

# W55MH32 Reference Manual

Version 1.0.1

# CONTENTS

<b>1</b>	<b>Abbreviations in The Text.....</b>	<b>30</b>
<b>2</b>	<b>Memory and Bus Architecture .....</b>	<b>31</b>
2.1	System Framework .....	31
2.2	Memory Organization .....	32
2.3	Memory Image.....	32
2.3.1	Embedded SRAM .....	34
2.3.2	Bandwidth .....	34
2.3.3	Embedded Flash .....	34
2.4	Startup Configuration .....	36
<b>3</b>	<b>CRC Calculation Unit.....</b>	<b>38</b>
3.1	Introduction to CRC .....	38
3.2	CRC Key Features.....	38
3.3	CRC Functional Description .....	38
3.4	CRC Register.....	39
3.4.1	Data Register (CRC_DR) .....	39
3.4.2	Independent Data Register (CRC_IDR) .....	39
3.4.3	Control Register (CRC_CR) .....	39
3.4.4	Control Status Extension Register (CRC_CSR).....	40
3.4.5	Initial Value Expansion Register (CRC_INI) .....	40
3.4.6	Result XOR Register (CRC_XOR) .....	41
3.4.7	CRC Register Image .....	41
<b>4</b>	<b>Power Control (PWR) .....</b>	<b>42</b>
4.1	Power Supply .....	42
4.1.1	Separate A/D Converter Supply and Reference Voltage .....	42
4.1.2	Battery Backup Area .....	42
4.1.3	Voltage Regulator .....	43
4.2	Power Manager.....	43
4.2.1	Power-On Reset (POR) and Power-Down Reset (PDR) .....	43
4.2.2	Programmable Voltage Monitor (PVD).....	44
4.3	Low Power Mode .....	45
4.3.1	Reduced System Clock.....	45
4.3.2	Control of External Clock .....	45
4.3.3	Sleep Mode .....	45
4.3.4	Stop Mode.....	46
4.3.5	Standby Mode .....	47
4.3.6	Automatic Wake-up in Low-Power Mode (AWU) .....	48
4.4	Power Control Register .....	48
4.4.1	Power Control Register (PWR_CR) .....	48
4.4.2	Power Control/Status Register (PWR_CSR).....	49
4.4.3	PWR Register Address Image.....	50
<b>5</b>	<b>Backup Register (BKP).....</b>	<b>51</b>
5.1	Introduction to BKP .....	51
5.2	BKP Characteristics.....	51
5.3	BKP Functional Description .....	51
5.3.1	Intrusion Detection.....	51



5.3.2	RTC Calibration .....	51
5.4	BKP Register Description .....	52
5.4.1	Backup Data Register x (BKP_DRx) (x=1..10) .....	52
5.4.2	RTC Clock Calibration Register (BKP_RTCCR).....	52
5.4.3	Backup Control Register (BKP_CR).....	52
5.4.4	Backup Control/Status Register (BKP_CSR) .....	53
5.4.5	BKP Register Image.....	53
<b>6</b>	<b>Reset and Clock Control (RCC) .....</b>	<b>56</b>
6.1	Reset (a Dislocated Joint, an Electronic Device etc) .....	56
6.1.1	System Reset .....	56
6.1.2	Power Reset .....	56
6.1.3	Backup Domain Reset.....	57
6.2	Clocks.....	57
6.2.1	HSE Clock .....	59
6.2.2	HSI Clock.....	59
6.2.3	PLL .....	60
6.2.4	LSE Clock.....	60
6.2.5	LSI Clock .....	60
6.2.6	System Clock (SYSCLK) Selection .....	61
6.2.7	Clock Safety System (CSS) .....	61
6.2.8	RTC Clock.....	61
6.2.9	Watchdog Clock.....	61
6.2.10	Clock Output .....	62
6.3	RCC Register Description .....	62
6.3.1	Clock Control Register (RCC_CR).....	62
6.3.2	Clock Configuration Register (RCC_CFGR).....	63
6.3.3	Clock Interrupt Register (RCC_CIR).....	65
6.3.4	APB2 Peripheral Reset Register ( RCC_APB2RSTR ) .....	66
6.3.5	APB1 Peripheral Reset Register (RCC_APB1RSTR).....	67
6.3.6	AHB Peripheral Clock Enable Register (RCC_AHBENR).....	69
6.3.7	APB2 Peripheral Clock Enable Register (RCC_APB2ENR) .....	70
6.3.8	APB1 Peripheral Clock Enable Register (RCC_APB1ENR) .....	71
6.3.9	Backup Domain Control Register (RCC_BDCR) .....	72
6.3.10	Control/Status Register (RCC_CSR).....	73
6.3.11	Control/Status Register (RCC_MCO_VAL).....	74
6.3.12	SYSCFG_CONFIG Register (RCC_SYSCFG_CONFIG).....	75
6.3.13	RCC Register Address Image .....	75
<b>7</b>	<b>General Purpose and Multiplexed Function I/O (GPIO and AFIO) .....</b>	<b>77</b>
7.1	GPIO Function Description .....	77
7.1.1	General Purpose I/O (GPIO) .....	78
7.1.2	Individual Bit Set or Bit Clear .....	79
7.1.3	External Interrupt/Wakeup Line .....	79
7.1.4	Multiplexing Function (AF).....	79
7.1.5	Software Remapping of I/O Multiplexing Functions .....	79
7.1.6	GPIO Locking Mechanism .....	79
7.1.7	Input Configuration .....	79
7.1.8	Output Configuration .....	80
7.1.9	Multiplexing Function Configuration .....	80

7.1.10	Analog Input Configuration .....	81
7.1.11	GPIO Configuration for Peripherals.....	82
7.2	GPIO Register Description .....	84
7.2.1	Port Configuration Low Register (GPIOx_CRL) (x=A..G) .....	84
7.2.2	Port Configuration High Register (GPIOx_CRH) (x=A..G) .....	84
7.2.3	Port Input Data Register (GPIOx_IDR) (x=A..G) .....	85
7.2.4	Port Output Data Register (GPIOx_ODR) (x=A..G).....	85
7.2.5	Port Bit Set/Clear Register (GPIOx_BSRR) (x=A..G).....	85
7.2.6	Port Bit Clear Register (GPIOx_BRR) (x=A..G) .....	86
7.2.7	Port Configuration Lock Register (GPIOx_LCKR) (x=A..G).....	86
7.2.8	Forced Pull-Up/Pull-Down Configuration Register (GPIOx_PU_PD_EN) (x=A..G) .....	87
7.2.9	Forced Pull-Up/Pull-Down Lock Register (GPIOx_PU_PD_LCKR) (x=A..G) .....	87
7.3	Multiplexed Functional I/O and Debug Configuration (AFIO) .....	87
7.3.1	OSC32_IN/OSC32_OUT as GPIO Port PC14/PC15 .....	87
7.3.2	Use OSC_IN/OSC_OUT Pins as GPIO Ports PD0/PD1 .....	88
7.3.3	CAN1 Multiplexing Function Remapping .....	88
7.3.4	JTAG/SWD Multiplexing Function Remapping.....	88
7.3.5	ADC multiplexing Function Remapping .....	88
7.3.6	Timer Reuse Function Remapping .....	89
7.3.7	USART Multiplexing Function Remapping.....	90
7.3.8	I2C1 Multiplexing Function Remapping .....	91
7.3.9	SPI1 Multiplexing Function Remapping.....	91
7.3.10	SPI3 Multiplexing Function Remapping.....	91
7.4	AFIO Register Description .....	91
7.4.1	Event Control Register (AFIO_EVCR) .....	91
7.4.2	Multiplexed Remap and Debug I/O Configuration Register (AFIO_MAPR) .....	92
7.4.3	External Interrupt Configuration Register 1 (AFIO_EXTICR1).....	93
7.4.4	External Interrupt Configuration Register 2 (AFIO_EXTICR2).....	94
7.4.5	External Interrupt Configuration Register 3 (AFIO_EXTICR3).....	94
7.4.6	External Interrupt Configuration Register 4 (AFIO_EXTICR4).....	94
7.4.7	AF Remaps and Debugs I/O Configuration Register 2 (AFIO_MAPR2) .....	95
7.5	GPIO and AFIO Register Address Mapping.....	96
<b>8</b>	<b>Interrupts and Events .....</b>	<b>98</b>
8.1	Nested Vector Interrupt Controller .....	98
8.1.1	SysTick Calibration Value Registers .....	98
8.1.2	Interrupt and Exception Vector .....	98
8.2	External Interrupt/Event Controller (EXTI).....	100
8.2.1	Main Characteristics .....	100
8.2.2	Flowchart .....	101
8.2.3	Wake-up Event Management .....	101
8.2.4	Functional Description.....	101
8.2.5	External Interrupt/Event Line Image.....	103
8.3	EXTI Register Description.....	103
8.3.1	Interrupt Mask Register (EXTI_IMR).....	104
8.3.2	Event Mask Register (EXTI_EMR) .....	104
8.3.3	Rising Edge Trigger Select Register (EXTI_RTSTR) .....	104
8.3.4	Falling Edge Trigger Select Register (EXTI_FTSR) .....	105

8.3.5	Software Interrupt Event Register (EXTI_SWIER) .....	105
8.3.6	Pending Register (EXTI_PR) .....	105
8.3.7	External Interrupt/Event Register Image .....	106
<b>9</b>	<b>TCP/IP Offload Engine (TOE) .....</b>	<b>107</b>
9.1	TOE Introduction.....	107
9.2	TOE Key Features.....	107
9.3	Register and Memory Composition .....	107
9.3.1	General Register Area (GRA) .....	108
9.3.2	Socket Register Area .....	109
9.4	Memory .....	110
9.5	Register Description.....	110
9.5.1	General-Purpose Register .....	110
9.5.2	Socket Register .....	116
<b>10</b>	<b>Direct Memory Access controller (DMA) .....</b>	<b>125</b>
10.1	Introduction to DMA.....	125
10.2	DMA Key Features .....	125
10.3	Functional Description .....	126
10.3.1	DMA Processing .....	126
10.3.2	Arbiter.....	127
10.3.3	DMA channel.....	127
10.3.4	Programmable Data Transfer Width, Alignment and Data Size Terminals ....	128
10.3.5	Error Management.....	129
10.3.6	Disruptions.....	129
10.3.7	DMA Request Mapping .....	130
10.4	DMA Registers .....	132
10.4.1	DMA Interrupt Status Register (DMA_ISR).....	132
10.4.2	DMA Interrupt Flag Clear Register (DMA_IFCR) .....	132
10.4.3	DMA Channel x Configuration Register (DMA_CCRx) (x=1..7).....	133
10.4.4	DMA Channel x Transfer Count Register (DMA_CNDTRx) (x=1..7).....	134
10.4.5	DMA Channel x Peripheral Address Register (DMA_CPARx) (x=1..7).....	134
10.4.6	DMA Channel x Memory Address Register (DMA_CMARx) (x=1..7) .....	134
10.4.7	DMA Register Image .....	135
<b>11</b>	<b>Analog/Digital Conversion (ADC) .....</b>	<b>137</b>
11.1	ADC Introduction.....	137
11.2	ADC Key Features.....	137
11.3	ADC Functional Description .....	138
11.3.1	ADC Switch Control .....	139
11.3.2	ADC Clock.....	139
11.3.3	Channel Selection .....	139
11.3.4	Single Conversion Mode .....	139
11.3.5	Continuous Conversion Mode .....	140
11.3.6	Timing Diagram .....	140
11.3.7	Analog Watchdog .....	140
11.3.8	Scanning Mode .....	141
11.3.9	Injection Channel Management .....	141
11.3.10	Intermittent Mode.....	142
11.4	Calibrations.....	143

11.5	Data Alignment .....	143
11.6	Programmable Channel Sampling Time .....	144
11.7	Externally Triggered Conversion .....	144
11.8	DMA Request.....	145
11.9	Dual ADC Mode .....	145
11.9.1	Synchronous Injection Model .....	147
11.9.2	Synchronous Rule Model.....	148
11.9.3	Rapid Crossover Mode .....	148
11.9.4	Slow Crossover Mode.....	149
11.9.5	Alternate Trigger Mode.....	149
11.9.6	Stand-Alone Mode .....	150
11.9.7	Mixed Rule/Injection Synchronization Patterns .....	150
11.9.8	Hybrid Synchronization Rules+ Alternate Trigger Modes.....	150
11.9.9	Hybrid Synchronized Injection+ Cross Mode.....	151
11.10	Temperature Sensor.....	151
11.11	ADC Interrupt.....	152
11.12	ADC Registers.....	153
11.12.1	ADC Status Register (ADC_SR) .....	153
11.12.2	ADC Control Register 1 (ADC_CR1) .....	153
11.12.3	ADC Control Register 2 (ADC_CR2) .....	155
11.12.4	ADC Sample Time Register 1 (ADC_SMPR1).....	157
11.12.5	ADC Sample Time Register 2 (ADC_SMPR2).....	157
11.12.6	ADC Injection Channel Data Offset Register x (ADC_JOFRx) (x=1..4) .....	157
11.12.7	ADC Watchdog High Threshold Register (ADC_HTR) .....	158
11.12.8	ADC Watchdog Low Threshold Register (ADC_LRT) .....	158
11.12.9	ADC Rule Sequence Register 1 (ADC_SQR1) .....	158
11.12.10	ADC Rule Sequence Register 2 (ADC_SQR2) .....	159
11.12.11	ADC Rule Sequence Register 3 (ADC_SQR3) .....	159
11.12.12	ADC Injection Sequence Register (ADC_JSQR) .....	160
11.12.13	ADC Injection Data Register x (ADC_JDRx) (x=1..4) .....	160
11.12.14	ADC Rule Data Register (ADC_DR) .....	160
11.12.15	ADC Register Address Map .....	161
12	<b>Digital-to-Analog Conversion (DAC).....</b>	<b>163</b>
12.1	DAC Introduction.....	163
12.2	DAC Key Features.....	163
12.3	DAC Functional Description .....	164
12.3.1	Enable DAC Channel.....	164
12.3.2	Enable DAC output buffering .....	164
12.3.3	DAC Data Format .....	164
12.3.4	DAC Conversion .....	165
12.3.5	DAC Output Voltage.....	165
12.3.6	Select DAC Trigger .....	165
12.3.7	DMA Request .....	166
12.3.8	Noise Generation .....	166
12.3.9	Triangle Wave Generation .....	167
12.4	Dual DAC Channel Conversion .....	168
12.4.1	Stand-Alone Triggering Without Waveform Generator .....	168
12.4.2	Independent Triggering Using the Same LFSR .....	168

12.4.3	Independent Triggering Using Different LFSRs.....	168
12.4.4	Separate Triggers that Generate the Same Triangle Wave .....	169
12.4.5	Separate triggers for generating different triangle waves.....	169
12.4.6	Simultaneous Software Startup .....	169
12.4.7	Simultaneous Triggering Without Waveform Generator.....	169
12.4.8	Simultaneous Triggering Using the Same LFSR.....	169
12.4.9	Simultaneous Triggering Using Different LFSRs .....	170
12.4.10	Simultaneous Triggering Using the Same Triangle Wave Generator.....	170
12.4.11	Simultaneous Triggering Using Different Triangle Wave Generators.....	170
12.5	DAC Register.....	171
12.5.1	DAC Control Register (DAC_CR) .....	171
12.5.2	DAC Software Trigger Register (DAC_SWTRIGR) .....	172
12.5.3	12-Bit Right-aligned Data hold Register for DAC Channel 1 (DAC_DHR12R1) .	173
12.5.4	12-Bit Left-aligned Data hold Register for DAC Channel 1 (DAC_DHR12L1) ...	173
12.5.5	8-Bit Right-aligned Data hold Register for DAC Channel 1 (DAC_DHR8R1) ....	173
12.5.6	12-Bit Right-aligned Data hold Register for DAC Channel 2 (DAC_DHR12R2) .	174
12.5.7	12-Bit Left-aligned Data hold Register for DAC Channel 2 (DAC_DHR12L2) ...	174
12.5.8	8-Bit Right-aligned Data hold Register for DAC Channel 2 (DAC_DHR8R2) ....	174
12.5.9	12-Bit Right-aligned Data hold Register for Dual DACs (DAC_DHR12RD).....	174
12.5.10	12-Bit Left-aligned Data hold Register for Dual DACs (DAC_DHR12LD) .....	175
12.5.11	8-Bit Right-aligned Data hold Register for Dual DACs (DAC_DHR8RD).....	175
12.5.12	DAC Channel 1 Data Output Register (DAC_DOR1) .....	176
12.5.13	DAC Channel 2 Data Output Register (DAC_DOR2) .....	176
12.5.14	DAC Register Image .....	177
13	<b>Advanced Control Timer (TIM1 and TIM8) .....</b>	<b>178</b>
13.1	Introduction to TIM1 and TIM8.....	178
13.2	TIM1 and TIM8 Main Features .....	178
13.3	TIM1 and TIM8 Functional Description.....	179
13.3.1	Time Base Unit (in computing) .....	179
13.3.2	Counter Mode .....	181
13.3.3	Repetition Counter .....	190
13.3.4	Clock Selection .....	191
13.3.5	Capture/Compare Channel .....	193
13.3.6	Input Capture Mode .....	195
13.3.7	PWM Input Mode .....	195
13.3.8	Forced Output Mode .....	196
13.3.9	Output Comparison Mode .....	196
13.3.10	PWM mode.....	197
13.3.11	Complementary Outputs and Deadband Insertion.....	200
13.3.12	Using the Brake Function.....	201
13.3.13	Clearing the OCxREF Signal on an External Event.....	203
13.3.14	Generates a Six-step PWM Output .....	204
13.3.15	Single Pulse Mode .....	205
13.3.16	Encoder Interface Mode .....	207
13.3.17	Timer Input Heterodyne Function .....	208
13.3.18	Interfacing with Hall Sensors .....	208
13.3.19	Synchronization of TIMx Timers and External Triggers .....	210
13.3.20	Timer Synchronization.....	213

13.3.21	Debug Mode .....	213
13.4	TIM1 and TIM8 Register Descriptions .....	213
13.4.1	TIM1 and TIM8 Control Register 1 (TIMx_CR1) .....	213
13.4.2	TIM1 and TIM8 Control Register 2 (TIMx_CR2) .....	214
13.4.3	TIM1 and TIM8 Slave Mode Control Register (TIMx_SMCR) .....	215
13.4.4	TIM1 and TIM8 DMA/Interrupt Enable Registers (TIMx_DIER).....	217
13.4.5	TIM1 and TIM8 Status Registers (TIMx_SR).....	218
13.4.6	TIM1 and TIM8 Event Generation Registers (TIMx_EGR).....	219
13.4.7	TIM1 and TIM8 Capture/Compare Mode Register 1 (TIMx_CCMR1).....	219
13.4.8	TIM1 and TIM8 Capture/Compare Mode Register 2 (TIMx_CCMR2) .....	222
13.4.9	TIM1 and TIM8 Capture/Compare Enable Registers (TIMx_CCER).....	223
13.4.10	TIM1 and TIM8 Counters (TIMx_CNT) .....	224
13.4.11	TIM1 and TIM8 Prescalers (TIMx_PSC) .....	224
13.4.12	TIM1 and TIM8 Auto-Reload Registers (TIMx_ARR) .....	225
13.4.13	TIM1 and TIM8 Repeat Counter Registers (TIMx_RCR) .....	225
13.4.14	TIM1 and TIM8 Capture/Compare Register 1 (TIMx_CCR1) .....	225
13.4.15	TIM1 and TIM8 Capture/Compare Register 2 (TIMx_CCR2) .....	226
13.4.16	TIM1 and TIM8 Capture/Compare Register 3 (TIMx_CCR3) .....	226
13.4.17	TIM1 and TIM8 Capture/Compare Registers (TIMx_CCR4).....	226
13.4.18	TIM1 and TIM8 Brake and Deadband Registers (TIMx_BDTR) .....	227
13.4.19	TIM1 and TIM8 DMA Control Registers (TIMx_DCR) .....	228
13.4.20	TIM1 and TIM8 Continuous Mode DMA Address (TIMx_DMAR) .....	228
13.4.21	TIM1 and TIM8 Registers.....	230
14	General-Purpose Timer (TIM2 to TIM5) .....	232
14.1	Introduction to TIM2 to TIM5 .....	232
14.2	TIM2 to TIM5 Main Functions.....	232
14.3	TIM2 to TIM5 Functional Description .....	233
14.3.1	Time Base Unit (In Computing).....	233
14.3.2	Counter Mode .....	234
14.3.3	Clock Selection .....	242
14.3.4	Capture/Compare Channel .....	244
14.3.5	Input Capture Mode .....	246
14.3.6	PWM Input Mode .....	247
14.3.7	Forced Output Mode .....	247
14.3.8	Output Comparison Mode .....	248
14.3.9	PWM Mode .....	248
14.3.10	Single Pulse Mode .....	251
14.3.11	Clearing the OCxREF Signal on an External Event.....	252
14.3.12	Encoder Interface Mode .....	252
14.3.13	Timer Input Heterodyne Function .....	254
14.3.14	Synchronization of Timers and External Triggers .....	254
14.3.15	Timer Synchronization.....	257
14.3.16	Debug Mode .....	261
14.4	TIM2 to TIM5 Register Descriptions .....	261
14.4.1	TIM2 to TIM5 Control Register 1 (TIMx_CR1) .....	261
14.4.2	TIM2 to TIM5 Control Register 2 (TIMx_CR2) .....	262
14.4.3	TIM2 to TIM5 Slave Mode Control Register (TIMx_SMCR) .....	263
14.4.4	TIM2 to TIM5 DMA/Interrupt Enable Register (TIMx_DIER).....	264

14.4.5	TIM2 to TIM5 Status Register (TIMx_SR).....	265
14.4.6	TIM2 to TIM5 Event Generation Register (TIMx_EGR).....	266
14.4.7	TIM2 to TIM5 Capture/Compare Mode Register 1 (TIMx_CMR1).....	267
14.4.8	TIM2 to TIM5 Capture/Compare Mode Register 2 (TIMx_CMR2).....	269
14.4.9	TIM2 to TIM5 Capture/Compare Enable Register (TIMx_CCER).....	270
14.4.10	TIM2 to TIM5 Counter (TIMx_CNT).....	271
14.4.11	TIM2 to TIM5 Prescaler (TIMx_PSC).....	271
14.4.12	TIM2 to TIM5 Auto-Reload Register (TIMx_ARR).....	271
14.4.13	TIM2 to TIM5 Capture/Compare Register 1 (TIMx_CCR1).....	271
14.4.14	TIM2 to TIM5 Capture/Compare Register 2 (TIMx_CCR2).....	272
14.4.15	TIM2 to TIM5 Capture/Compare Register 3 (TIMx_CCR3).....	272
14.4.16	TIM2 to TIM5 Capture/Compare Register 4 (TIMx_CCR4).....	272
14.4.17	TIM2 to TIM5 DMA Control Register (TIMx_DCR).....	273
14.4.18	TIM2 to TIM5 Continuous Mode DMA Address (TIMx_DMAR).....	273
14.4.19	TIM2 to TIM5 Registers.....	274
<b>15</b>	<b>General Purpose Timers (TIM9 to TIM14) .....</b>	<b>276</b>
15.1	TIM9 to TIM14 Introduction .....	276
15.2	Main Functions of TIM9 to TIM14.....	276
15.2.1	Main Functions of TIM9/TIM12.....	276
15.2.2	Main Functions of TIM10/TIM11 and TIM13/TIM14 .....	277
15.3	TIM9 to TIM14 Functional Description.....	278
15.3.1	Time Base Unit (in computing).....	278
15.3.2	Counter Mode .....	280
15.3.3	Clock Selection .....	283
15.3.4	Capture/Compare Channel .....	284
15.3.5	Input Capture Mode .....	285
15.3.6	PWM Input Mode (TIM9/12 only) .....	286
15.3.7	Forced Output Mode .....	287
15.3.8	Output Comparison Mode .....	287
15.3.9	PWM Mode .....	288
15.3.10	Single Pulse Mode .....	289
15.3.11	Synchronization of TIM9/12 Timers and External Triggers.....	290
15.3.12	Timer Synchronization (TIM9/12).....	292
15.3.13	Debug Mode .....	292
15.4	TIM9/TIM12 Registers .....	292
15.4.1	TIM9/12 Control Register 1 (TIMx_CR1).....	292
15.4.2	TIM9/12 Slave Mode Control Register (TIMx_SMCR).....	293
15.4.3	TIM9/12 Interrupt Enable Register (TIMx_DIER) .....	294
15.4.4	TIM9/12 Status Register (TIMx_SR) .....	294
15.4.5	TIM9/12 Event Generation Register (TIMx_EGR) .....	295
15.4.6	TIM9/12 Capture/Compare Mode Register 1 (TIMx_CMR1) .....	296
15.4.7	TIM9/12 Capture/Compare Enable Register (TIMx_CCER) .....	298
15.4.8	TIM9/12 Counter (TIMx_CNT).....	298
15.4.9	TIM9/12 Prescaler (TIMx_PSC).....	299
15.4.10	TIM9/12 Automatic Reload Register (TIMx_ARR) .....	299
15.4.11	TIM9/12 Capture/Compare Register 1 (TIMx_CCR1) .....	299
15.4.12	TIM9/12 Capture/Compare Register 2 (TIMx_CCR2) .....	299
15.4.13	TIM9/12 Registers .....	300



