12	11	10	9	8	7	6	5	4	3	2	1	0
4	3														
Res	DBG_IWDG_STOP	DBG_WWDG_STOP	DBG_RTC_STOP	Res		Res	DBG_TIM7_STOP		DBG_TIM6_STOP	Res		DBG_TIM3_STOP	DBG_TIM2_STOP		
-	RW	RW	RW	-		-	RW		RW	-		RW	RW		RW

Bit	Name	R/W	Reset Value	Function
31	DBG_LPTIM_STOP	RW	0	<p>When the CPU core is in halt state, the counter clock control bit of LPTIM</p> <p>0: Enabled</p> <p>1: Disabled</p>

Bit	Name	R/W	Reset Value	Function
30: 23	Reserved	-	-	Reserved
22	DBG_I2C2_SMBUS_TIMEOUT	R	0	Fixed at 0
21	DBG_I2C1_SMBUS_TIMEOUT	RW	0	Controls whether I2C1 SMBUS timeout is frozen when the CPU core is in halt state. 0: Processed the same as normal mode; 1: SMBUS timeout counter frozen.
20	Reserved	-	-	Reserved
19	DBG_CAN_STOP	RW	0	Controls CAN stop when CPU core is in halt state. 0: CAN operates same as normal mode when CPU is halted; 1: CAN receive register disabled when CPU is halted;
18: 13	Reserved	-	-	Reserved
12	DBG_IWDG_STOP	RW	0	Clock control bit for IWDG counter when CPU core is in halt state 0: Enabled 1: Disabled
11	DBG_WWDG_STOP	RW	0	Clock control bit for WWDG counter when CPU core is in halt state 0: Enabled 1: Disabled
10	DBG_RTC_STOP	RW	0	When the CPU core is in halt state, the clock control bit of the RTC counter 0: Enabled 1: Disabled
9: 6	Reserved	-	-	Reserved
5	DBG_TIM7_STOP	RW	0	When the CPU core is in halt state, control the counting clock of TIM7. 0: Clock enabled 1: Clock disabled
4	DBG_TIM6_STOP	RW	0	When the CPU core is in halt state, control the counting clock of TIM6. 0: Clock enabled 1: Clock disabled
3: 2	Reserved	-	-	Reserved
1	DBG_TIM3_STOP	RW	0	Clock control bit for TIM3 counter when the CPU core is in halt state 0: Enabled

Bit	Name	R/W	Reset Value	Function
				1: Disabled
0	DBG_TIM2_STOP	RW	0	Controls the counting clock of TIM2 when the CPU core is in halt state. 0: Clock enabled 1: Clock disabled

35.9.4. DBG APB Freeze Register 2 (DBG_APB_FZ2)

This register is used to configure the timer's clock control in debug mode. This register is asynchronously reset by power-on reset (not system reset). It can be written by the debugger during system reset.

Address offset: 0x0C

Power on Reset value: 0x0000 0000

Supports only 32-bit address access, read-only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res													DBG_ TIM17_ STOP	DBG_ TIM16_STO P	DBG_ TIM15_STO P
-													RW	RW	RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DBG_ TIM14_STO P	Res			DBG_ TIM1_S TOP	Res										
RW	-			RW	-										

Bit	Name	R/W	Reset Value	Function
31: 19	Reserved	-		Reserved
18	DBG_TIM17_STOP	RW	0	When the CPU core is in halt state, the clock control bit of TIM17 counter 0: Enabled 1: Disabled
17	DBG_TIM16_STOP	RW	0	When the CPU core is in halt state, the clock control bit of TIM16 counter 0: Enabled 1: Disabled
16	DBG_TIM15_STOP	RW	0	Controls the counting clock of TIM15 when CPU core is in halt state. 0: Clock enabled 1: Clock disabled
15	DBG_TIM14_STOP	RW	0	Clock control bit for TIM14 counter when CPU core is in halt state

Bit	Name	R/W	Reset Value	Function
				0: Enabled 1: Disabled
14: 12	Reserved	-		Reserved
11	DBG_TIM1_STOP	RW	0	Clock control bit for TIM1 counter when CPU core is in halt state 0: Enabled 1: Disabled
10: 0	Reserved	-		Reserved

36. Version History

Version	Date	Updated Record
V0.2	2025.05.30	Initial version



Puya Semiconductor Co., Ltd.

IMPORTANT NOTICE

Puya reserve the right to make changes, corrections, enhancements, modifications to Puya products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information of Puya products before placing orders.

Puya products are sold pursuant to terms and conditions of sale in place at the time of order acknowledgement.

Purchasers are solely responsible for the choice and use of Puya products. Puya does not provide service support and assumes no responsibility when products that are used on its own or designated third party products.

Puya hereby disclaims any license to any intellectual property rights, express or implied.

Resale of Puya products with provisions inconsistent with the information set forth herein shall void any warranty granted by Puya.

Any with Puya or Puya logo are trademarks of Puya. All other product or service names are the property of their respective owners.

The information in this document supersedes and replaces the information in the previous version.

Puya Semiconductor Co., Ltd.– All rights reserved