15.5	TIM10/11/13/14 Registers .....	301
15.5.1	TIM10/11/13/14 Control Register 1 (TIMx_CR1).....	301
15.5.2	TIM10/11/13/14 Interrupt Enable Register (TIMx_DIER) .....	302
15.5.3	TIM10/11/13/14 Status Register (TIMx_SR) .....	302
15.5.4	TIM10/11/13/14 Event Generation Register (TIMx_EGR) .....	302
15.5.5	TIM10/11/13/14 Capture/Compare Mode Register 1 (TIMx_CMR1) .....	303
15.5.6	TIM10/11/13/14 Capture/Compare Enable Register (TIMx_CCER) .....	304
15.5.7	TIM10/11/13/14 Counters (TIMx_CNT).....	305
15.5.8	TIM10/11/13/14 Prescaler (TIMx_PSC).....	305
15.5.9	TIM10/11/13/14 Automatic Reload Register (TIMx_ARR) .....	305
15.5.10	TIM10/11/13/14 Capture/Compare Register 1 (TIMx_CCR1) .....	306
15.5.11	TIM10/11/13/14 Registers .....	306
<b>16</b>	<b>Basic Timer (TIM6 and TIM7).....</b>	<b>308</b>
16.1	Introduction to TIM6 and TIM7.....	308
16.2	Key Features of TIM6 and TIM7 .....	308
16.3	TIM6 and TIM7 Functions .....	308
16.3.1	Time Base Unit (In Computing).....	308
16.3.2	Counting Mode .....	310
16.3.3	Clock Source .....	312
16.3.4	Debug Mode .....	312
16.4	TIM6 and TIM7 Registers .....	312
16.4.1	TIM6 and TIM7 Control Register 1 (TIMx_CR1) .....	313
16.4.2	TIM6 and TIM7 Control Register 2 (TIMx_CR2) .....	313
16.4.3	TIM6 and TIM7 DMA/Interrupt Enable Register (TIMx_DIER) .....	314
16.4.4	TIM6 and TIM7 Status Registers (TIMx_SR).....	314
16.4.5	TIM6 and TIM7 Event Generation Registers (TIMx_EGR).....	314
16.4.6	TIM6 and TIM7 Counters (TIMx_CNT) .....	315
16.4.7	TIM6 and TIM7 Prescalers (TIMx_PSC) .....	315
16.4.8	TIM6 and TIM7 Auto-Reload Registers (TIMx_ARR) .....	315
16.4.9	TIM6 and TIM7 Registers.....	316
<b>17</b>	<b>Real Time Clock (RTC) .....</b>	<b>317</b>
17.1	Introduction to RTC .....	317
17.2	Main Characteristics.....	317
17.3	Functional Description.....	317
17.3.1	Summarize.....	317
17.3.2	Reset Process .....	318
17.3.3	Read RTC Registers.....	318
17.3.4	Configure RTC Registers.....	319
17.3.5	RTC Flag Setting .....	319
17.4	RTC Register Description .....	319
17.4.1	RTC Control Register High (RTC_CRH) .....	320
17.4.2	RTC Control Register Low (RTC_CRL) .....	320
17.4.3	RTC Prescaler Load Register (RTC_PRLH/RTC_PRLL).....	321
17.4.4	RTC Prescaler Remainder Register (RTC_DIVH/RTC_DIVL) .....	322
17.4.5	RTC Counter Register (RTC_CNTH/RTC_CNTL) .....	322
17.4.6	RTC Alarm Clock Register (RTC_ALRH/RTC_ALRL) .....	323
17.4.7	RTC Register Image .....	324
<b>18</b>	<b>Independent Watchdog Dog (IWDG) .....</b>	<b>325</b>



---

18.1	Introduction .....	325
18.2	IWDG Main Properties .....	325
18.3	IWDG Functional Description .....	325
18.3.1	Hardware Watchdog .....	325
18.3.2	Register Access Protection.....	325
18.3.3	Debug Mode .....	325
18.4	IWDG Register Description .....	326
18.4.1	Key Register (IWDG_KR) .....	326
18.4.2	Prescaler Register (IWDG_PR) .....	327
18.4.3	Reload Register (IWDG_RLR).....	327
18.4.4	Status Register (IWDG_SR) .....	328
18.4.5	IWDG Register Image.....	328
<b>19</b>	<b>Window Watchdog (WWDG) .....</b>	<b>329</b>
19.1	WWDG Profile .....	329
19.2	WWDG Key Features .....	329
19.3	WWDG Functional Description.....	329
19.4	How to Write a Watchdog Timeout Program .....	330
19.5	Debug Mode.....	331
19.6	Register Description.....	331
19.6.1	Control Register (WWDG_CR).....	331
19.6.2	Configuration Register (WWDG_CFR).....	331
19.6.3	Status Register (WWDG_SR) .....	332
19.6.4	WWDG Register Image .....	332
<b>20</b>	<b>True Random Number Generator (TRNG) .....</b>	<b>333</b>
20.1	TRNG Profile.....	333
20.2	Register Description.....	333
20.2.1	Control Status Register (RNG_CSR) .....	333
20.2.2	Data Register (RNG_DR) .....	333
20.2.3	Analog Control Register (RNG_AMA) .....	334
20.2.4	Pseudo-Random Sequence Register (RNG_PN).....	334
20.2.5	FIFO Status Register (RNG_INDEX) .....	334
20.2.6	RNG Register Image .....	335
<b>21</b>	<b>System Configuration Register (SYSCFG) .....</b>	<b>336</b>
21.1	Register Description.....	336
21.1.1	Configure the Lock Register (SYSCFG_LOCK) .....	336
21.1.2	SENSOR Enable Register (SENSOR_EN) .....	336
21.1.3	SENSOR Status Register (SENSOR_STAT) .....	337
21.1.4	SENSOR Voltage Glitch Detection Configuration Register (SENSOR_VG_ACTION) .....	337
21.1.5	Security Algorithm Configuration Register (SSC_CLK_EN).....	338
21.1.6	Security Algorithm Configuration Register (SSC_CLK_EN).....	338
21.1.7	SENSOR Status Register (SENSOR_STATE_FLAG) .....	339
21.1.8	SENSOR Interrupt Status Register (SENSOR_INT_FLAG).....	339
21.1.9	SYSCFG Register Image .....	340
<b>22</b>	<b>One-Time Programmable (OTP) .....</b>	<b>341</b>
22.1	Profile .....	341
22.2	Register Description.....	341

---

---

22.2.1	OTP Data 0 Register (OTP_DATA0).....	341
22.2.2	OTP Data 1 Register (OTP_DATA1).....	341
22.2.3	OTP Data 2 Register (OTP_DATA2).....	341
22.2.4	OTP Data 3 Register (OTP_DATA3).....	342
22.2.5	OTP Data 4 Register (OTP_DATA4).....	342
22.2.6	OTP Data 5 Register (OTP_DATA5).....	342
22.2.7	OTP Data 6 Register (OTP_DATA6).....	342
22.2.8	OTP Data 7 Register (OTP_DATA7).....	343
22.2.9	OTP Configuration Register (OTP_CTRL).....	343
22.2.10	OTP Programming Register (OTP_WR) .....	343
22.2.11	OTP_10ns Register (OTP_10ns).....	344
22.2.12	OTP_LDO Register (OTP_LDO) .....	344
22.2.13	OTP Register Image .....	345
<b>23</b>	<b>SDIO Interface (SDIO) .....</b>	<b>346</b>
23.1	SDIO Main Functions.....	346
23.2	SDIO Bus Topology .....	346
23.3	SDIO Functional Description .....	348
23.3.1	SDIO Adapter .....	349
23.3.2	SDIO AHB Interface.....	356
23.4	Card Function Description.....	356
23.4.1	Card Recognition Mode .....	356
23.4.2	Card Reset .....	356
23.4.3	Confirmation of Operating Voltage Range.....	357
23.4.4	Card Recognition Process .....	357
23.4.5	Write Data Block.....	358
23.4.6	Read Data Block .....	358
23.4.7	Data Stream Operations, Data Stream Writes and Data Stream Reads (Multimedia Cards Only).....	359
23.4.8	Erase: Group Erase and Sector Erase.....	360
23.4.9	Wide Bus Selection and Deselection.....	360
23.4.10	Conservation Management .....	360
23.4.11	Card Status Register .....	363
23.4.12	SD Status Register .....	364
23.4.13	I/O Mode of SD.....	367
23.4.14	Command and Response.....	368
23.5	Response Format .....	371
23.5.1	R1 (Normal Response Command) .....	371
23.5.2	R1b.....	372
23.5.3	R2 (CID, CSD Registers) .....	372
23.5.4	R3 (OCR Register) .....	372
23.5.5	R4 (Fast I/O) .....	372
23.5.6	R4b.....	373
23.5.7	R5 (Interrupt Request).....	373
23.5.8	R6 (Interrupt Request).....	373
23.6	SDIO I/O Card Specific Operations .....	374
23.6.1	SDIO I/O Read Wait Operation using SDIO_D2 Signal Line .....	374
23.6.2	SDIO Read Wait Operation using Stop SDIO_CK .....	374
23.6.3	SDIO Suspend/Resume Operation .....	374

---

23.6.4	SDIO Interrupt.....	375
23.7	CE-ATA Specific Operations .....	375
23.7.1	Command Completion Instructions Close.....	375
23.7.2	Command Completion Indication Enable .....	375
23.7.3	CE-ATA Interrupt .....	375
23.7.4	Suspension of CMD 61 .....	375
23.8	Hardware Flow Control .....	375
23.9	SDIO Register .....	375
23.9.1	SDIO Power Control Register (SDIO_POWER) .....	376
23.9.2	SDIO Clock Control Register (SDIO_CLKCR) .....	376
23.9.3	SDIO Parameter Register (SDIO_ARG) .....	377
23.9.4	SDIO Command Register (SDIO_CMD).....	377
23.9.5	SDIO Command Response Register (SDIO_RESPCMD).....	378
23.9.6	SDIO Response 1..4 Registers (SDIO_RESPx).....	378
23.9.7	SDIO Data Timer Register (SDIO_DTIMER) .....	378
23.9.8	SDIO Data Length Register (SDIO_DLEN) .....	379
23.9.9	SDIO Data Control Register (SDIO_DCTRL).....	379
23.9.10	SDIO Data Counter Register (SDIO_DCOUNT).....	380
23.9.11	SDIO Status Register (SDIO_STA) .....	380
23.9.12	SDIO Clear Interrupt Register (SDIO_ICR).....	381
23.9.13	SDIO Interrupt Mask Register (SDIO_MASK) .....	382
23.9.14	SDIO FIFO Counter Register (SDIO_FIFOCNT).....	384
23.9.15	SDIO Data FIFO Register (SDIO_FIFO).....	384
23.9.16	SDIO Register Image.....	385
<b>24</b>	<b>USB Full Speed Device Interface (USB) .....</b>	<b>386</b>
24.1	Introduction to USB .....	386
24.2	USB Main Features.....	386
24.3	USB Function Description.....	387
24.3.1	USB Function Module Description .....	387
24.4	Issues to Consider in Programming.....	388
24.4.1	System Reset and Power-on Reset.....	388
24.4.2	Double Buffered Endpoints .....	393
24.4.3	Synchronous Transmission.....	394
24.4.4	Suspend/Resume Events.....	395
24.5	USB Register Description .....	396
24.5.1	General-Purpose Register .....	397
24.5.2	Endpoint Registers .....	401
24.5.3	Buffer Description Table .....	404
24.5.4	USB Register Image.....	406
<b>25</b>	<b>Controller Local Area Network (bxCAN).....</b>	<b>408</b>
25.1	Introduction to bxCAN .....	408
25.2	bxCAN Key Features.....	408
25.3	General Description of bxCAN .....	408
25.3.1	CAN2.0B Active Kernel.....	409
25.3.2	Control, Status and Configuration Registers .....	409
25.3.3	Send Email.....	409
25.3.4	Receiving Filter .....	409
25.4	bxCAN operating mode .....	409

---

25.4.1	Initialization Mode .....	409
25.4.2	Normal Mode .....	410
25.4.3	Sleep Mode (Low Power Consumption).....	410
25.5	Test Pattern .....	411
25.5.1	Silent Mode .....	411
25.5.2	Loopback Mode .....	412
25.5.3	Loopback Silent Mode.....	412
25.6	When the W55MH32 is in Debug Mode.....	412
25.7	bxCAN Functional Description .....	412
25.7.1	Send Processing.....	412
25.7.2	Time-triggered Communication Mode.....	414
25.7.3	Reception Management.....	414
25.7.4	Identifier Filtering.....	416
25.7.5	Message Storage .....	419
25.7.6	Error Management.....	420
25.7.7	Bit-Time Characterization .....	421
25.8	bxCAN Interrupt.....	423
25.9	CAN Register Description .....	424
25.9.1	Register Access Protection.....	424
25.9.2	CAN Control and Status Registers.....	424
25.9.3	CAN Mailbox Register .....	431
25.9.4	CAN Filter Register .....	435
25.9.5	bxCAN Register List .....	437
<b>26</b>	<b>Serial Peripheral Interface (SPI) .....</b>	<b>440</b>
26.1	SPI Introduction .....	440
26.2	SPI and I <sup>2</sup> S Key Features .....	440
26.2.1	SPI Features .....	440
26.2.2	I <sup>2</sup> S-Function.....	440
26.3	SPI Functional Description .....	441
26.3.1	Summarize.....	441
26.3.2	Configuring SPI for Slave Mode .....	443
26.3.3	Configuring SPI Master Mode.....	444
26.3.4	Configure SPI for Simplex Communication .....	445
26.3.5	The Process of Sending and Receiving Data.....	445
26.3.6	CRC Calculations.....	450
26.3.7	Status Flag.....	451
26.3.8	Shutdown SPI.....	452
26.3.9	SPI Communication using DMA.....	453
26.3.10	Error Message .....	454
26.3.11	SPI Interrupt.....	455
26.4	I <sup>2</sup> S Functional Description.....	456
26.4.1	I <sup>2</sup> S Functional Description .....	456
26.4.2	Supported Audio Protocols.....	457
26.4.3	Clock Generator .....	462
26.4.4	I <sup>2</sup> S Master Mode .....	463
26.4.5	I <sup>2</sup> S Slave Mode.....	465
26.4.6	Status Flag.....	466
26.4.7	Error Flag .....	467

---

26.4.8	I <sup>2</sup> S Interrupt .....	467
26.4.9	DMA Function .....	467
26.5	SPI and I <sup>2</sup> S Register Descriptions .....	467
26.5.1	SPI Control Register 1 (SPI_CR1) (not used in I <sup>2</sup> S mode) .....	467
26.5.2	SPI Control Register 2 (SPI_CR2) .....	469
26.5.3	SPI Status Register (SPI_SR).....	469
26.5.4	SPI Data Register (SPI_DR) .....	470
26.5.5	SPI CRC Polynomial Register (SPI_CRCPR) (not used in I <sup>2</sup> S mode) .....	470
26.5.6	SPI RxCRC Register (SPI_RXCRCR) (Not Used in I <sup>2</sup> S Mode).....	470
26.5.7	SPI TxCRC Register (SPI_TXCRCR) (Not Used in I <sup>2</sup> S Mode).....	471
26.5.8	SPI_I <sup>2</sup> S Configuration Register (SPI_I2SCFGR) .....	471
26.5.9	SPI_I <sup>2</sup> S Prescaler Register (SPI_I2SPR).....	472
26.5.10	Register Address Map .....	473
<b>27</b>	<b>Inter-Integrated Circuit(I<sup>2</sup>C) Interface .....</b>	<b>474</b>
27.1	Introduction to I <sup>2</sup> C.....	474
27.2	I <sup>2</sup> C Main Features .....	474
27.3	I <sup>2</sup> C Functional Description.....	474
27.3.1	Mode Selection.....	474
27.3.2	I <sup>2</sup> C Slave Mode .....	476
27.3.3	I <sup>2</sup> C Master Mode.....	478
27.3.4	Error Condition (math.) .....	480
27.3.5	SDA/SCL Line Control .....	481
27.3.6	SMBus.....	481
27.3.7	DMA Request .....	483
27.3.8	Packet Error Check (PEC) .....	484
27.4	I <sup>2</sup> C Interrupt Request.....	485
27.5	I <sup>2</sup> C Debug Mode .....	485
27.6	I <sup>2</sup> C Register Description.....	485
27.6.1	Control Register 1 (I2C_CR1) .....	486
27.6.2	Control Register 2 (I2C_CR2) .....	487
27.6.3	Self Address Register 1 (I2C_OAR1).....	488
27.6.4	Own Address Register 2 (I2C_OAR2) .....	488
27.6.5	Data Register (I2C_DR) .....	488
27.6.6	Status Register 1 (I2C_SR1) .....	489
27.6.7	Status Register 2 (I2C_SR2) .....	491
27.6.8	Clock Control Register (I2C_CCR).....	491
27.6.9	TRISE Register (I2C_TRISE).....	492
27.6.10	I2C Register Address Map.....	493
<b>28</b>	<b>Universal Synchronous Asynchronous Transceiver (USART) .....</b>	<b>494</b>
28.1	Introduction to USART .....	494
28.2	USART Key Features.....	494
28.3	USART Function Overview.....	495
28.3.1	USART Characterization .....	496
28.3.2	Transmitters.....	497
28.3.3	Refraction .....	499
28.3.4	Fractional Baud Rate Generation .....	502
28.3.5	USART Receiver Tolerates Clock Changes .....	503
28.3.6	Multiprocessor Communication.....	503

---

28.3.7	Calibration Control .....	505
28.3.8	LIN (Local Area Network) Mode .....	505
28.3.9	USART Synchronization Mode .....	507
28.3.10	Single-Line Half-Duplex Communication .....	509
28.3.11	Smart Card .....	509
28.3.12	IrDA SIR ENDEC Function Module .....	511
28.3.13	Continuous Communication using DMA .....	512
28.3.14	Hardware Flow Control .....	514
28.4	USART Interrupt Request .....	515
28.5	USART Mode Configuration .....	516
28.6	USART Register Description .....	516
28.6.1	Status Register (USART_SR) .....	517
28.6.2	Data Register (USART_DR) .....	518
28.6.3	Baud Rate Register (USART_BRR) .....	518
28.6.4	Control Register 1 (USART_CR1) .....	519
28.6.5	Control Register 2 (USART_CR2) .....	520
28.6.6	Control Register 3 (USART_CR3) .....	521
28.6.7	Guard Time and Prescaler Register (USART_GTPR) .....	522
28.6.8	USART Register Address Map .....	523
<b>29</b>	<b>USB On-The-Go Full-Speed (OTG_FS) .....</b>	<b>524</b>
29.1	OTG Module Introduction .....	524
29.2	OTG_FS Main Functions .....	524
29.2.1	Common Functionality .....	524
29.2.2	Host Mode Functions .....	525
29.3	OTG_FS Functional Description .....	525
29.3.1	OTG Pin .....	526
29.3.2	OTG Full Speed Controller .....	526
29.3.3	Full-Speed OTGPHY (Physical Interface) .....	526
29.4	OTG Dual Role Device (DRD) .....	527
29.4.1	ID Signal Detection .....	527
29.4.2	HNP Dual Role Devices .....	527
29.4.3	SRP Dual Role Devices .....	527
29.5	USB Device Mode .....	528
29.5.1	SRP-Capable Devices .....	528
29.5.2	Device Status .....	528
29.5.3	Device Endpoint .....	529
29.6	USB Host .....	531
29.6.1	SRP-Capable Hosts .....	531
29.6.2	USB Host Status .....	532
29.6.3	Host Channel .....	533
29.6.4	Host Scheduler .....	534
29.7	SOF Trigger .....	535
29.7.1	Host SOFs .....	535
29.7.2	Device SOFs .....	535
29.8	Power Supply Options .....	536
29.9	Dynamic Update of OTG FS HFIR Registers .....	537
29.10	USB Data FIFOs .....	537
29.11	FIFO Structure in Device Mode .....	538

---

---

29.11.1	Receive FIFO in Device Mode .....	538
29.11.2	Transmit FIFO in Device Mode.....	538
29.12	FIFO Structure in Host Mode .....	539
29.12.1	Receive FIFO in Host Mode.....	539
29.12.2	Transmit FIFO in Host Mode .....	540
29.13	FIFO RAM Allocation.....	540
29.13.1	Device Mode.....	540
29.14	USB System Performance .....	540
29.15	OTG_FS Interrupt .....	541
29.16	OTG_FS Control and Status Registers.....	542
29.16.1	CSR Memory Image .....	543
29.16.2	OTG_FS Global Registers .....	545
29.16.3	Registers in Host Mode .....	559
29.16.4	Registers in Device Mode.....	566
29.16.5	OTG_FS Power and Clock Gating Register (OTG_FS_PCGCCTL) .....	581
29.16.6	OTG_FS Register Image.....	581
29.17	OTG_FS Programming Rules.....	589
29.17.1	Controller Initialization .....	589
29.17.2	Initialization in Host Mode .....	589
29.17.3	Initialization in Device Mode .....	589
29.17.4	Programming Rules in Host Mode .....	590
29.17.5	Programming Rules in Device Mode .....	603
29.17.6	Workflow.....	604
29.17.7	Worst Case Response Time.....	617
29.17.8	OTG Programming Rules.....	618
<b>30</b>	<b>Device Electronic Signature.....</b>	<b>624</b>
30.1	Memory Capacity Register.....	624
30.1.1	Flash Capacity Register.....	624
30.2	Product Unique Identification Register (96 bits) .....	624
<b>31</b>	<b>Debugging Support (DBG) .....</b>	<b>626</b>
31.1	General Situation .....	626
31.2	ARM References.....	627
31.3	SWJ Debug Port (Serial Wire and JTAG) .....	627
31.3.1	Mechanism for JTAG-DP and SW-DP Switching.....	627
31.4	Pinouts and Debug Port Pins .....	628
31.4.1	SWJ Debug Port Pin .....	628
31.4.2	Flexible SWJ-DP Pin Assignment .....	628
31.4.3	Internal Pull-ups and Pull-downs on the JTAG Pin .....	629
31.4.4	Utilizes the Serial Interface and Releases the Unused Debug Pins as Normal I/O Ports.....	629
31.5	W55MH32JTAG TAP Connection .....	630
31.6	ID Codes and Locking Mechanisms .....	630
31.6.1	Microcontroller Device ID Code .....	630
31.6.2	Boundary Scan TAP.....	631
31.6.3	Cortex-M3 TAP .....	631
31.6.4	Cortex-M3 JEDEC-106 ID Code.....	631
31.7	JTAG Debug Port.....	631
31.8	SW Debug Port .....	632

---

---

31.8.1	Introduction to SW Protocol .....	632
31.8.2	SW Protocol Sequence .....	632
31.8.3	SW-DP State Machine (Reset, Idle States, IDcode) .....	633
31.8.4	DP and AP Read/Write Access .....	633
31.8.5	SW-DP Register .....	633
31.8.6	SW-AP Register.....	634
31.9	AHB-AP (AHB Access Port) Valid for Either JTAG-DP or SWDP .....	634
31.10	Kernel Debugging .....	634
31.11	Debugger Host Connectivity under System Reset.....	635
31.12	FPB (Flash Patch Breakpoint) .....	635
31.13	DWT (Data Watch Point ) .....	635
31.14	ITM (Instruction Trace Microcell Instrumentation Trace Macrocell).....	636
31.14.1	Summarize.....	636
31.14.2	Timestamp Packets, Synchronization and Overflow Packets .....	636
31.15	ETM Module (Embedded Trace Macrocell).....	637
31.15.1	Summarize.....	637
31.15.2	Signaling Protocols and Packet Types .....	637
31.15.3	Main ETM Registers.....	637
31.15.4	Configuration Example .....	637
31.16	MCU Debug Module (MCUDBG).....	638
31.16.1	Debugging Support for Low-Power Modes .....	638
31.16.2	Supports Timer, Watchdog, bxCAN and I2C Debugging.....	638
31.16.3	Debugging MCU Configuration Registers .....	638
31.17	TPIU (Trace Port Interface ) .....	640
31.17.1	Introductory .....	640
31.17.2	Tracking Pin Assignment .....	640
31.17.3	TPUI Formatter .....	641
31.17.4	TPUI Frame Asynchronous Package .....	642
31.17.5	Synchronized Frame Packet Delivery.....	642
31.17.6	Synchronous Mode .....	642
31.17.7	Asynchronous Mode .....	642
31.17.8	TRACECLKIN Connections inside the W55MH32 .....	642
31.17.9	TPIU Register.....	643
31.17.10	Examples of Configurations .....	643
31.18	DBG Register Address Map.....	644
32	Document History .....	645



## List of illustrations

Figure1 System Architecture.....	31
Figure2 Block diagram of the CRC calculation unit .....	38
Figure3 Power Supply Block Diagram.....	42
Figure4 Power-on reset and power-down reset waveforms.....	44
Figure5 Threshold for PVD .....	44
Figure6 Reset Circuit.....	57
Figure7 Clock tree .....	58
Figure8 HSE/LSE Clock Source .....	59
Figure9 Basic Structure of I/O Port Bits .....	77
Figure10 Basic Structure of 5-Volt Compatible I/O Port Bits .....	78
Figure11 Input Float/Pull-Up/Pull-Down Configuration .....	80
Figure12 Output Configuration .....	80
Figure13 Multiplexing Function Configuration .....	81
Figure14 High Impedance Analog Input Configuration.....	82
Figure15 External Interrupt/Event Controller Block Diagram .....	101
Figure16 External Interrupt General Purpose I/O Image .....	103
Figure17 Register and Memory Composition .....	108
Figure18 INTLEVEL Timing Sequence .....	112
Figure19 DMA Block Diagram.....	126
Figure20 DMA1 request mapping .....	130
Figure21 DMA2 request mapping .....	131
Figure22 Single ADC Block Diagram.....	138
Figure23 Timing diagram .....	140
Figure24 Analog Watchdog Alert Area.....	141
Figure25 Injection Conversion Delay .....	142
Figure26 Calibration Timing Diagram.....	143
Figure27 Data right-aligned.....	144
Figure28 Data Left Alignment .....	144
Figure29 Dual ADC Block Diagram <sup>(1)</sup> .....	147
Figure30 Synchronized injection pattern on 4 channels.....	148
Figure31 Synchronization rule pattern on 16 channels.....	148
Figure.32 Fast crossover mode in continuous conversion mode on 1 channel .....	149
Figure33 Slow crossover mode on 1 channel .....	149
Figure34 Alternate triggering: injected channel groups per ADC1 .....	150
Figure35 Alternate triggering: 4 injection channels on each ADC in intermittent mode .....	150
Figure36 Alternate+ Rule Synchronization.....	151
Figure37 Trigger event occurs during injection transition .....	151
Figure38 Crossed single-channel conversions are injected into the sequence CH11 and CH12 interrupts.....	151
Figure39 Temperature Sensor and VREFINT Channel Block Diagram .....	152
Figure40 DAC Channel Module Block Diagram.....	163
Figure41 Data Registers for Single DAC Channel Mode .....	164
Figure42 Data Registers for Dual DAC Channel Mode .....	165
Figure43 Time block diagram of conversion when TEN=0 triggers deactivation.....	165
Figure44 DACLFSR Register Algorithm.....	166
Figure45 DAC Conversion with LFSR Waveform Generation (Enable Software Trigger) .....	167
Figure46 DAC Triangle Wave Generation .....	167

---

Figure47 DAC conversion with delta generation (enable software trigger) .....	167
Figure48 Level Control Timer Block Diagram .....	179
Figure.49 Timing diagram of the counter when the parameter of the prescaler is changed from 1 to 2 .....	180
Figure50 Timing diagram of the counter when the parameter of the prescaler is changed from 1 to 4 .....	181
Figure51 Timing diagram of the counter with internal clock division factor of 1 .....	182
Figure52 Timing diagram of the counter with internal clock division factor of 2 .....	182
Figure53 Timing diagram of the counter with internal clock division factor of 4 .....	182
Figure54 Timing diagram of the counter with internal clock division factor N .....	183
Figure55 Counter timing diagram, update event when ARPE=0 (TIMx_ARR is not preloaded) .....	183
Figure56 Counter timing diagram, update event when ARPE=1 (preloaded with TIMx_ARR) .....	184
Figure57 Timing diagram of the counter with internal clock division factor of 1 .....	185
Figure58 Timing diagram of the counter with internal clock division factor of 2 .....	185
Figure59 Timing diagram of the counter with internal clock division factor of 4 .....	186
Figure60 Timing diagram of the counter with internal clock division factor N .....	186
Figure61 Counter Timing Diagram, Update Events When Repeat Counter is Not Used .....	187
Figure62 Counter Timing Diagram with Internal Clock Division Factor of 1 and TIMx_ARR=0x6 .....	188
Figure63 Timing diagram of the counter with internal clock division factor of 2 .....	188
Figure64 Counter Timing Diagram with Internal Clock Division Factor of 4, TIMx_ARR=0x36 .....	188
Figure65 Timing diagram of the counter with internal clock division factor N .....	189
Figure66 Counter Timing Diagram, Update Event at ARPE=1 (Counter Underflow) .....	189
Figure67 Counter Timing Diagram, Update Event at ARPE=1 (Counter Overflow) .....	189
Figure68 Examples of update rates in different modes, and register settings for TIMx_RCR .....	190
Figure69 Control circuit in general mode with internal clock division factor of 1 .....	191
Figure70 TI2 External Clock Connection Example .....	191
Figure71 Control circuit in external clock mode 1 .....	192
Figure72 External Trigger Input Block Diagram .....	192
Figure73 Control Circuit in External Clock Mode 2 .....	193
Figure74 Capture/compare channels (e.g. channel 1 input section) .....	193
Figure75 Main circuit for capture/compare channel 1 .....	194
Figure76 Output section of capture/compare channels (channels 1 to 3) .....	194
Figure77 Output section of the capture/compare channel (channel 4) .....	194
Figure78 PWM Input Mode Timing .....	196
Figure79 Output Compare Mode, Flip OC1 .....	197
Figure80 Edge-aligned PWM waveform (ARR=8) .....	198
Figure81 Centrally aligned PWM waveform (APR=8) .....	199
Figure82 Complementary output with deadband insertion .....	200
Figure83 Deadband waveform delay greater than negative pulse .....	200
Figure84 Deadband waveform delay greater than positive pulse .....	201
Figure85 Outputs in response to braking .....	203
Figure86 Clear OCxREF for TIMx .....	204
Figure87 Example of generating a six-step PWM, using COM (OSSR=1) .....	205
Figure88 Example of single pulse mode .....	206
Figure89 Example of counter operation in encoder mode .....	208
Figure90 Example of encoder interface mode for IC1FP1 inversion .....	208
Figure91 Example of Hall sensor interface .....	210
Figure92 Control circuit in reset mode .....	211

---

---

Figure93 Control circuit in gated mode.....	211
Figure94 Control Circuit in Trigger Mode .....	212
Figure95 Control Circuit in External Clock Mode 2 + Trigger Mode.....	213
Figure96 General Purpose Timer Block Diagram.....	233
Figure97 Timing diagram of the counter when the parameter of the prescaler is changed from 1 to 2 .....	234
Figure98 Timing diagram of the counter when the parameter of the prescaler is changed from 1 to 4.....	234
Figure99 Timing diagram of the counter with internal clock division factor of 1 .....	235
Figure100 Timing diagram of the counter with internal clock division factor of 2 .....	235
Figure101 Timing diagram of the counter with internal clock division factor of 4 .....	236
Figure102 Timing diagram of the counter with internal clock division factor N .....	236
Figure103 Counter timing diagram, update event when ARPE=0 (TIMx_ARR is not preloaded) .....	236
Figure104 Counter timing diagram, update event when ARPE=1 (preloaded with TIMx_ARR) .....	237
Figure105 Timing diagram of the counter with internal clock division factor of 1 .....	238
Figure106 Timing diagram of the counter with internal clock division factor of 2 .....	238
Figure107 Timing diagram of the counter with internal clock division factor of 4 .....	238
Figure108 Timing diagram of the counter with internal clock division factor N .....	239
Figure109 Counter Timing Diagram, Update Events When Repeat Counter is Not Used .....	239
Figure110 Counter Timing Diagram with Internal Clock Division Factor of 1 and TIMx_ARR=0x6 .....	240
Figure111 Timing diagram of the counter with internal clock division factor of 2 .....	240
Figure112 Counter Timing Diagram with Internal Clock Division Factor of 4, TIMx_ARR=0x36241 .....	241
Figure113 Timing diagram of the counter with internal clock division factor N .....	241
Figure114 Counter Timing Diagram, Update Event at ARPE=1 (Counter Underflow) .....	241
Figure115 Counter Timing Diagram, Update Event at ARPE=1 (Counter Overflow) .....	242
Figure116 Control circuit in general mode with internal clock division factor of 1 .....	242
Figure117 TI2 External Clock Connection Example.....	243
Figure118 Control circuit in external clock mode 1.....	243
Figure119 External Trigger Input Block Diagram .....	244
Figure120 Control Circuit in External Clock Mode 2 .....	244
Figure121 Capture/compare channels (e.g. channel 1 input section) .....	245
Figure122 Main circuit for capture/compare channel 1 .....	245
Figure123 Output section of the capture/compare channel (channel 1) .....	246
Figure124 PWM Input Mode Timing .....	247
Figure125 Output Compare Mode, Flip OC1 .....	248
Figure126 Edge-aligned PWM waveform (ARR=8).....	249
Figure127 Centrally aligned PWM waveform (APR=8).....	250
Figure128 Example of single pulse mode .....	251
Figure129 Clear OCxREF for TIMx .....	252
Figure130 Example of counter operation in encoder mode.....	254
Figure131 Example of encoder interface mode for IC1FP1 inversion .....	254
Figure132 Control circuit in reset mode .....	255
Figure133 Control circuit in gated mode .....	256
Figure134 Control Circuit in Trigger Mode.....	256
Figure135 Control Circuit in External Clock Mode 2 + Trigger Mode .....	257
Figure136 Example of a Master/Slave Timer .....	257
Figure137 OC1REF of Timer1 controls Timer2 .....	258

---

---

Figure138 Timer 2 can be controlled by enabling Timer 1.....	259
Figure139 Triggering Timer 2 using an update from Timer 1.....	259
Figure140 Triggering Timer 2 using Timer 1's Enable .....	260
Figure141 Triggering Timer 1 and Timer 2 using the TI1 input of Timer 1 .....	261
Figure142 General Purpose Timer Block Diagram (TIM9/TIM12) .....	277
Figure143 General purpose timer block diagram (TIM10/11/12/13/14).....	278
Figure144 Timing diagram of the counter when the parameter of the prescaler is changed from 1 to 2 .....	279
Figure145 Timing diagram of the counter when the parameter of the prescaler is changed from 1 to 4.....	279
Figure146 Timing diagram of the counter with internal clock division factor of 1 .....	280
Figure147 Timing diagram of the counter with internal clock division factor of 2 .....	281
Figure148 Timing diagram of the counter with internal clock division factor of 4 .....	281
Figure149 Timing diagram of the counter with internal clock division factor N .....	281
Figure150 Counter timing diagram, update event when ARPE=0 (TIMx_ARR is not preloaded) .....	282
Figure151 Counter timing diagram, update event when ARPE=1 (preloaded with TIMx_ARR) .....	282
Figure152 Control circuit in general mode with internal clock division factor of 1 .....	283
Figure153 TI2 External Clock Connection Example.....	283
Figure154 Control circuit in external clock mode 1.....	284
Figure155 Capture/compare channels (e.g. channel 1 input section) .....	284
Figure156 Main circuit for capture/compare channel 1 .....	285
Figure157 Output section of the capture/compare channel (channel 1) .....	285
Figure158 PWM Input Mode Timing .....	286
Figure159 Output Compare Mode, Flip OC1 .....	288
Figure160 Edge-aligned PWM waveform (ARR=8).....	289
Figure161 Example of single pulse mode .....	289
Figure162 Control circuit in reset mode .....	291
Figure163 Control Circuit in Gated Mode .....	291
Figure164 Control Circuit in Trigger Mode.....	292
Figure165 Basic Timer Block Diagram .....	308
Figure166 Timing diagram of the counter with prescaler factor changed from 1 to 2 .....	309
Figure167 Timing diagram of the counter with prescaler coefficient changed from 1 to 4 ...	309
Figure168 Timing diagram of the counter with internal clock division factor of 1 .....	310
Figure169 Timing diagram of the counter with internal clock division factor of 2 .....	310
Figure170 Timing diagram of the counter with internal clock division factor of 4 .....	311
Figure171 Timing diagram of the counter with internal clock division factor N .....	311
Figure172 Counter Timing Diagram, Update Event when ARPE=0 (TIMx_ARR not preloaded). .....	311
Figure173 Counter Timing Diagram, Update Event when ARPE=1 (Preloaded TIMx_ARR) .....	312
Figure174 Normal mode timing diagram with internal clock divider factor of 1 .....	312
Figure175 Simplified RTC Block Diagram .....	318
Figure176 Example RTC Seconds and Alarm Clock Waveform Graph, PR=0003, ALARM=00004 .....	319
Figure177 RTC Overflow Waveform Diagram Example, PR=0003 .....	319
Figure178 Standalone Watchdog Block Diagram.....	326
Figure179 Watchdog Block Diagram .....	329
Figure180 Window Watchdog Timing Chart .....	330
Figure181 SDIO "No Response" and "No Data" Operations .....	346
Figure182 SDIO (Multi) Data Block Read Operation .....	347

---

Figure183 SDIO (Multi) Data Block Write Operation.....	347
Figure184 SDIO Sequential Read Operation .....	347
Figure185 SDIO Sequential Write Operation.....	348
Figure186 SDIO Block Diagram.....	348
Figure187 SDIO Adapter .....	349
Figure188 Control Unit.....	350
Figure189 SDIO Adapter Command Channel .....	350
Figure190 Command Channel State Machine (CPSM) .....	351
Figure191 SDIO Command Transfer .....	351
Figure192 Data Channel .....	353
Figure193 Data Channel State Machine (DPSM) .....	354
Figure194 USB Device Block Diagram.....	386
Figure195 Locating the corresponding buffer description table entries for grouped buffers .....	390
Figure196 CAN network topology .....	409
Figure197 bxCAN operating mode.....	411
Figure198 bxCAN operating in silent mode .....	411
Figure199 bxCAN operating in loopback mode.....	412
Figure200 bxCAN operating in loopback silent mode .....	412
Figure201 Send Mailbox Status .....	414
Figure202 Receive FIFO Status .....	415
Figure203 Filter Group Bit Width Setting-Register Organization.....	417
Figure204 Example of filter numbering .....	418
Figure205 Examples of filter mechanisms .....	419
Figure206 CAN Error Status Diagram .....	420
Figure207 Bit Timing .....	421
Figure208 Various CAN frames.....	422
Figure209 Event Flags and Interrupt Generation .....	423
Figure210 SPI Block Diagram.....	441
Figure211 Single Master and Single Slave Applications .....	442
Figure212 Data Clock Timing Diagram .....	443
Figure213 Diagram of TXE/RXNE/BSY changes during continuous transmission in main mode, full duplex mode (BIDIMODE=0 and RXONLY=0) .....	447
Figure214 Schematic diagram of TXE/RXNE/BSY changes during continuous transmission in slave mode, full duplex mode (BIDIMODE=0 and RXONLY=0).....	448
Figure215 Intention of TXE/BSY change during continuous transmission in the master device transmit-only mode (BIDIMODE=0 and RXONLY=0) .....	448
Figure216 Diagram of TXE/BSY change during continuous transmission in transmit-only mode (BIDIMODE=0 and RXONLY=0) from slave device .....	449
Figure217 Schematic diagram of RXNE variation during continuous transmission in receive-only mode (BIDIMODE=0 and RXONLY=1) .....	449
Figure218 Schematic diagram of TXE/BSY variation when sending in discontinuous transmission (BIDIMODE=0 and RXONLY=0).....	450
Figure219 Send using DMA.....	453
Figure220 Receiving with DMA.....	454
Figure221 I2S Block Diagram.....	456
Figure222 I2S Philips protocol waveforms (16/32 bit full precision, CPOL=0) .....	457
Figure223 I2S Philips Protocol Standard Waveform (24-bit frame, CPOL=0) .....	458
Figure224 sends 0x8EAA33 .....	458
Figure225 Receive 0x8EAA33 .....	458

---

Figure226 I2S Philips Protocol Standard Waveforms (16-bit extended to 32-bit packet frame, CPOL=0).....	458
Figure227 Example .....	459
Figure228 MSB aligned 16-bit or 32-bit full-precision, CPOL=0 .....	459
Figure229 MSB aligned 24-bit data, CPOL=0.....	459
Figure230 MSB aligned 16-bit data extended to 32-bit packet frame with CPOL=0 .....	459
Figure231 LSB aligned 16-bit or 32-bit full precision, CPOL=0 .....	460
Figure232 LSB aligned 24-bit data, CPOL=0 .....	460
Figure233 Request to send 0x3478AE operation .....	460
Figure234 Request to receive 0x3478AE operation .....	460
Figure235 LSB aligned 16-bit data extended to 32-bit packet frame with CPOL=0 .....	461
Figure236 Example .....	461
Figure237 PCM standard waveform (16-bit) .....	461
Figure238 PCM standard waveform (16-bit extended to 32-bit packet frame) .....	462
Figure239 Audio Sampling Frequency Definition .....	462
Figure240 I2S Clock Generator Architecture .....	462
Figure241 I <sup>2</sup> C Bus Protocols .....	475
Figure242 Functional Block Diagram of I <sup>2</sup> C .....	476
Figure243 Transmission Sequence Diagram from Transmitter.....	477
Figure244 Transmission Sequence Diagram from Receiver .....	477
Figure245 Master Transmitter Transmission Sequence Diagram .....	479
Figure246 Master Receiver Transmission Sequence Diagram .....	480
Figure247 I <sup>2</sup> C Interrupt Mapping Map.....	485
Figure248 USART Block Diagram .....	496
Figure249 Word length setting .....	497
Figure250 Configuring the Stop Bit .....	498
Figure251 Changes in TC/TXE when transmitting .....	499
Figure252 Start Bit Detection.....	499
Figure253 Data sampling for noise detection .....	501
Figure254 Silent mode utilizing idle bus detection .....	504
Figure255 Silent mode using address tag detection.....	504
Figure256 Disconnect detection in LIN mode (11-bit disconnect length - LBDL bit set) .....	506
Figure257 Disconnect detection in LIN mode with frame error detection .....	507
Figure258 Example of a synchronized USART transmission .....	508
Figure259 USART Data Clock Timing Example (M=0) .....	508
Figure260 USART Data Clock Timing Example (M=1) .....	508
Figure261 RX Data Sample/Hold Time .....	509
Figure262 ISO7816-3 asynchronous protocols .....	510
Figure263 Detecting parity check errors using 1.5 stop bits.....	511
Figure264 IrDA SIR ENDEC Block Diagram .....	512
Figure265 IrDA Data Modulation (3/16) Normal Mode .....	512
Figure266 Send using DMA.....	513
Figure267 Receiving with DMA.....	514
Figure268 Hardware flow control between two USARTs .....	514
Figure269 RTS Flow Control.....	515
Figure270 CTS Flow Control.....	515
Figure271 USART interrupt image.....	516
Figure272 Block Diagram .....	525
Figure273 OTG A-B Device Connections.....	527

---



---

Figure274 Simple USB peripheral connection .....	528
Figure275 Simple USB Host Connection .....	531
Figure276 SOF Connections .....	535
Figure277 Dynamic Update OTG FS HFIR .....	537
Figure278 FIFO address image in device mode and FIFO operation image in AHB .....	538
Figure279 Host mode FIFO address image and FIFO operation image for AHBs .....	539
Figure280 Interrupt Architecture .....	542
Figure281 CSR memory image .....	543
Figure282 Send FIFO Write Task.....	591
Figure283 Receive FIFO readout task .....	592
Figure284 Normal block/control OUT/SETUP and block/control IN transfer process.....	593
Figure285 IN transfer process of Bulk/Control.....	595
Figure286 Normal Interrupt OUT/IN transmission process .....	597
Figure287 Normal Isochronous OUT/IN transmission process.....	600
Figure288 Reading out a data message from a receive FIFO in slave mode.....	605
Figure289 Processing a SETUP data message.....	607
Figure290 Block (Bulk) OUT transfer process in slave mode.....	611
Figure291 TRDT Maximum Timing of the Case .....	618
Figure292 SRP for the A device.....	619
Figure293 Class B Device SRP .....	620
Figure294 Class A equipment HNP .....	621
Figure295 Class B equipment HNP .....	622
Figure296 Debugging Block Diagram for W55MH32 level and Cortex™-M3 level .....	626
Figure297 SWJ Debug Port .....	627
Figure298 JTAG TAP connection .....	630
Figure299 TPIU Block Diagram.....	640

## List of Tables

Table1 Register Group Starting Address .....	32
Table2 Flash Module Organization .....	35
Table3 Startup Modes .....	36
Table4 CRC Calculation Unit Register Images and Reset values .....	41
Table5 Low Power Mode List .....	45
Table6 SLEEP-NOW Mode .....	46
Table7 SLEEP-ON-EXIT Modes .....	46
Table8 Stop Mode .....	47
Table9 Standby Modes .....	48
Table10 PWR Register Address Map and Reset values .....	50
Table11 BKP register image and Reset values .....	53
Table12 RCC Register Address Map and Reset values .....	75
Table13 Port Bit Configuration Table .....	78
Table14 Output Mode Bits .....	78
Table15 Advanced Timer TIM1/TIM8 .....	82
Table16 General Purpose Timer TIM2/3/4/5 .....	82
Table17 USART s .....	82
Table18 SPI .....	82
Table19 I2S .....	83
Table20 I2C Interface .....	83
Table21 BxCAN .....	83
Table22 USB .....	83
Table23 SDIO .....	83
Table24 ADC/DAC .....	83
Table25 Other I/O Functions .....	83
Table26 CAN1 Multiplexing Function Remapping .....	88
Table27 Debug Interface Signals .....	88
Table28 Debug Port Images .....	88
Table29 ADC1 External Trigger Injection Conversion Multiplexing Function Remapping .....	88
Table30 ADC1 External Trigger Rule Conversion Multiplexing Function Remapping .....	89
Table31 ADC2 External Trigger Injection Conversion Multiplexing Function Remapping .....	89
Table32 ADC2 External Trigger Rule Conversion Multiplexing Function Remapping .....	89
Table33 TIM5 Multiplexing Function Remap .....	89
Table34 TIM4 Multiplexing Function Reimage .....	89
Table35 TIM3 Multiplexing Functionality Remap .....	89
Table36 TIM2 Multiplexing Functional Remapping .....	89
Table37 TIM1 Multiplexing Function Reimage .....	89
Table38 TIM9 Remapping <sup>(1)</sup> .....	90
Table39 TIM10 Remapping <sup>(1)</sup> .....	90
Table40 TIM11 Remapping <sup>(1)</sup> .....	90
Table41 TIM13 Remapping <sup>(1)</sup> .....	90
Table42 TIM14 Remap <sup>(1)</sup> .....	90
Table43 USART3 Reimage .....	90
Table44 USART2 Reimage .....	90
Table45 USART1 Reimage .....	90
Table46 I2C1 Remap .....	91
Table47 SPI1 Reimage .....	91



Table48 SPI3 Remapping .....	91
Table49 GPIO Register Address Map and Reset values.....	96
Table50 AFIO Register Address Map and Reset values .....	97
Table51 W55MH32 Product Vector Table .....	98
Table52 External Interrupt/Event Controller Register Images and Reset values.....	106
Table53 Offset Addresses for General Purpose Registers .....	108
Table54 Socket n Offset addresses in registers (0 $n \leq 7$ ) .....	109
Table55 Programmable Data Transfer Width and Size End Operation (when PINC=MINC=1) ..	128
Table56 DMA Interrupt Requests .....	129
Table57 List of DMA1 requests for each channel .....	131
Table58 List of DMA2 requests for each channel .....	132
Table59 DMA Register Images and Resets.....	135
Table60 ADC Pins .....	139
Table61 Analog Watchdog Channel Selection .....	141
Table62 External triggering of ADC1 and ADC2 for rule channels .....	144
Table63 External triggering of ADC1 and ADC2 for injection channels .....	145
Table64 ADC3 External Trigger for Rule Channel.....	145
Table65 External triggering of ADC3 for injection channels.....	145
Table66 ADC Interrupts.....	153
Table67 ADC Register Image and Reset values.....	161
Table68 DAC Pins .....	164
Table69 External Trigger .....	166
Table70 DAC Register Image .....	177
Table71 Relationship between count direction and encoder signal .....	207
Table72 TIMx Internal Trigger Connections .....	216
Table73 Control bits for complementary output channels OCx and OCxN with brake function .....	224
Table74 TIM1 and TIM8 Register and Reset values .....	230
Table75 Relationship between count direction and encoder signal .....	253
Table76 TIMx Internal Trigger Connections .....	264
Table77 Output control bits for standard OCx channels .....	270
Table78 TIM2 to TIM5 Registers and Reset values.....	274
Table79 TIMx Internal Trigger Connections .....	294
Table80 Output control bits for standard OCx channels .....	298
Table81 TIM9/12 registers and Reset values .....	300
Table82 Output control bits for standard OCx channels .....	305
Table83 TIM10/11/13/14 Registers and Reset values .....	306
Table84 TIM6 and TIM7 Registers and Reset values.....	316
Table85 RTC-Register Image and Reset values .....	324
Table86 Watchdog Timeout (40kHz Input Clock (LSI)).....	326
Table87 IWDG Register Image and Reset values.....	328
Table88 Minimum and maximum timeout values @36 MHz ( $f_{PCLK1}$ ) .....	331
Table89 WWDG Register Image and Reset values .....	332
Table89 RNG Register Image and Reset values .....	335
Table89 SYSCFG Register Image and Reset values .....	340
Table89 OTP Register Image and Reset values.....	345
Table90 SDIO Pin Definitions.....	349
Table91 Command Format .....	352

---

Table92 Short Response Format.....	352
Table93 Long Response Format .....	352
Table94 Command Channel Status Flags.....	352
Table95 Data Token Format .....	355
Table96 Send FIFO Status Flags .....	355
Table97 Receive FIFO Status Flags.....	355
Table98 Card Status.....	363
Table99 SD Status .....	365
Table100 Speed Type Codes .....	366
Table101 Mobile Performance Codes.....	366
Table102 AU_SIZE codes.....	366
Table103 Maximum AU length .....	366
Table104 ERASE_SIZE codes.....	367
Table105 Erase Timeout Codes.....	367
Table106 Erase Offset Codes .....	367
Table107 Block Transfer Based Write Commands .....	369
Table108 Block Transfer Based Write Protect Commands .....	370
Table109 Erase Commands.....	370
Table110 I/O Mode Commands .....	370
Table111 Locking Commands .....	371
Table112 Application-related commands.....	371
Table113 R1 Response .....	371
Table114 R2 Response .....	372
Table115 R3 Response .....	372
Table116 R4 Response .....	372
Table117 R4b response .....	373
Table118 R5 Response .....	373
Table119 R6 Response .....	373
Table120 Response Type and SDIO_RESPx Registers .....	378
Table121 SDIO register images .....	385
Table122 Double Buffered Batch Endpoint Buffer Identification Definitions .....	393
Table123 Buffer usage identifiers for double-buffered batch endpoints .....	393
Table124 Buffer usage identifiers for synchronization endpoints .....	395
Table125 Wake-up event detection .....	396
Table126 Receive Status Codes .....	403
Table127 Endpoint Type Codes.....	403
Table128 Endpoint Special Type Definitions .....	403
Table129 Transmission Status Codes .....	403
Table130 Definition of Grouping Buffer Sizes .....	405
Table131 USB Register Image and Reset values .....	406
Table132 Send Mailbox Register List .....	419
Table133 Receive Mailbox Register List.....	420
Table134 List of bxCAN-registers and their Reset values .....	437
Table135 SPI Interrupt Requests .....	455
Table136 Getting Accurate Audio Frequencies with a Standard 8MHz HSE Clock .....	463
Table137 I <sup>2</sup> S Interrupt Requests.....	467
Table138 List of SPI registers and their Reset values.....	473
Table139 Comparison of SMBus and I <sup>2</sup> C .....	482
Table140 I <sup>2</sup> C Interrupt Request Table: .....	485

---

---

Table141 I2C Register Address Maps and Reset values .....	493
Table142 Data sampling for detection of noise.....	501
Table143 Error calculation when setting baud rate.....	503
Table144 USART receiver tolerance when DIV_Fraction=0 .....	503
Table145 USART receiver tolerance when DIV_Fraction!=0 .....	503
Table146 Frame format .....	505
Table147 USART Mode Settings <sup>(1)</sup> .....	516
Table148 List of USART registers and their Reset values.....	523
Table149 OTG Pins .....	526
Table150 W55MH32 Low Power and OTG Compatibility.....	536
Table151 Controller Global Control and Status Registers (CSRs) .....	543
Table152 Control and Status Registers (CSRs) in Host Mode.....	544
Table153 Device Mode Control and Status Registers.....	544
Table154 Data FIFO (DFIFO) Access Registers.....	545
Table155 Power Supply and Gating Control and Status Registers .....	545
Table156 TRDT values .....	549
Table157 Registers of the OTG_FS module and their Reset values.....	581
Table158 SWJ Debug Port Pins .....	628
Table159 Flexible SWJ_DP Pin Assignments.....	628
Table160 JTAG Debug Port Data Registers .....	631
Table161 32-bit Debug Interface Register Addresses Defined by A[3:2].....	632
Table162 Request Packet (8 bits).....	632
Table163 ACK Definitions (3 bits).....	632
Table164 Transmission data (33 bits) .....	633
Table165 SW-DP registers .....	633
Table166 Cortex-M3AHB-AP Registers.....	634
Table167 Kernel Debug Registers .....	635
Table168 Primary ITM Registers.....	636
Table169 Primary ETM registers.....	637
Table170 Asynchronous Tracking Pin Assignments .....	640
Table171 Synchronous Tracking Pin Assignments.....	640
Table172 Flexible Tracking Pin Assignments .....	641
Table173 Important TPIU Registers.....	643
Table174 DBG-Register and Reset values .....	644

## Abbreviations in The Text

The following abbreviations are used in the description of the registers:

nomenclature	descriptive
read/write(rw)	The software can read and write this bit.
read-only(r)	The software can only read this bit.
write-only(w)	Software can only write this bit; reading this bit will return the Reset value.
read/clear(rc_w1)	Software can read this bit or clear it by writing a '1'; writing a '0' has no effect on this bit.
read/clear(rc_w0)	Software can read this bit or clear it by writing a '0'; writing a '1' has no effect on this bit.
read/clear by read(rc_r)	Software can read this bit, reading this bit will automatically clear it to '0', writing '0' has no effect on this bit.
read/set(rs)	Software can read or set this bit, writing '0' has no effect on this bit.
read-only write trigger(rt_w)	The software can read or set this bit: writing '0' or '1' triggers an event but has no effect on the value of this bit.
toggle(t)	The software can only flip this bit by writing a '1', writing a '0' has no effect on this bit.
reserved(Res.)	reserved bit, the default value must be left unchanged.

## 2 Memory and Bus Architecture

### 2.1 System Framework

The main system consists of the following components:

- Four drive units:
  - Cortex™ -M3 core DCode bus (D-bus), and system bus (S-bus)
  - Universal DMA1 and Universal DMA2
- Three passive units
  - internal SRAM
  - internal flash memory
  - AHB to APB bridge (AHB2APBx), which connects all APB devices

These are interconnected through a multi-level AHB bus architecture as below in Figure shown Figure1 :

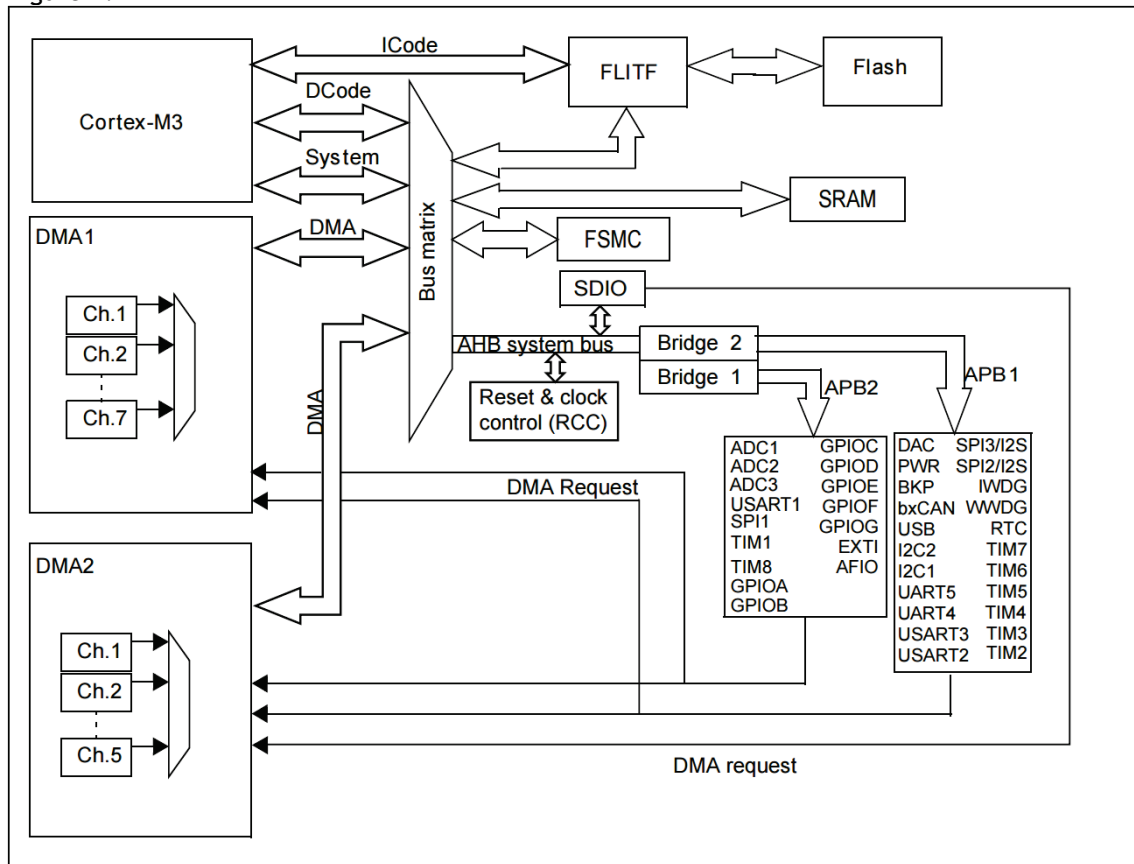


Figure1 System Architecture

#### ICode Bus

This bus connects the instruction bus of the Cortex™ -M3 core to the flash instruction interface. Instruction prefetching is done on this bus.

#### DCode Bus

This bus connects the DCode bus of the Cortex™ -M3 core to the data interface of the flash memory (constant load and debug access).

#### System Bus

This bus connects the system bus (peripheral bus) of the Cortex™ -M3 core to the bus matrix, which coordinates accesses between the core and the DMA.

#### DMA Bus

This bus connects the DMA's AHB master interface to the bus matrix, which coordinates the CPU's DCode and DMA accesses to SRAM, flash memory, and peripherals.

### Bus Matrix

The bus matrix coordinates access arbitration between the kernel system bus and the DMA master bus, which utilizes a rotation algorithm. The bus matrix contains four driving components (the CPU's DCode, system bus, DMA1 bus, and DMA2 bus) and three passive components (flash memory interface, SRAM, and AHB2APB bridge).

The AHB peripherals are connected to the system bus via a bus matrix that allows DMA access.

### AHB/APB Bridge (APB)

Two AHB/APB bridges provide a synchronous connection between the AHB and the 2 APB buses. APB1 operates at speed limited to 108MHz and APB2 operates at full speed (up to 216MHz). Refer to Table1 for address mapping of the different peripherals connected to each bridge. After each reset, all peripherals except SRAM and FLITF are turned off, and before using a peripheral, register RCC\_AHBENR must be set to turn on the clock for that peripheral.

*Notes: When an 8-bit or 16-bit access is made to an APB register, the access is automatically converted to a 32-bit access: the bridge automatically extends the 8-bit or 32-bit data to match the 32-bit vector.*

## 2.2 Memory Organization

Program memory, data memory, registers, and input/output ports are organized in the same 4GB linear address space.

Data bytes are stored in memory in little end format. The lowest address byte in a word is considered to be the least significant byte of the word, while the highest address byte is the most significant byte.

Refer to the relevant sections for the peripheral register images.

The accessible memory space is divided into 8 major blocks of 512MB each.

All other memory space not allocated to on-chip memory and peripherals is reserved address space.

## 2.3 Memory Image

Table1 lists the starting addresses of the built-in peripherals in the W55MH32 used.

Table1 Register Group Starting Address

starting address	peripherals	computer bus	register image
0x5006 0800-0x5006 0BFF	TRNG	AHB	See 20.2.6
0x5004 0000-0x5006 07FF	Reserved		
0x5000 0000-0x5003 ffff	USB OTG Full Speed		See 29.16.6
0x4003 0000-0x4FFF FFFF	Reserved		
0x4002 8000-0x4002 9FFF	Reserved		
0x4002 3400-0x4002 7FFF	Reserved		
0x4002 3000-0x4002 33FF	CRC		See 3.4.7
0x4002 2400-0x4002 2FFF	Reserved		
0x4002 2000-0x4002 23FF	flash memory interface		
0x4002 1400-0x4002 1FFF	Reserved		
0x4002 1000-0x4002 13FF	Reset and Clock Control (RCC)		See 6.3.13
0x4002 0800-0x4002 0FFF	Reserved		
0x4002 0400-0x4002 07FF	DMA2		See 10.4.7
0x4002 0000-0x4002 03FF	DMA1		See 10.4.7
0x4001 8400-0x4001 ffff	Reserved		
0x4001 8000-0x4001 83FF	SDIO		See 23.9.16
0x4001 7000-0x4001 7FFF	Reserved	APB2	
0x4001 6C50-0x4001 6CBF	OTP		See 22.2.13
0x4001 6C00-0x4001 6FFF	SYSCFG		See 21.1.9

starting address	peripherals	computer bus	register image
0x4001 5800-0x4001 7FFF	Reserved		
0x4001 5400-0x4001 57FF	TIM11 Timer		See 15.5.11
0x4001 5000-0x4001 53FF	TIM10 Timer		See 15.5.11
0x4001 4C00-0x4001 4FFF	TIM9 Timer		See 15.4.13
0x4001 4000-0x4001 4BFF	Reserved		
0x4001 3C00-0x4001 3FFF	ADC3		See 11.12.15
0x4001 3800-0x4001 3BFF	USART1		See 0
0x4001 3400-0x4001 37FF	TIM8 Timer		See 13.4.21
0x4001 3000-0x4001 33FF	SPI1		See 26.5.10
0x4001 2c00-0x4001 2fff	TIM1 Timer		See 13.4.21
0x4001 2800-0x4001 2BFF	ADC2		See 11.12.15
0x4001 2400-0x4001 27FF	ADC1		See 11.12.15
0x4001 2000-0x4001 23FF	GPIO Port G		See 0
0x4001 1c00-0x4001 1fff	GPIO Port F		See 0
0x4001 1800-0x4001 1BFF	GPIO Port E		See 0
0x4001 1400-0x4001 17FF	GPIO Port D		See 0
0x4001 1000-0x4001 13FF	GPIO Port C		See 0
0x4001 0c00-0x4001 0fff	GPIO Port B		See 0
0x4001 0800-0x4001 0BFF	GPIO Port A		See 0
0x4001 0400-0x4001 07FF	EXTI		See 8.3.7
0x4001 0000-0x4001 03FF	AFIO		See 0
0x4000 7800-0x4000 FFFF	Reserved	APB1	
0x4000 7400-0x4000 77FF	DAC		See 12.5.14
0x4000 7000-0x4000 73FF	Power Control (PWR)		See 4.4.3
0x4000 6C00-0x4000 6FFF	Backup Register (BKP)		See 5.4.5
0x4000 6800-0x4000 6BFF	Reserved		
0x4000 6400-0x4000 67FF	bxCAN		See 25.9.5
0x4000 6000-0x4000 63FF	USB/CAN shared 512 bytes SRAM		
0x4000 5C00-0x4000 5FFF	USB Full Speed Device Register		See section 21.5.4
0x4000 5800-0x4000 5BFF	I2C2		See 27.6.10
0x4000 5400-0x4000 57FF	I2C1		See 27.6.10
0x4000 5000-0x4000 53FF	UART5		See 0
0x4000 4C00-0x4000 4FFF	UART4		See 0
0x4000 4800-0x4000 4BFF	USART3		See 0
0x4000 4400-0x4000 47FF	USART2		See 0
0x4000 4000-0x4000 43FF	Reserved		
0x4000 3C00-0x4000 3FFF	SPI3/I2S3		See 26.5.10
0x4000 3800-0x4000 3BFF	Reserved		
0x4000 3400-0x4000 37FF	Reserved		
0x4000 3000-0x4000 33FF	Independent Watchdog Dog (IWDG)		See 18.4.5
0x4000 2C00-0x4000 2FFF	Window Watchdog (WWDG)		See 19.6.4
0x4000 2800-0x4000 2BFF	RTC		See 17.4.7
0x4000 2400-0x4000 27FF	Reserved		
0x4000 2000-0x4000 23FF	TIM14 Timer		See 15.5.11
0x4000 1C00-0x4000 1FFF	TIM13 Timer		See 15.5.11
0x4000 1800-0x4000 1BFF	TIM12 Timer		See 15.4.13
0x4000 1400-0x4000 17FF	TIM7 Timer		See 16.4.9
0x4000 1000-0x4000 13FF	TIM6 Timer		See 16.4.9

starting address	peripherals	computer bus	register image
0x4000 0C00-0x4000 0FFF	TIM5 Timer		See 14.4.19
0x4000 0800-0x4000 0BFF	TIM4 Timer		See 14.4.19
0x4000 0400-0x4000 07FF	TIM3 Timer		See 14.4.19
0x4000 0000-0x4000 03FF	TIM2 Timer		See 14.4.19

### 2.3.1 Embedded SRAM

The W55MH32 has a built-in 96K bytes of static SRAM. it can be accessed as a byte, half word (16-bit), or full word (32-bit). the starting address of the SRAM is 0x2000 0000.

### 2.3.2 Bandwidth

The Cortex™ -M3 memory image consists of two bit-band areas. These two bit-band areas map each word in the alias memory area to a bit in the bit-band memory area, and writing a word in the alias memory area has the same effect as performing a read-modify-write operation on the target bit in the bit-band area.

In the W55MH32, the peripheral registers and SRAM are mapped into a single bit-segment area, which allows single bit-segment write and read operations to be performed.

The following mapping equation gives how each word in the alias region corresponds to the corresponding bit in the bit-band region:

$$\text{bit\_word\_addr} = \text{bit\_band\_base} + (\text{byte\_offset} \times 32) + (\text{bit\_number} \times 4)$$

Among them:

bit\_word\_addr is the address of the word in the alias memory area that maps to some target bit.

bit\_band\_base is the starting address of the alias zone.

byte\_offset is the serial number of the byte containing the target bit in the bit field

bit\_number is the target bit location (0-31)

Example:

The following example shows how to map bit 2 in the byte with SRAM address 0x2000 0300 in the alias area:

$$0x2200\ 6008 = 0x2200\ 0000 + (0x300 \times 32) + (2 \times 4).$$

A write operation to address 0x2200 6008 has the same effect as performing a read-modify-write operation to bit 2 at address 0x2000 0300 bytes in SRAM.

Reading the 0x2200 6008 address returns the value (0x01 or 0x00) of bit 2 of address 0x2000 0300 bytes in SRAM.

Refer to the Cortex™ -M3 Technical Reference Manual for more information on bit segments.

### 2.3.3 Embedded Flash

The high-performance flash memory module has the following key features:

- Up to 1M bytes flash memory structure: flash memory has a main storage block and information block composition:

- Primary storage block capacity:

Main storage blocks up to 128K× 64-bit, each divided into 256 4K-byte pages (seeTable2

- Information block capacity

258× 64-bit (seeTable2

The flash memory interface is characterized by:

- Read interface with prefetch buffer (2× 64 bits per word)
- Select Byte Loader
- Flash Program/Erase Operations
- Access/write protection



Table2 Flash Module Organization

module (in software)	name (of a thing)	address	magnitude
primary storage block	Page 0	0x0800 0000 - 0x0800 0FFF	4K Byte
	Page 1	0x0800 1000 - 0x0800 1FFF	4K Byte
	Page 2	0x0800 2000 - 0x0800 2FFF	4K Byte
	Page 3	0x0800 3000 - 0x0800 3FFF	4K Byte
	...	...	...
	Page 255	0x080f f000 - 0x080f ffff	4K Byte
information block	system memory	0x1FFF F000 - 0x1FFF F7FF	2K Byte
	Select Byte	0x1FFF F000 - 0x1FFF F7FF	16Byte
Flash Memory Interface Register	FLASH_ACR	0x4002 2000 - 0x4002 2003	4 Byte
	FLASH_KEYR	0x4002 2004 - 0x4002 2007	4 Byte
	FLASH_OPTKEYR	0x4002 2008 - 0x4002 200B	4 Byte
	FLASH_SR	0x4002 200c - 0x4002 200f	4 Byte
	FLASH_CR	0x4002 2010 - 0x4002 2013	4 Byte
	FLASH_AR	0x4002 2014 - 0x4002 2017	4
	Reserved	0x4002 2018 - 0x4002 201B	4
	FLASH_OBR	0x4002 201c - 0x4002 201f	4
	FLASH_WRP	0x4002 2020 - 0x4002 2023	4
	FLASH_WRP	0x4002 2020 - 0x4002 2023	4

### Flash Memory Readout

Instruction and data access to the flash memory is done through the AHB bus. The prefetch module is used to read instructions over the ICode bus. Arbitration is acting on the flash interface and data access on the DCode bus takes precedence.

Read access can have the following configuration options:

- Wait Time: The number of wait states for read operations that can be changed at any time.
- Prefetch buffers (2 64-bit): are opened automatically after each reset. Since the size of each buffer (64-bit) is the same as the bandwidth of the flash memory, the contents of the entire buffer can be updated by a single read of the flash memory. Because of the prefetch buffer, the CPU can operate at higher frequencies. the CPU fetches up to 32-bit words at a time, and fetches one instruction while the next instruction is already waiting in the buffer.
- Half-cycle: for power consumption optimization.

Notes: 1. These options should be used in conjunction with the flash memory access time. The wait period reflects the relationship between the system clock (SYSCLK) frequency and the flash memory access time:

0 wait cycle when  $0 < \text{SYSCLK} < 24\text{MHz}$

1 wait cycle when  $24\text{MHz} < \text{SYSCLK} \leq 48\text{MHz}$

2 wait cycle when  $48\text{MHz} < \text{SYSCLK} \leq 72\text{MHz}$

2. The half-cycle configuration cannot be used with an AHB that uses a prescaler, and the clock system should be equal to the HCLK clock. This feature can only be used with an internal RC oscillator (HSI) that can be used directly when the clock frequency is 8MHz or less, or a master oscillator (HSE), but not a PLL.

3. When the AHB prescaler factor is not 1, the prefetch buffer must be turned on.

4. The prefetch buffer open and close operations can only be performed if the system clock (SYSCLK) is less than 24MHz and the AHB prescaler is not turned on (i.e., HCLK must be equal to SYSCLK). Generally, the prefetch buffer open and close operations are performed during initialization, when the microcontroller clock is provided by an 8MHz internal RC oscillator (HSI).

5. Using DMA: DMA accesses flash memory on the DCode bus, which has a higher priority than fetch fingers on the ICode. DMA has a spare cycle after each transfer is completed. Some instructions can be executed in conjunction with a DMA transfer.

### Programming and Erasing Flash Memory

Flash memory can be programmed to write 16 bits (half words) at a time.

Flash erase operations can be performed on a page-by-page basis or a complete erase (full erase). A full erase does not affect blocks of information.

To ensure that over-programming does not occur, the flash programming and erasure controller blocks are controlled by a fixed clock.

An interrupt can be triggered at the end of a write operation (program or erase). This interrupt can be used to exit from WFI mode only when the flash controller interface clock is on.

## Source Control/Status Register (PWR\_CSR)

Address offset: 0x00

Reset value: 0x0000 0030

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

Bit	notation	clarification
31:6	Reserved	Reserved
5	PRFTBS	PRFTBS: prefetch buffer status 0: prefetch buffer disabled 1: Prefetch buffer enabled
4	PRFTBE	PRFTBE: Enable prefetch buffer 0: prefetch buffer disabled 1: Prefetch buffer enabled
3	HLFCYA	HLFCYA: Enabling Half-Cycle Access to Flash Memory 0: Half-cycle banned 1: Half-cycle enabled
2:0	LATENCY	LATENCY: Delay These bits indicate the ratio of system clock cycles to Flash access time 000 Zero wait state 0 < SYSCLK <= 24 MHz 001 A wait state 24 MHz < SYSCLK <= 48 MHz 010 Two wait states 48 MHz < SYSCLK <= 72 MHz

## 2.4 Startup Configuration

In the W55MH32, three different boot modes can be selected via the BOOT[1:0] pins.

Table3 Startup Modes

Startup Mode Selection Pin		activation mode	clarification
BOOT1	BOOT0		
X	0	main flash memory	Main flash memory selected as boot area
0	1	system memory	System memory selected as boot area
1	1	internal SRAM	Internal SRAM selected as boot area

After system reset, the value of BOOT pin will be latched on the 4th rising edge of SYSCLK. The user can select the startup mode after a reset by setting the state of the BOOT1 and BOOT0 pins.

On exit from standby mode, the value of the BOOT pin is relocked; therefore, the BOOT pin should remain in the desired startup configuration in standby mode. After the boot delay, the CPU fetches the address of the top of the stack from address 0x0000 0000 and begins executing code from the address indicated by 0x0000 0004 in boot memory.

Because of the fixed memory image, the code area always starts at address 0x0000 0000 (accessed via the ICode and DCode buses), while the data area (SRAM) always starts at address 0x2000 0000 (accessed via the system bus). The Cortex-M3 CPU always gets the reset vector from the ICode bus, i.e., booting is only appropriate to start from the code area (typically from Flash). Starting (typically from Flash).The W55MH32 microcontroller implements a special mechanism whereby the system can boot not only from Flash memory or system memory, but also from the built-in SRAM.

Depending on the selected boot mode, the main flash memory, system memory, or SRAM can be accessed as follows:

- Booting from the main flash memory: the main flash memory is mapped to boot space (0x0000 0000), but it can still be accessed at its original address (0x0800 0000), i.e., the contents of the flash memory can be accessed in two address areas, 0x0000 0000 or 0x0800 0000.
- Booting from system memory: System memory is mapped to boot space (0x0000 0000), but it can still be accessed at its original address (original address 0x1FFF B000 for interconnect products, 0x1FFF F000 for other products).
- Booting from the built-in SRAM: The SRAM can only be accessed in the address area starting at 0x2000 0000.

---

*Notes: When booting from the built-in SRAM, the vector table must be remapped into the SRAM using the NVIC's exception table and offset registers in the application's initialization code.*

#### **Embedded Bootstrap Program**

The embedded bootstrap program is stored in the system memory and is written on the production line for reprogramming the flash memory via the available serial interface.

## 3 CRC Calculation Unit

### 3.1 Introduction to CRC

The cyclic redundancy check (CRC) computation unit is to get the CRC computation result of any 32-bit full word based on a fixed generating polynomial.

In other applications, the CRC technique is mainly used to verify the correctness and integrity of data transfer or data storage. The standard EN/IEC 60335-1 provides a method for verifying the integrity of flash memory, whereby a CRC calculation unit calculates a software identifier at program runtime, which is then compared with a reference identifier generated at connection time and stored in a designated memory space.

### 3.2 CRC Key Features

- Using the CRC-32 (Ethernet) polynomial:  $0x4C11DB7$ 
  - $x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^4+x^2+x+1$
- One 32-bit data register for input/output
- CRC calculation time: 4 AHB clock cycles (HCLK)
- General purpose 8-bit register (can be used to store temporary data)

The following figure shows the block diagram of the CRC calculation unit

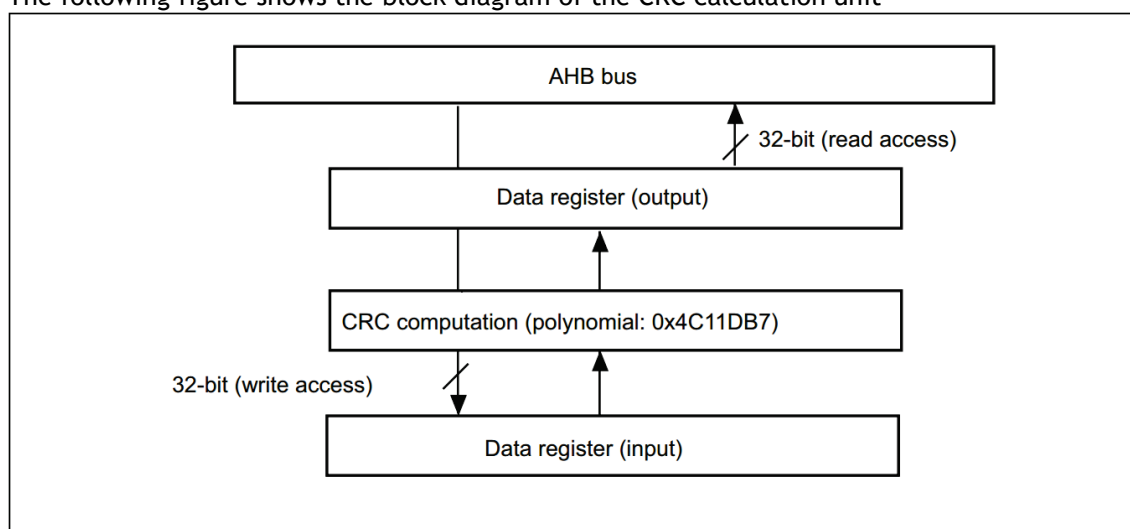


Figure2 Block diagram of the CRC calculation unit

### 3.3 CRC Functional Description

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

- When a write operation is performed on 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 (CRC calculations are performed on the entire 32-bit word, not byte by byte).

CPU write operations are suspended during CRC calculations, so back-to-back writes or sequential write-read operations can be performed to register CRC\_DR.

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 within register CRC\_IDR.

## 3.4 CRC Register

The CRC calculation unit consists of two data registers and one control register.

### 3.4.1 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	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
DR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	notation	clarification
31:0	CRC_DR	data register bits When new data is written to the CRC calculator, the result of the CRC calculation is returned when it is read as an input register.

### 3.4.2 Independent Data Register (CRC\_IDR)

Address offset: 0x04

Reset value: 0x0000 0000

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

Bit	notation	clarification
31:8	Reserved	Reserved
7:0	CRC_IDR	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

### 3.4.3 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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															RESET
															w

Bit	notation	clarification
31:1	Reserved	Reserved
0	RESET	RESET bit Resume the CRC calculation unit and set the data register to 0xFFFF FFFF. Only '1' can be written to this bit, it is cleared '0' automatically by hardware.

### 3.4.4 Control Status Extension Register (CRC\_CSR)

Address offset: 0x0C

Reset value: 0x0000 00CA

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

Bit	notation	clarification
31:8	Reserved	Reserved
7:6	BYTE_NUM	<b>BYTE_NUM:</b> Select the number of bytes for CRC calculation 0: Perform CRC operation only on byte0 1: byte0, byte1 2: byte0, byte1, byte2 3: byte0, byte1, byte2, byte3 Note: When configured as CRC16, only 0 and 1 can be configured When configured as CRC32, all can be configured
5	XOR_OUT	<b>XOR_OUT:</b> XOR the CRC calculation result with 0xffffffff 0: Output the calculation result directly 1: XOR CRC32 with 0xffffffff, and XOR CRC16 with 0x0000ffff
4	REV_OUT	<b>REV_OUT:</b> Reverse the high and low bits of the CRC calculation result 0: Do not reverse 1: Reverse
3	BYTE_REV	<b>BYTE_REV:</b> CRC input byte endian reversal For CRC32: 0: Calculation order byte0->byte1->byte2->byte3 1: Calculation order byte3->byte2->byte1->byte0 For CRC16: 0: Calculation order byte0->byte1 1: Calculation order byte1->byte0
2	REV_IN	<b>REV_IN:</b> Calculate by reversing the endianness of the CRC 8-bit input size. For example, bit7 participates in the operation as bit0, bit6 as bit1, and so on. 0: Do not reverse 1: Reverse
1	TYPE	<b>TYPE:</b> CRC Type Selection 0: CRC16 1: CRC32
0	POLY	<b>POLY:</b> Multiple options for CRC16. This bit is invalid when CRC32 is selected. 0: 0x8005 1: 0x1021

### 3.4.5 Initial Value Expansion Register (CRC\_INI)

Address offset: 0x10

Reset value: 0xFFFF FFFF

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

Bit	notation	clarification
31:0	CRC_INI	<b>CRC_INI:</b> Initial value for CRC calculation After writing to this register, it is necessary to configure the RESET register to load this value. Only the lower 16 bits are valid in CRC16 mode.

### 3.4.6 Result XOR Register (CRC\_XOR)

Address offset: 0x14

Reset value: 0xFFFF FFFF

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

Bit	notation	clarification
31:0	CRC_XOR	CRC_XOR: XOR value of the CRC result After writing to this register, it is necessary to configure the RESET register to load this value. Only the lower 16 bits are valid in CRC16 mode.

### 3.4.7 CRC Register Image

The following table lists the register images and Reset values for CRCs

Table4 CRC Calculation Unit Register Images and Reset values

Offset	Register	31	31	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h	CRC_DR	Data Register0xFFFF FFFF																															
	Reset value																																
004h	CRC_IDR	Reserved																								Independent data register0x00							
	Reset value																																
008h	CRC_CR	Reserved																															RESET
	Reset value																																
00Ch	CRC_CSR	Reserved																								BYTE_NUM		XOR_OUT	REV_OUT	BYTE_REV	REV_IN	TYPE	POLY
	Reset value																																
010h	CRC_INI	CRC_INI[31:0]																															
	Reset value																																
014h	CRC_XOR	CRC_XOR[31:0]																															
	Reset value																																

For the starting addresses of the registers, see Table1



## 4 Power Control (PWR)

### 4.1 Power Supply

The W55MH32 operates from a voltage ( $V_{DD}$ ) of 2.0 ~ 3.6 V. The required 1.8 V supply is provided through the built-in voltage regulator. When the main power supply  $V_{DD}$  is powered down, power is provided to the real-time clock (RTC) and backup registers through the  $V_{BAT}$  pin.

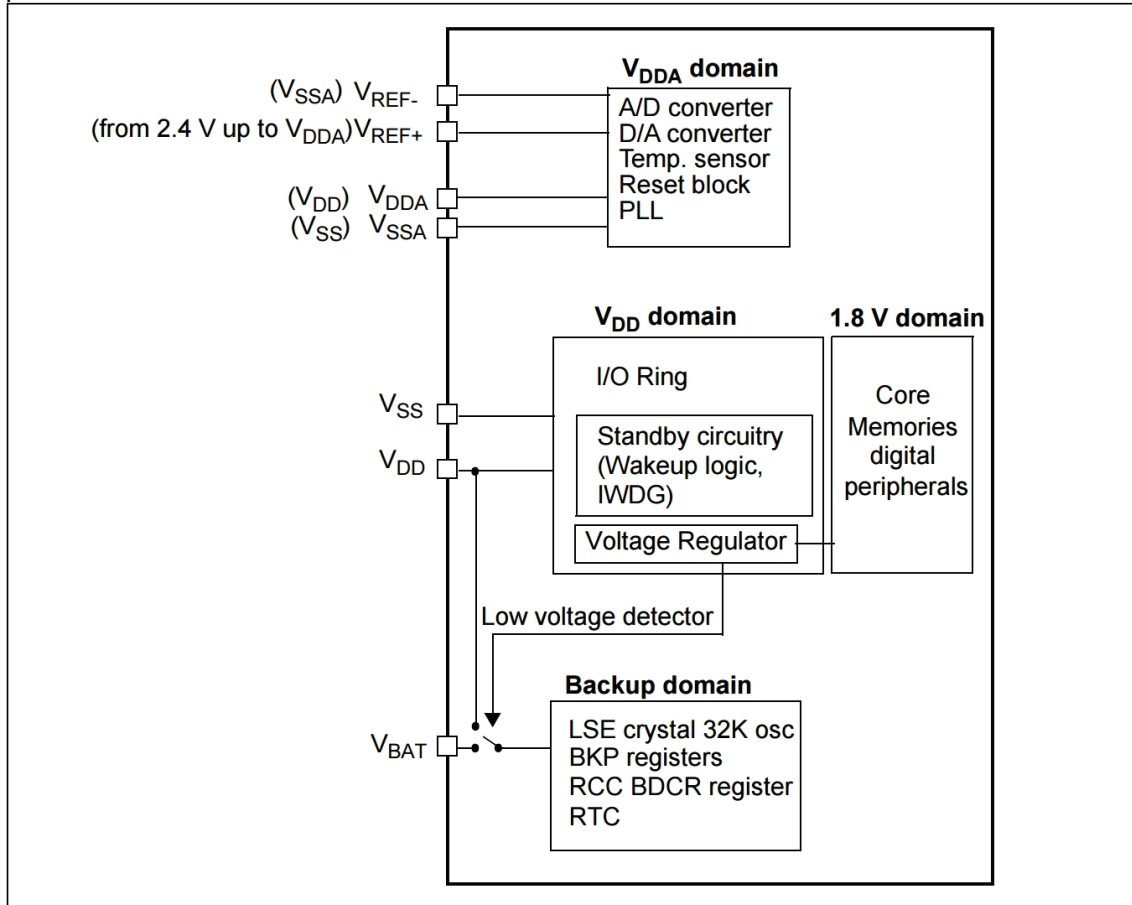


Figure3 Power Supply Block Diagram

Notes:  $V_{DDA}$  and  $V_{SSA}$  must be coupled to  $V_{DD}$  and  $V_{SS}$  respectively.

#### 4.1.1 Separate A/D Converter Supply and Reference Voltage

To improve conversion accuracy, the ADC is powered by a separate power supply that filters and shields it from burr interference on the printed circuit board.

- The ADC power pins are  $V_{DDA}$
- Separate power ground  $V_{SSA}$

If there is a  $V_{REF}$ -pin (depending on the package), it must be connected to  $V_{SSA}$ .

To ensure better accuracy when the input is low voltage, the user can connect a separate external reference voltage ADC to the  $V_{REF+}$  and  $V_{REF-}$  pins. The voltage range at  $V_{REF+}$  is 2.4V ~  $V_{DDA}$ .

#### 4.1.2 Battery Backup Area

Connecting to the  $V_{BAT}$  pin using a battery or other power source preserves the contents of the backup registers and maintains the function of the RTC when  $V_{DD}$  is powered down.

The  $V_{BAT}$  pin supplies power to the RTC, LSE oscillator, and ports PC13 through PC15, which ensures that the RTC will continue to operate when the main power supply is cut off. The switch to switch to  $V_{BAT}$  power supply is controlled by the power-down reset function in the reset module.

---

**Warning:**

The power switch between VBAT and VDD remains connected to VBAT during the VDD rise phase (tRSTTEMPO) or after a PDR (power-down reset) is detected.

During the VDD rise phase, current may be injected into VBAT through the internal diode between VDD and VBAT if VDD reaches a steady state in less than tRSTTEMPO (refer to the relevant section in the datasheet for the value of tRSTTEMPO) and  $VDD > VBAT + 0.6V$ .

If the power supply or battery connected to the VBAT cannot withstand such an injection current, it is highly recommended to connect a low dropout diode between the external VBAT and the power supply.

---

If an external battery is not available in the application, it is recommended that the VBAT be connected externally to VDD and a 100nF ceramic filter capacitor.

When the backup area is powered by VDD (internal analog switch connected to VDD), the following functions are available:

- PC14 and PC15 can be used for GPIO or LSE pins
- The PC13 can be used as a general-purpose I/O port, a TAMPER pin, an RTC calibration clock, an RTC alarm clock, or a seconds output (see Chapter 5: Backup Registers (BKP).)

*Notes: Because the analog switches can only pass a small amount of current (3mA), there are limitations to using the I/O port functions of the PC13 to PC15 in output mode: the speed must be limited to less than 2MHz, the maximum load is 30pF, and these I/O ports must never be used as a current source (e.g., to drive LEDs).*

When the backup area is powered by VBAT (analog switch connected to VBAT after VDD disappears), the following function can be used:

- PC14 and PC15 can only be used on the LSE pin
- PC13 can be used as a TAMPER pin, RTC alarm clock, or seconds output (see Section 5.4.2 : RTC Clock Calibration Register (BKP\_RTCCR))

### 4.1.3 Voltage Regulator

The regulator is always enabled after reset. Depending on the application it works in 3 different modes.

- Running Mode: the regulator provides 1.8V power (core, memory and peripherals) in normal power mode.
- Stop Mode: the regulator provides a 1.8V supply in low power mode to save the contents of the registers and SRAM.
- Standby mode: the regulator is de-energized. The contents of the registers and SRAM are all lost except for the standby circuit and backup domain.

## 4.2 Power Manager

### 4.2.1 Power-On Reset (POR) and Power-Down Reset (PDR)

The W55MH32 has a complete internal power-on reset (POR) and power-down reset (PDR) circuitry that allows the system to operate normally when the supply voltage reaches 2V.

When VDD/VDDA falls below the specified limit voltage VPOR/VPDR, the system remains in the reset state without the need for an external reset circuit. Refer to the Electrical Characteristics section of the datasheet for details on power-on reset and power-down reset.

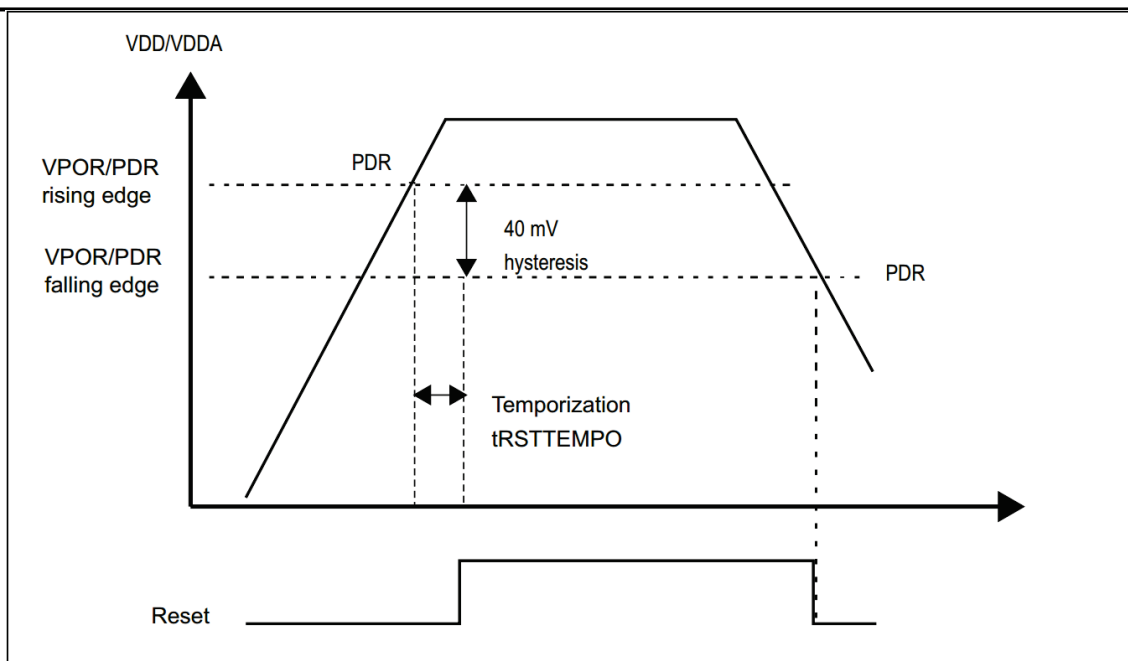


Figure4 Power-on reset and power-down reset waveforms

## 4.2.2 Programmable Voltage Monitor (PVD)

The user can monitor the power supply by using PVD to compare the VDD voltage with the PLS[2:0] bits in the Power Control Register (PWR\_CR), which select the threshold value for monitoring the voltage.

PVD is enabled by setting the PVDE bit.

The PVDO flag in the Power Control/Status Register (PWR\_CSR) is used to indicate whether VDD is above or below the voltage threshold for PVD. This event is internally connected to line 16 of the external interrupt and generates an interrupt if the interrupt is enabled in the external interrupt register. A PVD interrupt is generated when VDD falls below the PVD threshold and/or when VDD rises above the PVD threshold, depending on the rising/falling edge trigger setting of line 16 of the external interrupt. This feature can be used, for example, for performing emergency shutdown tasks.

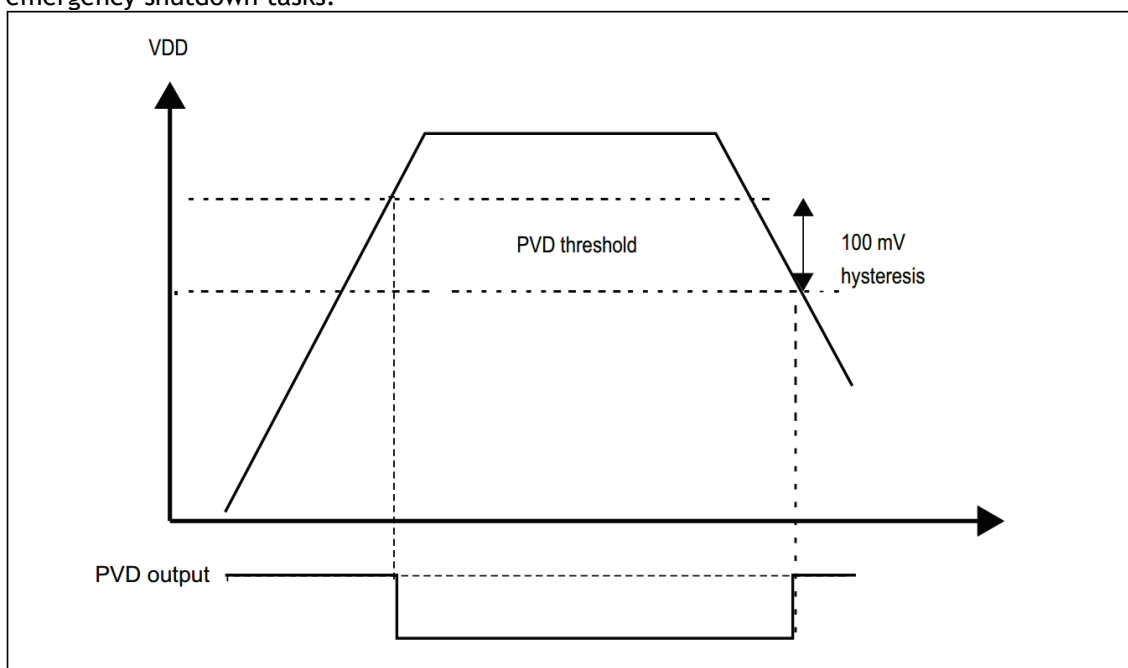


Figure5 Threshold for PVD

## 4.3 Low Power Mode

After a system or power reset, the microcontroller is in a running state. Various low-power modes can be utilized to save power when the CPU does not need to continue running, such as when waiting for an external event. The user needs to select an optimal low-power mode based on conditions such as lowest power consumption, fastest boot time, and available wake-up sources.

Three low-power modes:

- Sleep mode (Cortex™ -M3 core is stopped, all peripherals including those of the Cortex-M3 core, such as NVIC, system clock (SysTick), etc. are still running)
- Stop mode (all clocks are stopped)
- Standby mode (1.8V power off)

In addition, in the operation mode, power consumption can be reduced in one of the following ways:

- Reduced system clock
- Turns off unused peripheral clocks on the APB and AHB buses.

Table5 Low Power Mode List

paradigm	go into	awakens	For the 1.8V region Impact of the clock	Impact on V <sub>DD</sub> area clocks	voltage regulator
sleep (SLEEP-NOW or SLEEP-ON-EXIT)	WFI	Any interruption	CPU clock off, no effect on other clocks and ADC clocks	not have	write out (a prescription, check, invoice etc)
	WFE	wake-up event			
(of a prepaid mobile phone) be out of credit	PDDS and LPDS bits + SLEEPDEEP bits + WFI or WFE	Any external interrupt (set in the external interrupt register)	Turn off all clocks in the 1.8V region	Oscillator shutdown for HSI and HSE	On or in low power mode (according to the setting of the power control register (PWR_CR))
pragmatic	PDDS bit + SLEEPDEEP bit + WFI or WFE	Rising edge on WKUP pin, RTC alarm event, external reset on NRST pin, IWDG reset			mountain pass

### 4.3.1 Reduced System Clock

In Run mode, any of the system clocks (SYSCLK, HCLK, PCLK1, PCLK2) can be slowed down by programming the prescaler registers. The prescaler can also be utilized to lower the peripheral clocks before entering sleep mode. See 6.3.2for detailsSection : Clock Configuration Register (RCC\_CFGR) .

### 4.3.2 Control of External Clock

In Run mode, power consumption can be reduced at any time by stopping the clocks (HCLK and PCLKx) for the peripherals and memory. To reduce power consumption even more in sleep mode, all peripheral clocks can be turned off before executing a WFI or WFE instruction.

The clocks of each peripheral module are switched on and off by setting the AHB peripheral clock enable register (RCC\_AHBENR), the APB2 peripheral clock enable register (RCC\_APB2ENR), and the APB1 peripheral clock enable register (RCC\_APB1ENR).

### 4.3.3 Sleep Mode

#### Go to Sleep Mode

The sleep state is entered by executing the WFI or WFE instruction. Depending on the value of the SLEEPONEXIT bit in the Cortex™ -M3 System Control Register, two options are available to select the sleep mode entry mechanism:

- SLEEP-NOW: If the SLEEPONEXIT bit is cleared, the microcontroller enters sleep mode immediately when WFI or WFE is executed.

- **SLEEP-ON-EXIT:** If the SLEEPONEXIT bit is set, the microcontroller enters sleep mode immediately when the system exits from the lowest priority interrupt handler. In sleep mode, all I/O pins remain in the same state as they were in run mode. Refer to Table 6 and Table 7 for more details on how to enter sleep mode.

### Exit Sleep Mode

If the WFI instruction is executed to enter sleep mode, any one of the peripheral interrupts responded to by the Nested Vector Interrupt Controller can wake the system from sleep mode. If the WFE instruction is executed to enter sleep mode, the microprocessor will exit from sleep mode whenever a wake-up event occurs. A wake-up event can be generated in the following ways:

- Enable an interrupt in the peripheral control register, not in the NVIC (Nested Vector Interrupt Controller), and enable the SEVONPEND bit in the Cortex-M3 system control register. The peripheral interrupt pending bit and the peripheral NVIC interrupt channel pending bit (in the NVIC interrupt clear pending register) must be cleared when the MCU wakes up from WFE.
- Configure an external or internal EXIT line for event mode. When the MCU wakes up from WFE, it is not necessary to clear the peripheral's interrupt pending bit or the peripheral's NVIC interrupt channel pending bit because the pending bit corresponding to the event line is not set.

This mode takes the least amount of time to wake up because there is no time lost on interrupt entry or exit. Refer to Table 6 and Table 7 for more details on how to exit sleep mode.

Table 6 SLEEP-NOW Mode

SLEEP-NOW mode	instructions
go into	Execute the WFI (wait for interrupt) or WFE (wait for event) instruction under the following conditions: - SLEEPDEEP=0 and - SLEEPONEXIT=0 Refer to the Cortex-M3 System Control Register at .
abort	If the WFI is executed enter sleep mode: Interrupts: refer to the interrupt vector table (Table 51) If the WFE is executed enter sleep mode: Wake-up events: refer to Wake-up event management (section 8.2.3 )

Table 7 SLEEP-ON-EXIT Modes

SLEEP-ON_EXIT mode	instructions
go into	Execute the WFI command under the following conditions: - SLEEPDEEP=0 and - SLEEPONEXIT=1 Reference Cortex™ -M3 System Control Registers
abort	Interrupts: refer to the interrupt vector table (Table 51)
wake-up delay	not have

## 4.3.4 Stop Mode

Stop mode is a combination of peripheral clock control mechanisms based on the deep sleep mode of the Cortex™ -M3. In stop mode the voltage regulator can operate in normal or low power mode. All clocks in the 1.8V supply area are stopped, PLL, HSI and HSERC oscillator functions are disabled, and SRAM and register contents are preserved. In stop mode, all I/O pins remain in the same state as they were in run mode.

### Entering Stop Mode

See Table 8 for details on how to enter stop mode.

In stop mode, more power consumption can be reduced by setting the LPDS bit of the Power Control Register (PWR\_CR) to put the internal regulator into low-power mode.

If flash programming is in progress, the system does not enter stop mode until access to memory is complete.

If an access to the APB is in progress, the system does not enter stop mode until the access to the APB is complete. The following functions can be selected by programming separate control bits:

- **Independent Watchdog (IWDG):** the IWDG can be started by writing to the watchdog's key register or by hardware selection. once the independent watchdog has been started, it cannot be stopped again except by a system reset. See Section 17.3 for details.

- Real Time Clock (RTC): set via the RTCEN bit of the Backup Domain Control Register (RCC\_BDCR).
- Internal RC oscillator (LSIRC): set via the LSION bit of the control/status register (RCC\_CSR).
- External 32.768kHz oscillator (LSE): set via the LSEON bit of the Backup Domain Control Register (RCC\_BDCR).

In stop mode, if the ADC and DAC are not turned off before entering this mode, these peripherals still consume current. These 2 peripherals can be turned off by setting the ADON bit of register ADC\_CR2 and the ENx bit of register DAC\_CR to zero.

#### Exit Stop Mode

See the following table for details on how to exit stop mode.

The HSIRC oscillator is selected as the system clock when an interrupt or wake-up event causes an exit from stop mode.

When the voltage regulator is in low power mode, there will be an additional start-up delay when the system exits from stop mode. If the internal regulator is kept on during stop mode, the exit startup time will be shorter, but the corresponding power consumption will increase.

Table8 Stop Mode

Stop Mode	instructions
go into	Execute the WFI (wait for interrupt) or WFE (wait for event) instruction under the following conditions: -Sets the SLEEPDEEP bit in the Cortex-M3 system control registers. -clears the PDDS bit in the Power Control Register (PWR_CR) -Selects the mode of the voltage regulator by setting the LPDS bit in PWR_CR Note: In order to enter stop mode, all request bits for external interrupts (pending register (EXTI_PR)) and the alarm flag of the RTC must be cleared, otherwise the stop mode entry process will be skipped and the program continues to run.
abort	If the WFI is executed enter the stop mode: Set any external interrupt line to interrupt mode (the corresponding external interrupt vector must be enabled in the NVIC). See the interrupt vector table (Table51 ). If the WFE is executed enter the stop mode: Set any external interrupt line to event mode. See Wake-Up Event Management (Section8.2.3 ).
wake-up delay	HSIRC Wake-Up Time+ Time for the voltage regulator to wake up from low power.

### 4.3.5 Standby Mode

Standby mode enables the lowest power consumption of the system. This mode turns off the voltage regulator while the Cortex-M3 is in deep sleep mode. The entire 1.8V supply area is powered down. the PLL, HSI, and HSE oscillators are also powered down. the SRAM and register contents are lost. Only the backup registers and standby circuitry maintain power (seeFigure3 ).

#### Entering Standby Mode

SeeTable9 for details on how to enter standby mode.

The following standby mode functions can be selected by setting separate control bits:

- Independent Watchdog (IWDG): The IWDG can be started by writing to the watchdog's key register or by hardware selection. once the independent watchdog has been started, it cannot be stopped again except by a system reset. See section18.3 for details.
- Real Time Clock (RTC): set via the RTCEN bit of the Backup Region Control Register (RCC\_BDCR).
- Internal RC oscillator (LSIRC): set via the LSION bit of the control/status register (RCC\_CSR).
- External 32.768kHz oscillator (LSE): set via the LSEON bit of the Backup Region Control Register (RCC\_BDCR).

#### Exit Standby Mode

The microcontroller exits from standby mode when an external reset (NRST pin), an IWDG reset, a rising edge on the WKUP pin, or a rising edge of an RTC alarm event occurs (seeFigure175 : Simplified RTC Block Diagram). Upon wakeup from standby, all registers are reset except the power control/status register (PWR\_CSR) (see Section4.4.2 ).

Code execution after waking up from standby mode is equivalent to execution after a reset (sampling boot mode pins, reading reset vectors, etc.). The Power Control/Status Register (PWR\_CSR) (see Section4.4.2 ) will instruct the core to exit from standby.

See the following table for details on how to exit standby mode.

Table9 Standby Modes

standby mode	clarification
go into	Execute the WFI (wait for interrupt) or WFE (wait for event) instruction under the following conditions: - Setting the SLEEPDEEP bit in the Cortex™ -M3 system control registers - Setting the PDDS bit in the Power Control Register (PWR_CR) - Clear the WUF bit in the power control/status register (PWR_CSR)
abort	Rising edge of WKUP pin, rising edge of RTC alarm event, external reset on NRST pin, IWDG reset.
wake-up delay	Startup of the voltage regulator during the reset phase.

### Input/Output Port Status in Standby Mode

In standby mode, all I/O pins are in the high resistance state, except for the following pins:

- Reset pin (always active)
- TAMPER pin when set to anti-intrusion or calibration outputs
- Enabled wake-up pins

### Debug Mode

By default, if the microprocessor is put into stop or standby mode while debugging the microprocessor, the debug connection will be lost. This is because the Cortex™ -M3 core loses its clock.

However, by setting certain configuration bits in the DBGMCU\_CR register, it is possible to debug the software while using low-power mode. Refer to Section31.16.1 : Debugging Support for Low Power Mode for more details.

## 4.3.6 Automatic Wake-up in Low-Power Mode (AWU)

The RTC can wake up the microcontroller in low-power mode without relying on external interrupts (automatic wake-up mode). The RTC provides a programmable time base for periodic wake-up from stop or standby mode. Two of the three RTC clock sources can be selected to implement this function by programming the RTCSEL[1:0] bits of the Backup Region Control Register (RCC\_BDCR).

- Low power consumption 32.768kHz external crystal (LSE)

The clock source provides a low-power and accurate time reference. (Consumption is less than 1μA in typical cases).

- Low Power Internal RC Oscillator (LSIRC)

Using this clock source saves the cost of a 32.768kHz crystal. But the RC oscillator will increase the power consumption a little. In order to wake up the system from stop mode with an RTC alarm event, the following must be done:

- Configure external interrupt line 17 for rising edge triggering.
- Configure the RTC so that it can generate RTC alarm events.

It is not necessary to configure the external interrupt line 17 if you want to wake up from standby mode.

## 4.4 Power Control Register

These peripheral registers can be operated in half-word (16-bit) or word (32-bit) format.

### 4.4.1 Power Control Register (PWR\_CR)

Address offset: 0x00

Reset value: 0x0000 0000 (cleared on wake-up from standby mode)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							DBP	PLS[2:0]			PVDE	CSBF	CWUF	PDDS	LPDS
							rw	rw	rw	rw	rw	rc_w1	rc_w1	rw	rw



Bit	notation	clarification	
31:9	Reserved	Reserved	
8	DBP	<b>DBP:</b> Remove write protection from the backup area After a reset, the RTC and Backup registers are in a protected state to prevent accidental writes. Set this to allow writes to these registers. 0: Write to RTC and back-up registers disabled 1: Writes to the RTC and Backup Registers are allowed Note: If the RTC is clocked to HSE/128, this bit must be held to '1'.	
7:5	PLS	<b>PLS[2:0]:</b> PVD level selection These bits are used to select the voltage threshold for the supply voltage monitor 000: 2.2V 001: 2.3V 010: 2.4V 011: 2.5V	100: 2.6V 101: 2.7V 110: 2.8V 111: 2.9V Note: See the Electrical Characteristics section of the datasheet for a detailed description.
4	PVDE	<b>PVDE:</b> Power supply voltage monitor (PVD) enable 0: PVD disabled 1: Turn on PVD	
3	CSBF	<b>CSBF:</b> Clear standby bit always reads 0 0: No effect 1: Clear SBF standby bit (write)	
2	CWUF	<b>CWUF:</b> Clear Wake-Up Bit Always Read Out to 0 0: No effect 1: Clear WUF wake-up bit after 2 system clock cycles (write)	
1	PDDS	<b>PDDS:</b> Power-Down Deep Sleep and LPDS Bit Co-Operation 0: Enter shutdown mode when the CPU enters deep sleep and the state of the regulator is controlled by the LPDS bit. 1: The CPU enters standby mode when it enters deep sleep.	
0	LPDS	<b>LPDS:</b> Low power consumption under deep sleep PDDS=0 operates in concert with the PDDS bit 0: Voltage regulator on in shutdown mode 1: In shutdown mode the voltage regulator is in low power mode	

#### 4.4.2 Power Control/Status Register (PWR\_CSR)

Address offset: 0x04

Reset value: 0x0000 0000 (not cleared when waking up from standby mode)

Reading this register requires an additional APB cycle compared to a standard APB read

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

Bit	notation	clarification	
31:9	Reserved	Reserved	
8	EWUP	<b>EWUP:</b> Enable WKUP pin 0: WKUP pin is general purpose I/O. Events on the WKUP pin cannot wake up the CPU from standby mode 1: The WKUP pin is used to wake up the CPU from standby mode. The WKUP pin is forced to an input pull-down configuration (a rising edge on the WKUP pin wakes up the system from standby mode). Note: This bit is cleared during a system reset.	
7:3	Reserved	Reserved. Always reads 0.	
2	PVDO	<b>PVDO:</b> PVD output This bit is valid when PVD is enabled by the PVDE bit 0: VDD/VDDA above the PVD threshold value selected by PLS[2:0] 1: VDD/VDDA is below the PVD threshold value selected by PLS[2:0]. Note: In standby mode PVD is stopped. Therefore, after standby mode or after a reset, this bit is 0 until the PVDE bit is set.	
1	SBF	<b>SBF:</b> standby flag This bit is set by hardware and can only be cleared by POR/PDR (Power On/Power Off Reset) or by setting the CSBF bit of the Power Control Register (PWR_CR). 0: System is not in standby mode 1: System enters standby mode	
0	WUF	<b>WUF:</b> Wake Up Flag This bit is set by hardware and can only be cleared by POR/PDR (Power On/Power Off Reset)	

		or by setting the CWUF bit of the Power Control Register (PWR_CR). 0: No wakeup event occurred 1: A wake-up event occurs on the WKUP pin or an RTC alarm event occurs. Note: When the WKUP pin is already high, an additional event is detected when (by setting the EWUP bit) the WKUP pin is enabled.
--	--	--

### 4.4.3 PWR Register Address Image

The following table lists all PWR registers.

Table10 PWR Register Address Map and Reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h	PWR_CR	Reserved																							DBP	PLS [2:0]		PVDE		CSBF	CWUF	PDDS	LPDS
	Reset value																								0	0	0	0	0	0	0	0	0
004h	PWR_CSR	Reserved																							EWUP	Reserved				PVDD		SBF	WUF
	Reset value																								0					0	0	0	

SeeTable1 for register start addresses.

## 5 Backup Register (BKP)

### 5.1 Introduction to BKP

The backup registers are 42 16-bit registers that can be used to store 84 bytes of user application data. They are in the backup domain, and when the VDD power is cut off, they are still maintained powered by VBAT. They are also not reset when the system is woken up in standby mode, or when the system is reset or power is reset.

In addition, the BKP control register is used to manage the intrusion detection and RTC calibration functions.

After reset, access to the backup registers and RTC is disabled and the backup domain is protected against possible accidental write operations. Perform the following operations to enable access to the backup registers and RTC.

- Turn on the clock for the power and backup interfaces by setting the PWREN and BKPEN bits of register RCC\_APB1ENR.
- The DBP bit of the Power Control Register (PWR\_CR) is used to enable access to the Backup Register and RTC.

### 5.2 BKP Characteristics

- Data Backup Register.
- Status/control registers for managing anti-intrusion detection with interrupt capability
- A checksum register used to store the RTC checksum value.
- Output RTC calibration clock, RTC alarm pulse or seconds pulse on PC13 pin (when this pin is not used for intrusion detection).

### 5.3 BKP Functional Description

#### 5.3.1 Intrusion Detection

When the signal on the TAMPER pin changes from '0' to '1' or from '1' to '0' (depending on the TPAL bit of the Backup Control Register BKP\_CR), an Intrusion Detection event is generated. The Intrusion Detection event clears the contents of all data backup registers.

However, to avoid losing the intrusion event, the intrusion detection signal is the logical sum of the signal detected at the edge and the intrusion detection allow bit, so that an intrusion event that occurs before the intrusion detection pin is allowed can also be detected.

- When TPAL=0: If the TAMPER pin is already high before the intrusion detection TAMPER pin is activated (by setting the TPE bit), an additional intrusion event is generated once the intrusion detection function is activated (even though no rising edge occurs after TPE position '1').
- When TPAL=1: If this pin is already low before activating the intrusion detection pin TAMPER (by setting the TPE bit), an additional intrusion event is generated once the intrusion detection function is activated (even though no falling edge occurs after the TPE position '1'). Setting the TPIE bit of the BKP\_CSR register to '1' generates an interrupt when an intrusion event is detected.

After an intrusion event has been detected and cleared, the intrusion detection pin TAMPER should be disabled. The intrusion detection function is then re-enabled with the TPE bit before writing to the backup data register again. This prevents software from writing to the backup data register while there is still an intrusion event on the intrusion detection pin. This is equivalent to level detecting the intrusion pin TAMPER.

Notes: *The intrusion detection function remains active when the VDD supply is disconnected. To avoid unnecessary reset of the data backup register, the TAMPER pin should be connected off-chip to the correct level.*

#### 5.3.2 RTC Calibration

To facilitate measurements, the RTC clock can be output via a 64-division frequency to the intrusion detection pin TAMPER. This feature is enabled by setting the CCO bit of the RTC checksum register (BKP\_RTCCR).

By configuring the CAL[6:0] bits, this clock can be slowed down by up to 121 ppm.

## 5.4 BKP Register Description

For the abbreviations used in the register descriptions, refer to Section1 .  
These peripheral registers can be operated in half-word (16-bit) or word (32-bit) format.

### 5.4.1 Backup Data Register x (BKP\_DRx) (x=1..10)

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

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit	notation	clarification													
15:0	D[15:0]	<b>D[15:0]:</b> Backup data These bits can be used to write user data. Note: The BKP_DRx register will not be reset by a system reset, power reset, or wake-up from standby mode. They can be reset by a backup domain reset or (if the Intrusion Detection Pin TAMPER function is enabled) by an intrusion pin event.													

### 5.4.2 RTC Clock Calibration Register (BKP\_RTCCR)

Address offset: 0x2C

Reset value: 0x00000000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						ASOS	ASOE	CCO	CAL[6:0]						
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit	notation	clarification													
15:8	Reserved	Reserved, always reads 0. startup) is reset by an intrusion pin event.													
9	ASOS	<b>ASOS:</b> Alarm or second output selection When the ASOE bit is set, the ASOS bit can be used to select whether the output on the TAMPER pin is an RTC seconds pulse or an alarm clock pulse signal. 0: Output RTC alarm pulse 1: Output second pulse Note: This bit can only be cleared by a reset of the backup area													
8	ASOE	<b>ASOE:</b> Allow output of alarm or second pulse (Alarm or second output enable) Depending on the setting of the ASOS bit, this bit allows an RTC alarm or seconds pulse to be output to the TAMPER pin. The width of the output pulse is one cycle of the RTC clock. TAMPER cannot be enabled when the ASOE bit is set. Note: This bit can only be cleared by a reset of the backup area													
7	CCO	<b>CCO:</b> Calibration clock output 0: no effect 1: This position 1 allows the output of a 64-division RTC clock at the intrusion detection pin. When the CCO is in position 1, the intrusion detection function must be turned off to avoid detecting useless intrusion signals. Note: This bit is cleared when the VDD supply is disconnected.													
6:0	CAL	<b>CAL[6:0]:</b> Calibration value The calibration value indicates how many clock pulses will be skipped within every 220 clock pulses. This can be used to calibrate the RTC to slow down the clock by a ratio of 1000000/220ppm. The RTC clock can be slowed down by 0 to 121 ppm.													

### 5.4.3 Backup Control Register (BKP\_CR)

Offset address: 0x30

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													TPAL	TPE	
													rw	rw	
Bit	notation	clarification													
15:2	Reserved	Reserved, always reads 0.													
1	TPAL	<b>TPAL:</b> Intrusion detection TAMPER pin active level (TAMPERpinactivelevel) 0: Intrusion Detection A high level on the TAMPER pin clears all data backup registers (if the TPE bit is 1)													

		1: Intrusion Detection A low level on the TAMPER pin clears all data backup registers (if the TPE bit is 1)
0	TPE	TPE: Startup Intrusion Detection TAMPER pin (TAMPERpinenable) 0: Intrusion detection TAMPER pin used as general purpose IO port 1: Turn on the intrusion detection pin to be used as intrusion detection

Notes: It is always safe to set both the TPAL and TPE bits at the same time. However, clearing both at the same time will generate a false intrusion event. Therefore, it is recommended to change the state of the TPAL bit only when TPE is 0.

## 5.4.4 Backup Control/Status Register (BKP\_CSR)

Offset address: 0x34

Reset value: 0x00000000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						TIF	TEF	Reserved				TPIE	CTI	CTE	
						r	r					rw	w	w	

Bit	notation	clarification
15:10	Reserved	Reserved, always reads 0.
9	TIF	TIF: Trespass Interrupt Flag (Tamper interrupt flag) This bit is set by hardware when an intrusion event is detected and the TPIE bit is 1. This flag bit is cleared by writing a 1 to the CTI bit (which also clears the interrupt). If the TPIE bit is cleared, this bit is also cleared. 0: No intrusive interruptions 1: Generation of intrusive interruptions Note: Reset this bit only after a system reset or wake-up from standby mode.
8	TEF	TEF: Tamper event flag This bit is set by hardware to 1 when an intrusion event is detected, and can be cleared by writing a 1 to the CTE bit. 0: No intrusion 1: Intrusion detected Note: An intrusion event resets all BKP_DRx registers. All BKP_DRx registers remain reset as long as TEF is 1. When this bit is set to 1, if a write operation is performed to the BKP_DRx, the value written will not be saved.
7:3	Reserved	Reserved, always reads 0.
2	TPIE	TPIE: TAMPER pin interrupt enable 0: Disable intrusion detection interrupt 1: Allow intrusion detection interrupt (TPE bit of BKP_CR register must also be set to 1) Note 1: Intrusive interrupts cannot wake up the system core from low-power mode. Note 2: Reset this bit only when the system is reset or woken up from standby mode.
1	CTI	CTI: Clear tamper interrupt (Clear tamper interrupt) This bit can only be written to, and the read value is 0. 0: Invalid 1: Clear Intrusion Detection Interrupt and TIF Intrusion Detection Interrupt flags
0	CTE	CTE: Clear tamper event This bit can only be written to, and the read value is 0. 0: Invalid 1: Clear the TEF intrusion detection event flag (and reset the intrusion detector).

## 5.4.5 BKP Register Image

The BKP register is a 16-bit addressable register.

Table11 BKP register image and Reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h	Reserved																																
004h	BKP_DR1	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
008h	BKP_DR2	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
00Ch	BKP_DR3	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
010h	BKP_DR4	Reserved																D[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
014h	BKP_DR5	Reserved																D[15:0]																
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
018h	BKP_DR6	Reserved																D[15:0]																
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
01Ch	BKP_DR7	Reserved																D[15:0]																
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
020h	BKP_DR8	Reserved																D[15:0]																
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
024h	BKP_DR9	Reserved																D[15:0]																
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
028h	BKP_DR10	Reserved																D[15:0]																
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
02Ch	BKP_RTCCR	Reserved																						ASOS	ASOE	OCO	CAL[6:0]							
	Reset value																							0	0	0	0	0	0	0	0	0	0	0
030h	RTC_CR	Reserved																										TPAL	TPE					
	Reset value																											0	0					
034h	RTC_CSR	Reserved																						TIF	TFE	Reserved						TPIE	CTI	CTE
	Reset value																							0	0							0	0	0
038h	Reserved																																	
03Ch	Reserved																																	
040h	BKP_DR11	Reserved																D[15:0]																
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
044h	BKP_DR12	Reserved																D[15:0]																
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
048h	BKP_DR13	Reserved																D[15:0]																
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
04Ch	BKP_DR14	Reserved																D[15:0]																
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
050h	BKP_DR15	Reserved																D[15:0]																
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
054h	BKP_DR16	Reserved																D[15:0]																
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
058h	BKP_DR17	Reserved																D[15:0]																
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
05Ch	BKP_DR18	Reserved																D[15:0]																
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
060h	BKP_DR19	Reserved																D[15:0]																
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
064h	BKP_DR20	Reserved																D[15:0]																
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
068h	BKP_DR21	Reserved																D[15:0]																
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
06Ch	BKP_DR22	Reserved																D[15:0]																

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
070h	BKP_DR23	Reserved																D[15:0]																		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
074h	BKP_DR24	Reserved																D[15:0]																		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
078h	BKP_DR25	Reserved																D[15:0]																		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
07Ch	BKP_DR26	Reserved																D[15:0]																		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
080h	BKP_DR27	Reserved																D[15:0]																		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
084h	BKP_DR28	Reserved																D[15:0]																		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
088h	BKP_DR29	Reserved																D[15:0]																		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
08Ch	BKP_DR30	Reserved																D[15:0]																		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
090h	BKP_DR31	Reserved																D[15:0]																		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
094h	BKP_DR32	Reserved																D[15:0]																		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
098h	BKP_DR33	Reserved																D[15:0]																		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
09Ch	BKP_DR34	Reserved																D[15:0]																		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0A0h	BKP_DR35	Reserved																D[15:0]																		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0A4h	BKP_DR36	Reserved																D[15:0]																		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0A8h	BKP_DR37	Reserved																D[15:0]																		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0ACh	BKP_DR38	Reserved																D[15:0]																		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0B0h	BKP_DR39	Reserved																D[15:0]																		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0B4h	BKP_DR40	Reserved																D[15:0]																		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0B8h	BKP_DR41	Reserved																D[15:0]																		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0BCh	BKP_DR42	Reserved																D[15:0]																		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SeeTable1 for register start addresses.



## 6 Reset and Clock Control (RCC)

### 6.1 Reset (a Dislocated Joint, an Electronic Device etc)

The W55MH32 supports three forms of reset, namely system reset, power-on reset and backup area reset.

#### 6.1.1 System Reset

A system reset will reset all registers to their reset state except for the reset flag bit in the RCC\_CSR register of the clock controller and the registers in the backup area (see Figure 3).

A system reset is generated when any of the following events occur:

1. Low level on NRST pin (external reset)
2. Window watchdog count termination (WWDG reset)
3. Independent Watchdog Count Termination (IWDG Reset)
4. Software reset (SW reset)
5. Low power management reset

The source of the reset event can be identified by looking at the reset status flag bit in the RCC\_CSR control status register.

##### Software Reset

A software reset can be achieved by setting the SYSRESETREQ position '1' in the Cortex™ -M3 Interrupt Application and Reset Control Register. Please refer to the Cortex™ -M3 Technical Reference Manual for further information.

##### Low Power Management Reset

A low-power management reset can be generated in the following two cases:

1. Generates a low-power management reset when entering standby mode:  
This reset will be enabled by setting the nRST\_STDBY position '1' in the user selection byte. At this point, even if the process of entering standby mode is performed, the system will be reset instead of entering standby mode. process, the system will be reset instead of entering standby mode.
2. Generates a low-power management reset when entering stop mode:  
This reset will be enabled by setting the nRST\_STOP position '1' in the user selection byte. At this point, even if the process of entering the shutdown mode procedure is performed, the system will be reset instead of entering shutdown mode.

#### 6.1.2 Power Reset

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

1. Power-on/power-off reset (POR/PDR reset)
2. Return from standby mode

A power reset will reset all registers except the backup area. (See Figure 3)

The reset source in the figure will eventually act on the RESET pin and remain low during the reset process. The reset entry vector is fixed at address 0x0000\_0004.

The chip's internal reset signal is output on the NRST pin, and the pulse generator ensures that each (external or internal) reset source has a pulse delay of at least 20 µs; it generates a reset pulse when the NRST pin is pulled low to generate an external reset.

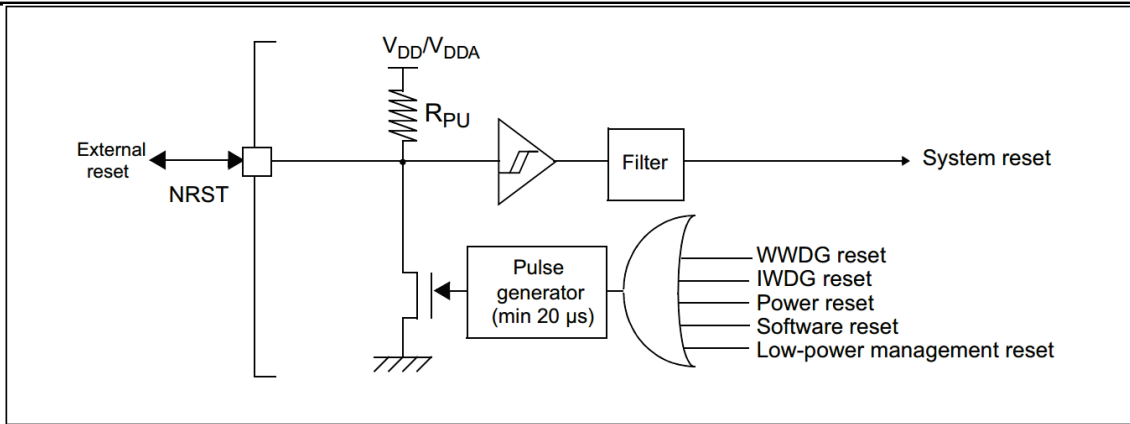


Figure6 Reset Circuit

### 6.1.3 Backup Domain Reset

The backup area has two specialized resets that affect only the backup area (see Figure 3). A backup area reset is generated when one of the following events occurs.

1. software reset, a backup area reset can be by setting the Backup Domain Control Register (RCC\_BDCR) generated BDRST bit in the (see section 6.3.9).
2. With both VDD and VBAT powered down, a VDD or VBAT power-up will trigger a backup area reset.

## 6.2 Clocks

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

- HSI Oscillator Clock
- HSE Oscillator Clock
- PLL clock

These devices have the following 2 secondary clock sources:

- 40kHz low-speed internal RC that can be used to drive a standalone watchdog and programmatically selected to drive the RTC. the RTC is used to automatically wake up the system from shutdown/standby mode.
- A 32.768kHz low-speed external crystal can also be used to drive the RTC (RTCCLK) via program selection.

When not in use, either clock source can be independently started up or shut down, thereby optimizing system power consumption.

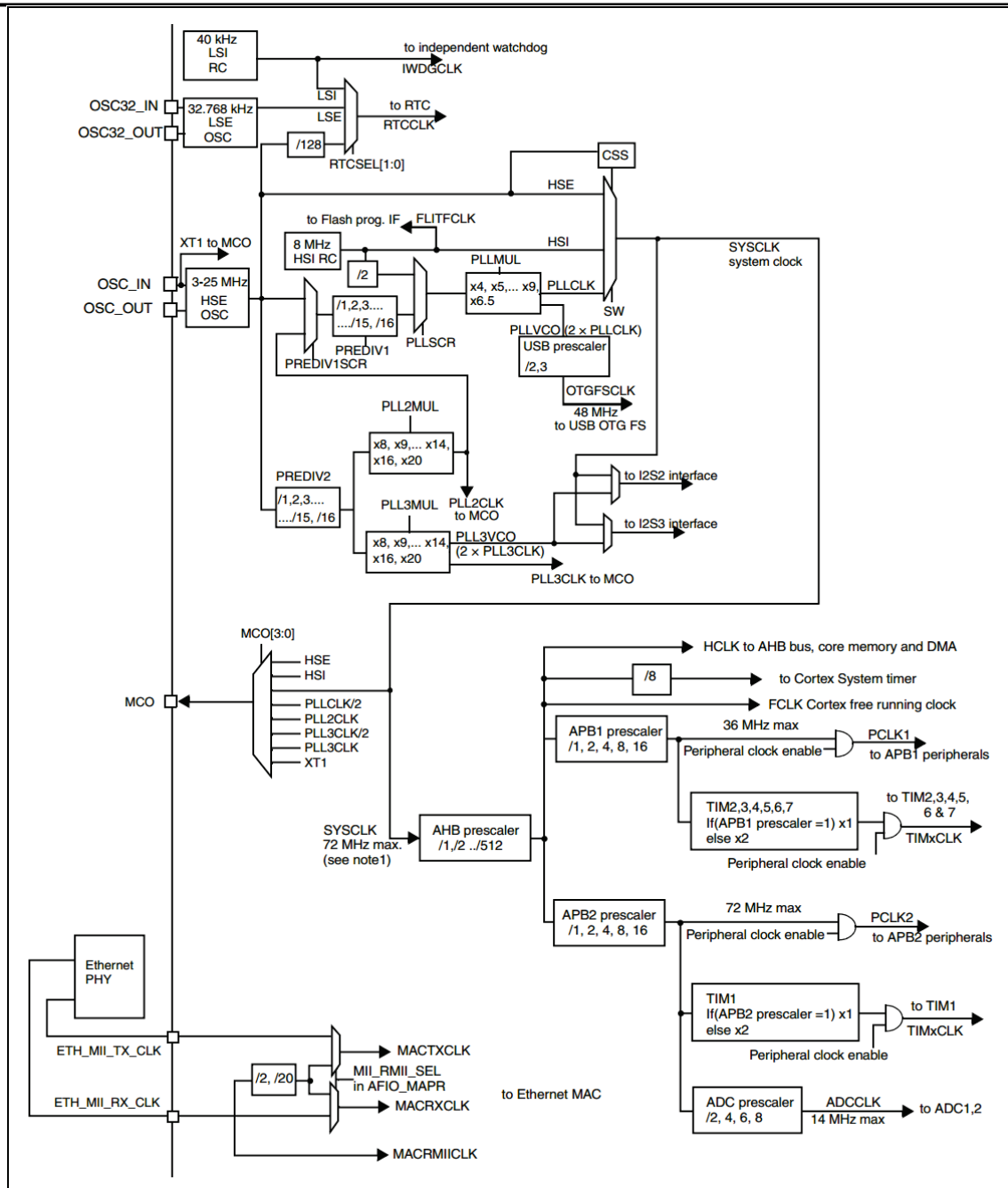


Figure7 Clock tree

1. When the HSI is used as an input to the PLL clock, the maximum frequency that can be obtained for the system clock is 108 MHz.
2. For the characteristics of the internal and external clock sources, please refer to the "Electrical Characteristics" section of the respective datasheets.

The user can configure the frequency of the AHB, high speed APB (APB2) and low speed APB (APB1) domains through multiple prescalers. The maximum frequency of the AHB and APB2 domains is 216 MHz. The maximum allowable frequency of the APB1 domain is 108 MHz. The clock frequency of the SDIO interface is fixed to HCLK/2.

The RCC is used as an external clock for the Cortex system timer (SysTick) through the AHB clock (HCLK) 8-division frequency. The above clock or the Cortex (HCLK) clock can be selected as the SysTick clock by setting the SysTick control and status registers. The ADC clock is obtained from the high-speed APB2 clock by dividing it by 2, 4, 6, or 8.

The timer clock frequency assignment is automatically set by the hardware in the following 2 cases:

1. If the corresponding APB prescaler factor is 1, the timer is clocked at the same frequency as the host APB bus.

2. Otherwise, the clock frequency of the timer is set to two times the frequency of the APB bus to which it is connected.

FCLK is the free running clock for Cortex™ -M3. See ARM's Cortex™ -M3 Technical Reference Manual for details.

## 6.2.1 HSE Clock

The high-speed external clock signal (HSE) is generated by the following two clock sources:

- HSE External Crystal/Ceramic Resonators
- HSE User External Clock

To minimize clock output distortion and shorten startup stabilization time, the crystal/ceramic resonator and load capacitor must be placed as close as possible to the oscillator pins. The load capacitance value must be adjusted according to the selected oscillator.

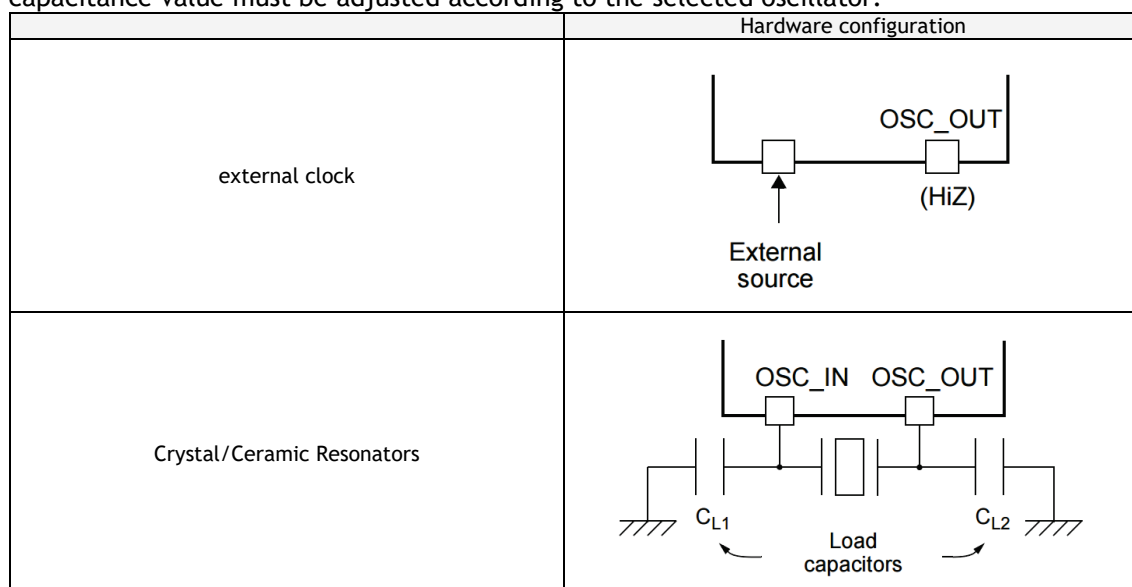


Figure8 HSE/LSE Clock Source

### External Clock Source (HSE Bypass)

In this mode, an external clock must be provided. This mode can be selected by setting the HSEBYP and HSEON bits in the clock control register. The external clock signal (50% duty cycle square, sine or triangle wave) must be connected to the SOC\_IN pin while ensuring that the OSC\_OUT pin is left open. SeeFigure8 .

### External Crystal/Ceramic Resonator (HSE Crystal)

A 4-16Mhz external oscillator provides a more accurate master clock for the system. The related hardware configuration can be found Figure8 and further information can be found in the Electrical Characteristics section of the datasheet.

The HSERDY bit in the clock control register RCC\_CR is used to indicate whether the high speed external oscillator is stable. At startup, the clock is not released until this bit is set '1' by hardware. If an interrupt is allowed to be generated in the clock interrupt register RCC\_CIR, the appropriate interrupt will be generated.

The HSE crystal can be turned on and off by setting the HSEON bit in RCC\_CR in the clock control register.

## 6.2.2 HSI Clock

The HSI clock signal is generated by an internal 8MHz RC oscillator and can be used directly as a system clock or as a PLL input after a 2-division frequency.

The HSIRC oscillator is capable of providing a system clock without the need for any external devices. It has a shorter start-up time than the HSE crystal oscillator. However, its clock frequency accuracy is poor even after calibration.

### Calibrations

The manufacturing process determines that the RC oscillator frequency will vary from chip to chip, which is why the HSI clock frequency of each chip has been factory calibrated to 1% (25° C) by ST. The factory calibration value is loaded into the HSICAL[7:0] bits of the Clock Control Register at system reset.

If the user's application is based on a different voltage or ambient temperature, this will affect the accuracy of the RC oscillator. The HSI frequency can be adjusted using the HSITRIM[4:0] bits in the clock control register.

The HSIRDY bit in the Clock Control Register is used to indicate whether the HSIRC oscillator is stable. The HSIRC output clock is not released during clock startup until this bit is set by hardware to '1.' The HSIRC can be started and shut down by the HSION bit in the Clock Control Register.

If the HSE crystal oscillator fails, the HSI clock is used as a backup clock source. Refer to section 6.2.7 Clock Safety Systems.

### 6.2.3 PLL

The internal PLL can be used to multiply the HSIRC output clock or the HSE crystal output clock. Refer to Figure 7 and the Clock Control Register.

PLL setup (selecting the HSI oscillator divided by 2 or the HSE oscillator as the PLL's input clock, and selecting the multiplication factor) must be done before it is activated. Once the PLL is activated, these parameters cannot be altered.

If PLL interrupts are allowed in the Clock Interrupt Register, an interrupt request can be generated when the PLL is ready.

If a USB interface is required for the application, the PLL must be set to output a 48 or 72 MHz clock, which is used to provide a 48 MHz USBCLK clock.

### 6.2.4 LSE Clock

The LSE crystal is a 32.768 kHz low-speed external crystal or ceramic resonator. It provides a low power and accurate clock source for real time clocks or other timing functions.

The LSE crystal is activated and deactivated by the LSEON bit in the Backup Domain Control Register (RCC\_BDCR).

LSERDY in the Backup Domain Control Register (RCC\_BDCR) indicates whether the LSE crystal oscillation is stable. During the startup phase, the LSE clock signal is not released until this bit is set '1' by hardware. If allowed in the clock interrupt register, an interrupt request can be generated.

#### External Clock Source (LSE Bypass)

An external clock source at 32.768 kHz must be provided in this mode. You can select this mode by setting the LSEBYP and LSEON bits in the Backup Domain Control Register (RCC\_BDCR). An external clock signal (square, sine, or triangle) with a 50% duty cycle must be connected to the OSC32\_IN pin while keeping the OSC32\_OUT pin free, see Figure 8.

### 6.2.5 LSI Clock

The LSIRC assumes the role of a low-power clock source that can be kept running in shutdown and standby modes to provide a clock for the standalone watchdog and auto-wakeup unit. The LSI clock frequency is approximately 40 kHz (between 30 kHz and 60 kHz). For further information please refer to the section on electrical characteristics in the datasheet.

LSIRC can be activated or deactivated by the LSION bit in the control/status register (RCC\_CSR). The LSIRDY bit in the control/status register (RCC\_CSR) indicates whether the low-speed internal oscillator is stable. This clock is not released during the startup phase until this bit is set to '1' by hardware. If allowed in the Clock Interrupt Register (RCC\_CIR), an LSI interrupt request will be generated.

#### LSI Calibration

The internal low-speed oscillator LSI can be calibrated to compensate for its frequency offset, resulting in an RTC time base with acceptable accuracy, as well as an independent watchdog dog (IWDG) timeout (when these peripherals use the LSI as a clock source).

Calibration can be achieved by measuring the LSI clock frequency using the input clock of the TIM5 (TIM5\_CLK). Measurements are guaranteed with HSE accuracy, and software can adjust the RTC's 20-bit prescaler to obtain the exact RTC clock base, as well as calculate to obtain the exact Independent Watchdog Dog (IWDG) timeout.

---

The LSI calibration procedure is as follows:

1. Open TIM5 and set channel 4 to input capture mode;
2. Set the TIM5\_CH4\_IEMAP bit of AFIO\_MAPR to '1' to internally connect the LSI to channel 4 of TIM5;
3. The LSI clock frequency is measured by TIM5 capture/compare 4 events or interrupts;
4. Set the 20-bit prescaler based on the measurement results and the desired RTC time base and independent watchdog timeout.

## 6.2.6 System Clock (SYSCLK) Selection

The HSI oscillator is selected as the system clock after a system reset. It cannot be stopped when the clock source is used as the system clock either directly or indirectly through a PLL. Switching from one clock source to another occurs only when the target clock source is ready (after a delay in the startup stabilization phase or PLL stabilization). Switching of the system clock does not occur when the selected clock source is not ready. The switching does not occur until the target clock source is ready.

Status bits in the Clock Control Register (RCC\_CR) indicate which clock is ready and which clock is currently being used as the system clock.

## 6.2.7 Clock Safety System (CSS)

The Clock Safety System can be activated via software. Once it is activated, the clock monitor is enabled after the HSE oscillator start-up delay and switched off after the HSE clock is switched off.

If the HSE clock fails, the HSE oscillator is automatically turned off and the clock failure event is sent to the brake inputs of the advanced timers (TIM1 and TIM8) and generates the Clock Safe Interrupt CSSI, which allows software to complete the rescue operation. This CSSI interrupt is connected to the NMI interrupt (non-maskable interrupt) of the Cortex™-M3.

*Notes: Once CSS is activated and the HSE clock fails, the CSS interrupt is generated and the NMI is automatically generated. The NMI will be executed continuously until the CSS interrupt pending bit is cleared. Therefore, the CSS interrupt must be cleared in the NMI handler by setting the CSSC bit in the Clock Interrupt Register (RCC\_CIR).*

If the HSE oscillator is being used directly or indirectly as the system clock, (indirectly meaning: it is being used as the PLL input clock and the PLL clock is being used as the system clock), a clock failure will result in the system clock automatically switching to the HSI oscillator while the external HSE oscillator is turned off. In the event of a clock failure, if the HSE oscillator clock (divided or undivided) is the input clock to the PLL used as the system clock, the PLL will also be turned off.

## 6.2.8 RTC Clock

The RTCCLK clock source can be provided by the HSE/128, LSE, or LSI clock by setting the RTCSEL[1:0] bits in the Backup Domain Control Register (RCC\_BDCR). This selection cannot be changed unless the backup domain is reset.

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

- If LSE is selected as the RTC clock:
  - As long as VBAT maintains power, the RTC continues to operate despite the VDD supply being cut off.
- If the LSI is selected as the Automatic Wake-Up Unit (AWU) clock:
  - The AWU status cannot be guaranteed if the VDD supply is cut off. See section LSI Clocking at 6.2.5 for details on LSI calibration.
- If the HSE clock is 128 divided and used as the RTC clock:
  - If the VDD supply is cut off or the internal voltage regulator is turned off (supply to the 1.8V domain is cut off), the RTC state is indeterminate.
  - The DPB bit of the Power Control Register (see section 4.4.1) (which removes write protection from the backup area) must be set to '1'.

## 6.2.9 Watchdog Clock

If the Independent Watchdog has been started by the hardware option or software, the LSI oscillator is forced in the on state and cannot be turned off. After the LSI oscillator has stabilized, the clock is supplied to the IWDG.

## 6.2.10 Clock Output

The microcontroller allows the output of a clock signal to an external MCO pin.

The corresponding GPIO port registers must be configured to function accordingly. The following four clock signals can be selected as the MCO clock:

- SYCLK
- HSI
- HSE
- PLL clock for division 2

Clock selection is controlled by the MCO[2:0] bits in the Clock Configuration Register (RCC\_CFGR).

## 6.3 RCC Register Description

Refer to Section1 for the abbreviations used in register descriptions.

### 6.3.1 Clock Control Register (RCC\_CR)

Offset address:0x00

Reset value:0x0000 XX83, X stands for undefined

Access: no wait state, word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved						PLL RDY	PLL ON	Reserved				CCS ON	HSE BYP	HSE RDY	HSE ON
						r	rW					rW	rW	r	rW

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HSICAL[7:0]								HSITRIM[4:0]					Reserved	HSIRDY	HSION
r	r	r	r	r	r	r	r	rW	rW	rW	rW	rW		r	rW

Bit	notation	clarification
31:26	Reserved	Reserved, always reads 0.
25	PLLRDY	<b>PLLRDY</b> : PLL clock ready flag (PLL clock ready flag) Set '1' by hardware after PLL lock. 0: PLL is not locked; 1: PLL lock.
24	PLLON	<b>PLLON</b> : PLL enable Set '1' or cleared by software. This bit is cleared by hardware when entering standby and stop modes. This bit cannot be cleared when the PLL clock is used or selected to be used as the system clock. 0: PLL off; 1: PLL enable.
23:20	Reserved	Reserved, always reads 0.
19	CSSON	<b>CSSON</b> : Clock security system enable Set '1' or cleared to zero by software to enable the clock monitor. 0: Clock monitor off; 1: If the external 4-16MHz oscillator is ready, the clock monitor turns on.
18	HSEBYP	<b>HSEBYP</b> : External high-speed clock bypass Bypass the external crystal oscillator by setting '1' or clearing it by software in debug mode. This bit can only be written if the external 4-16MHz oscillator is off. 0: External 4-16 MHz oscillator is not bypassed; 1: External 4-16MHz external crystal oscillator is bypassed.
17	HSERDY	<b>HSERDY</b> : External high-speed clock ready flag (External high-speed clock ready flag) A '1' is set by hardware to indicate that the external 4-16MHz oscillator has stabilized. After the HSEON bit is cleared, this bit requires six external 4-25MHz oscillator cycles to clear. 0: The external 4-16 MHz oscillator is not ready; 1: External 4-16MHz oscillator ready.
16	HSEON	<b>HSEON</b> : External high-speed clock enable Set '1' or cleared by software. This bit is cleared by hardware to turn off the 4-16MHz external oscillator when entering Standby and Stop modes. This bit cannot be cleared when the external 4-16MHz oscillator is used or selected to be used as the system clock. 0: HSE oscillator off; 1: HSE oscillator on.
15:8	HSICAL[7:0]	<b>HSICAL[7:0]</b> : Internal high-speed clock calibration These bits are automatically initialized at system startup





		00010: PLL4 octave output 00011: PLL5 octave output 00100: PLL6 octave output 00101: PLL7 octave output 00110: PLL8 octave output 00111: PLL9 octave output 01000: PLL10 octave output 01001: PLL11 octave output 01010: PLL12 octave output 01011: PLL13 octave output 01100: PLL14 octave output 01101: PLL15 octave output 01110: PLL16 octave output 01111: PLL16 octave output	10010: PLL19 octave output 10011: PLL20 octave output 10100: PLL21 octave output 10101: PLL22 octave output 10110: PLL23 octave output 10111: PLL24 octave output 11000: PLL25 octave output 11001: PLL26 octave output 11010: PLL27 octave output 11011: PLL28 octave output 11100: PLL29 octave output 11101: PLL30 octave output 11110: PLL31 octave output 11111: PLL32 octave output
17	PLLXTPRE	<b>PLLXTPRE:</b> HSE divider for PLL entry Set '1' or cleared '0' by software to divide the HSE and then use it as the PLL input clock. This bit can only be written when the PLL is turned off. 0: HSE without crossover 1: HSE 2-way	
16	PLLSRC	<b>PLLSRC:</b> PLL entry clock source Set '1' or cleared '0' by software to select the PLL input clock source. This bit can only be written when the PLL is turned off. 0: HSI oscillator clock divided by 2 as PLL input clock 1: The HSE clock is used as the PLL input clock.	
15:14	ADCPRE [1:0]	<b>ADCPRE[1:0]:</b> ADC pre-scaler (ADC prescaler) Set '1' or clear '0' by software to determine ADC clock frequency 00: PCLK22 divided and used as ADC clock 01: PCLK24 divided as ADC clock 10: PCLK26 divided and used as ADC clock 11: PCLK28 divided and used as ADC clock	
13:11	PPRE2 [2:0]	<b>PPRE2[2:0]:</b> high-speed APB prescaler (APB2) The prescaling factor of the high-speed APB2 clock (PCLK2) is controlled by software setting '1' or clearing '0'. 0xx: HCLK without crossover frequency 100: HCLK2 crossover 101: HCLK 4-way 110: HCLK 8-way 111: HCLK16 crossover	
10:8	PPRE1 [2:0]	<b>PPRE1[2:0]:</b> low-speed APB prescaler (APB1) The prescaling factor of the low-speed APB1 clock (PCLK1) is controlled by software setting '1' or clearing '0'. WARNING: Software must ensure that the APB1 clock frequency does not exceed 36MHz. 0xx: HCLK without crossover frequency 100: HCLK2 crossover 101: HCLK 4-way 110: HCLK 8-way 111: HCLK16 crossover	
7:4	HPRE [3:0]	<b>HPRE[3:0]:</b> AHB Prescaler The prescaling factor of the AHB clock is controlled by setting '1' or clearing '0' by software. 0xxx: SYSCLK without crossover frequency 1000: SYSCLK2 crossover frequency 1001: SYSCLK4 crossover frequency 1010: SYSCLK8 crossover frequency 1011: SYSCLK 16 crossover frequency 1100: SYSCLK64 crossover frequency 1101: SYSCLK128 crossover frequency 1110: SYSCLK256 frequency division 1111: SYSCLK512 frequency division Note: The prefetch buffer must be turned on when the preshunt factor of the AHB clock is greater than 1. See Flash Read (Section2.3.3 ) for details.	
3:2	SWS[1:0]	<b>SWS[1:0]:</b> System clock switch status (System clock switch status) Set '1' or cleared '0' by hardware to indicate which clock source is used as the system clock. 00: HSI is used as the system clock; 01: HSE as system clock; 10: PLL output is used as system clock; 11: Not available.	
1:0	SW[1:0]	<b>SW[1:0]:</b> System clock switching (System clock switch) The system clock source is selected by software by setting '1' or clearing '0'. Hardware-forced selection of the HSI as the system clock on return from stop or standby mode or in the event of a failure of the HSE that is directly or indirectly used as the system clock (if the clock security system has been activated) 00: HSI is used as the system clock; 01: HSE as system clock; 10: PLL output is used as system clock; 11: Not available.	

### 6.3.3 Clock Interrupt Register (RCC\_CIR)

Offset address: 0x08

Reset value: 0x0000 0000

Access: No wait cycle, word, half-word and byte accesses

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

Bit	notation	clarification
31:24	Reserved	Reserved, always reads 0.
23	MCO	<b>CSSC</b> : Clock security system interrupt clear Set '1' by software to clear the CSSF safety system interrupt flag bit CSSF. 0: No effect; 1: Clear the CSSF safety system interrupt flag bit.
22:21	Reserved	Reserved, always reads 0.
20	PLL RDYC	<b>PLL RDYC</b> : Clear PLL ready interrupt (PLL ready interrupt clear) Set '1' by software to clear the PLL ready interrupt flag bit PLLRDYF. 0: No effect; 1: Clear the PLL ready interrupt flag bit PLLRDYF.
19	HSERDYC	<b>HSERDYC</b> : clear HSE ready interrupt (HSE ready interrupt clear) Set '1' by software to clear the HSE ready interrupt flag bit HSERDYF. 0: No effect; 1: Clear the HSE ready interrupt flag bit HSERDYF.
18	HSIRDYC	<b>HSIRDYC</b> : clear HSI ready interrupt (HSI ready interrupt clear) Set '1' by software to clear the HSI ready interrupt flag bit HSIRDYF. 0: No effect; 1: Clear the HSI ready interrupt flag bit HSIRDYF.
17	LSERDYC	<b>LSERDYC</b> : clear LSE ready interrupt (LSE ready interrupt clear) Set '1' by software to clear the LSE ready interrupt flag bit LSERDYF. 0: No effect; 1: Clear the LSE ready interrupt flag bit LSERDYF.
16	LSIRDYC	<b>LSIRDYC</b> : clear LSI ready interrupt (LSI ready interrupt clear) Set '1' by software to clear the LSI ready interrupt flag bit LSIRDYF. 0: No effect; 1: Clear the LSI ready interrupt flag bit LSIRDYF.
15:13	Reserved	Reserved, always reads 0.
12	PLL RDYIE	<b>PLL RDYIE</b> : PLL ready interrupt enable Enable or disable the PLL ready interrupt by setting '1' or clearing '0' by software. 0: PLL ready interrupt off; 1: PLL ready interrupt enable.
11	HSERDYIE	<b>HSERDYIE</b> : HSE ready interrupt enable (HSE ready interrupt enable) Enable or disable the external 4-16MHz oscillator ready interrupt by setting '1' or clearing '0' by software. 0: HSE ready interrupt off; 1: HSE ready interrupt enable.
10	HSIRDYIE	<b>HSIRDYIE</b> : HSI ready interrupt enable (HSI ready interrupt enable) The internal 8MHz RC oscillator ready interrupt is enabled or disabled by software setting '1' or clearing '0'. 0: HSI ready interrupt off; 1: HSI ready interrupt enable.
9	LSERDYIE	<b>LSERDYIE</b> : LSE ready interrupt enable (LSE ready interrupt enable) Enable or disable the external 32kHz RC oscillator ready interrupt by setting '1' or clearing '0' by software. 0: LSE ready interrupt off; 1: LSE ready interrupt enable.
8	LSIRDYIE	<b>LSIRDYIE</b> : LSI ready interrupt enable (LSI ready interrupt enable) The internal 40kHz RC oscillator ready interrupt is enabled or disabled by software setting '1' or clearing '0'. 0: LSI ready interrupt off; 1: LSI ready interrupt enable.
7	CSSF	<b>CSSF</b> : Clock security system interrupt flag (Clock security system interrupt flag) Set '1' by hardware in the event of a fault in the external 4-16MHz oscillator clock. Cleared by software by setting the '1' CSSC bit. 0: No safety system interrupt generated by HSE clock failure; 1: Failure of the HSE clock resulted in an interruption of the clock safety system.

6:5	Reserved	Reserved, always reads 0.
4	PLLRDYF	<b>PLLRDYF</b> : PLL ready interrupt flag (PLL ready interrupt flag) Set '1' by hardware when PLL is ready and PLLRDYIE bit is set '1'. Cleared by software by setting the '1' PLLRDYC bit. 0: No clock-ready interrupt generated by PLL up-lock; 1: PLL up-lock causes clock ready interrupt.
3	HSERDYF	<b>HSERDYF</b> : HSE ready interrupt flag (HSE ready interrupt flag) Set '1' by hardware when the external low-speed clock is ready and the HSERDYIE bit is set '1'. Cleared by software by setting the '1' HSERDYC bit. 0: No clock-ready interrupt generated by external 4-16 MHz oscillator; 1: External 4-16MHz oscillator causes clock ready interrupt.
2	HSIRDYF	<b>HSIRDYF</b> : HSI ready interrupt flag (HSI ready interrupt flag) Set '1' by hardware when the internal high-speed clock is ready and the HSIRDYIE bit is set '1'. Cleared by software by setting the '1' HSIRDYC bit. 0: No clock-ready interrupt generated by the internal 8MHz RC oscillator; 1: Internal 8MHz RC oscillator causes clock ready interrupt.
1	LSERDYF	<b>LSERDYF</b> : LSE ready interrupt flag (LSE ready interrupt flag) Set '1' by hardware when the external low-speed clock is ready and the LSERDYIE bit is set '1'. Cleared by software by setting the '1' LSERDYC bit. 0: No clock ready interrupt generated by external 32kHz oscillator; 1: External 32kHz oscillator causes clock ready interrupt.
0	LSIRDYF	<b>LSIRDYF</b> : LSI ready interrupt flag (LSI ready interrupt flag) Set '1' by hardware when the internal low-speed clock is ready and the LSIRDYIE bit is set '1'. Cleared by software by setting the '1' LSIRDYC bit. 0: No clock-ready interrupt generated by the internal 40kHz RC oscillator; 1: Internal 40kHz RC oscillator causes clock ready interrupt.

### 6.3.4 APB2 Peripheral Reset Register ( RCC\_APB2RSTR )

Offset Address:0x0C

Reset value:0x0000 0000

Access: no wait cycle, word, half-word and byte accesses

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved										TIM11 RST	TIM10 RST	TIM9 RST	Reserved		
										rw	rw	rw			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC3 RST	USART1 RST	TIM8 RST	SPI1 RST	TIM1 RST	ADC2 RST	ADC1 RST	IOPG RST	IOPF RST	IOPE RST	IOPD RST	IOPC RST	IOPB RST	IOPA RST	Reserved	AFIO RST
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	res	rw

Bit	notation	clarification
31:22	Reserved	Reserved, always reads 0.
21	TIM11RST	<b>TIM8RST</b> : TIM11 timer reset Set '1' or clear '0' by software 0: No effect; 1: Reset the TIM11 timer.
20	TIM10RST	<b>TIM10RST</b> : TIM10 timer reset Set '1' or clear '0' by software 0: No effect; 1: Reset the TIM10 timer.
19	TIM9RST	<b>TIM9RST</b> : TIM9 timer reset (TIM8 timer reset) Set '1' or clear '0' by software 0: No effect; 1: Reset the TIM9 timer.
18:16	Reserved	Reserved, always reads 0.
15	ADC3RST	<b>ADC3RST</b> : ADC3 interface reset Set '1' or clear '0' by software 0: No effect; 1: Reset the ADC3 interface.
14	USART1RST	<b>USART1RST</b> : USART1 reset Set '1' or clear '0' by software 0: No effect; 1: Reset USART1.
13	TIM8RST	<b>TIM8RST</b> : TIM8 timer reset Set '1' or clear '0' by software 0: No effect; 1: Reset the TIM8 timer.
12	SPI1RST	<b>SPI1RST</b> : SPI1 reset Set '1' or clear '0' by software 0: No effect;

		1: Reset SPI1.
11	TIM1RST	<b>TIM1RST:</b> TIM1 timer reset (TIM1 timer reset) is set to '1' or cleared to '0' by software. 0: No effect; 1: Reset the TIM1 timer.
10	ADC2RST	<b>ADC2RST:</b> ADC2 interface reset Set '1' or clear '0' by software 0: No effect; 1: Reset the ADC2 interface.
9	ADC1RST	<b>ADC1RST:</b> ADC1 interface reset Set '1' or clear '0' by software 0: No effect; 1: Reset the ADC1 interface.
8	IOPGRST	<b>IOPGRST:</b> IO portG reset Set '1' or clear '0' by software 0: No effect; 1: Reset IO port G.
7	IOPFRST	<b>IOPFRST:</b> IO portF reset Set '1' or clear '0' by software 0: No effect; 1: Reset IO port F.
6	IOPERST	<b>IOPERST:</b> IO portE reset Set '1' or clear '0' by software 0: No effect; 1: Reset IO port E.
5	IOPDRST	<b>IOPDRST:</b> IO portD reset Set '1' or clear '0' by software 0: No effect; 1: Reset IO port D.
4	IOPCRST	<b>IOPCRST:</b> IO portC reset Set '1' or clear '0' by software 0: No effect; 1: Reset IO port C.
3	IOPBRST	<b>IOPBRST:</b> IO portB reset Set '1' or clear '0' by software 0: No effect; 1: Reset IO port B.
2	IOPARST	<b>IOPARST:</b> IO portA reset Set '1' or clear '0' by software 0: No effect; 1: Reset IO port A.
1	Reserved	Reserved, always reads 0.
0	AFIORST	<b>AFIORST:</b> Auxiliary function I/O reset (Alternate function I/O reset) Set '1' or clear '0' by software 0: No effect; 1: Reset Auxiliary Function.

### 6.3.5 APB1 Peripheral Reset Register (RCC\_APB1RSTR)

Offset address: 0x10

Reset value: 0x0000 0000

Access: no wait cycle, word, half-word and byte accesses

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	DAC RST	PWR RST	BKP RST	Reserved	CAN RST	Reserved	USB RST	I2C2 RST	I2C1 RST	UART5 RST	UART4 RST	USART3 RST	USART2 RST	Reserved	Reserved
	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
SPI3 RST	Reserved	Reserved	WWDG RST	Reserved	Reserved	TIM14 RST	TIM13 RST	TIM12 RST	TIM7 RST	TIM6 RST	TIM5 RST	TIM4 RST	TIM3 RST	TIM2 RST	Reserved
rw			rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	notation	clarification
31:30	Reserved	Reserved, always reads 0.
29	DACRST	<b>DACRST:</b> DAC interface reset Set '1' or clear '0' by software 0: No effect; 1: Reset the DAC interface.
28	PWRRST	<b>PWRRST:</b> Power interface reset Set '1' or clear '0' by software 0: No effect; 1: Reset power connector.
27	BKPRST	<b>BKPRST:</b> Backup interface reset

		Set '1' or clear '0' by software 0: No effect; 1: Reset the backup interface.
26	Reserved	Reserved, always reads 0.
25	CANRST	CANRST: CAN reset Set '1' or clear '0' by software 0: No effect; 1: Reset CAN.
24	Reserved	Reserved, always reads 0.
23	USBRST	USBRST: USB reset Set '1' or clear '0' by software 0: No effect; 1: Reset the USB.
22	I2C2RST	I2C2RST: I2C2 reset Set '1' or clear '0' by software 0: No effect; 1: Reset I2C2.
21	I2C1RST	I2C1RST: I2C1 reset Set '1' or clear '0' by software 0: No effect; 1: Reset I2C1.
20	UART5RST	UART5RST: UART5 reset Set '1' or clear '0' by software 0: No effect; 1: Reset UART5.
19	UART4RST	UART4RST: UART4 reset Set '1' or clear '0' by software 0: No effect; 1: Reset UART4.
18	USART3RST	USART3RST: USART3 reset is set to '1' or cleared to '0' by software. 0: No effect; 1: Reset USART3.
17	USART2RST	USART2RST: USART2 reset Set '1' or clear '0' by software 0: No effect; 1: Reset USART2.
16	Reserved	Reserved, always reads 0.
15	SPI3RST	SPI3RST: SPI3 reset (SPI3 reset) Set '1' or clear '0' by software 0: No effect; 1: Reset SPI3.
14:12	Reserved	Reserved, always reads 0.
11	WWDGRST	WWDGRST: Window watchdog reset Set '1' or clear '0' by software 0: No effect; 1: Reset the window watchdog.
10:9	Reserved	Reserved, always reads 0.
8	TIM14RST	TIM14RST: Timer14 reset Set '1' or clear '0' by software 0: No effect; 1: Reset the TIM14 timer.
7	TIM13RST	TIM13RST: Timer13 reset Set '1' or clear '0' by software 0: No effect; 1: Reset the TIM13 timer.
6	TIM12RST	TIM12RST: Timer12 reset Set '1' or clear '0' by software 0: No effect; 1: Reset the TIM12 timer.
5	TIM7RST	TIM7RST: Timer7 reset Set '1' or clear '0' by software 0: No effect; 1: Reset the TIM7 timer.
4	TIM6RST	TIM6RST: Timer6 reset Set '1' or clear '0' by software 0: No effect; 1: Reset the TIM6 timer.
3	TIM5RST	TIM5RST: Timer5 reset Set '1' or clear '0' by software 0: No effect; 1: Reset the TIM5 timer.
2	TIM4RST	TIM4RST: Timer4 reset

		Set '1' or clear '0' by software 0: No effect; 1: Reset the TIM4 timer.
1	TIM3RST	<b>TIM3RST:</b> Timer3 reset Set '1' or clear '0' by software 0: No effect; 1: Reset the TIM3 timer.
0	TIM2RST	<b>TIM2RST:</b> Timer2 reset Set '1' or clear '0' by software 0: No effect; 1: Reset the TIM2 timer.

### 6.3.6 AHB Peripheral Clock Enable Register (RCC\_AHBENR)

Offset address: 0x14

Reset value: 0x0000 0014

Access: no wait cycle, word, half-word and byte accesses

*Notes: When the peripheral clock is not enabled, the software cannot read the value of the peripheral register and the value returned is always 0x0.*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					SDIO EN	Reserved			CRC EN	Reserv ed	FLITF EN	Reserv ed	SRAM EN	DMA2 EN	DMA1 EN
					rw				rw		rw		rw	rw	rw

Bit	notation	clarification
31:11	Reserved	Reserved, always reads 0.
10	SDIOEN	<b>SDIOEN:</b> SDIO clock enable Set '1' or clear '0' by software. 0: SDIO clock off; 1: SDIO clock on.
9:7	Reserved	Reserved, always reads 0.
6	CRCEN	<b>CRCEN:</b> CRC clock enable Set '1' or clear '0' by software. 0: CRC clock off; 1: CRC clock on.
5	Reserved	Reserved, always reads 0.
4	FLITFEN	<b>FLITFEN:</b> Flash Memory Interface Circuit Clock Enable (FLITF clock enable) Set '1' or clear '0' by software to turn on or off the flash interface circuit clock when in sleep mode. 0: Flash interface circuit clocked off during sleep mode; 1: The flash interface circuit clock is turned on during sleep mode.
3	Reserved	Reserved, always reads 0.
2	SRAMEN	<b>SRAMEN:</b> SRAM interface clock enable Set '1' or clear '0' by software to turn on or off the SRAM clock during sleep mode. 0: SRAM clock off during sleep mode; 1: SRAM clock on during sleep mode.
1	DMA2EN	<b>DMA2EN:</b> DMA2 clock enable Set '1' or clear '0' by software. 0: DMA2 clock off; 1: DMA2 clock on.
0	DMA1EN	<b>DMA1EN:</b> DMA1 clock enable Set '1' or clear '0' by software. 0: DMA1 clock off; 1: DMA1 clock on.



### 6.3.7 APB2 Peripheral Clock Enable Register (RCC\_APB2ENR)

Offset address: 0x18

Reset value: 0x0000 0000

Access: word, half-word and byte access

Normally there are no access wait cycles. However, when a peripheral on the APB2 bus is accessed, a wait state is inserted until the peripheral access to APB2 is complete.

**Notes:** When the peripheral clock is not enabled, the software cannot read the value of the peripheral register and the value returned is always 0x0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved										TIM11 EN	TIM10 EN	TIM9 EN	Reserved		
										rw	rw	rw			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC3 EN	USART1 EN	TIM8 EN	SPI1 EN	TIM1 EN	ADC2 EN	ADC1 EN	IOPG EN	IOPF EN	IOPE EN	IOPD EN	IOPC EN	IOPB EN	IOPA EN	Reserv ed	AFIO EN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

Bit	notation	clarification
31:22	Reserved	Reserved, always reads 0.
21	TIM11EN	<b>TIM11EN:</b> TIM11 Timer clock enable Set '1' or clear '0' by software 0: TIM11 timer clock off; 1: TIM11 timer clock on.
20	TIM10EN	<b>TIM10EN:</b> TIM10 Timer clock enable Set '1' or clear '0' by software 0: TIM10 timer clock off; 1: TIM10 timer clock on.
19	TIM9EN	<b>TIM9EN:</b> TIM9 Timer clock enable Set '1' or clear '0' by software 0: TIM9 timer clock off; 1: TIM9 timer clock on.
18:16	Reserved	Reserved, always reads 0.
15	ADC3EN	<b>ADC3EN:</b> ADC3 interface clock enable Set '1' or clear '0' by software 0: ADC3 interface clock off; 1: ADC3 interface clock on.
14	USART1EN	<b>USART1EN:</b> USART1 clock enable Set '1' or clear '0' by software 0: USART1 clock off; 1: USART1 clock on.
13	TIM8EN	<b>TIM8EN:</b> TIM8 Timer clock enable Set '1' or clear '0' by software 0: TIM8 timer clock off; 1: TIM8 timer clock on.
12	SPI1EN	<b>SPI1EN:</b> SPI1 clock enable Set '1' or clear '0' by software 0: SPI1 clock off; 1: SPI1 clock on.
11	TIM1EN	<b>TIM1EN:</b> TIM1 Timer clock enable Set '1' or clear '0' by software 0: TIM1 timer clock off; 1: TIM1 timer clock on.
10	ADC2EN	<b>ADC2EN:</b> ADC2 interface clock enable Set '1' or clear '0' by software 0: ADC2 interface clock off; 1: ADC2 interface clock on.
9	ADC1EN	<b>ADC1EN:</b> ADC1 interface clock enable Set '1' or clear '0' by software 0: ADC1 interface clock off; 1: ADC1 interface clock on.
8	IOPGEN	<b>IOPGEN:</b> I/O portG clock enable Set '1' or clear '0' by software 0: IO port G clock off; 1: IO port G clock on.
7	IOPFEN	<b>IOPFEN:</b> I/O portF clock enable Set '1' or clear '0' by software 0: IO port F clock off; 1: IO port F clock on.
6	IOPEEN	<b>IOPEEN:</b> I/O portE clock enable



		Set '1' or clear '0' by software 0: IO port E clock off; 1: IO port E clock on.
5	IOPDEN	IOPDEN: I/O portD clock enable Set '1' or clear '0' by software 0: IO port D clock off; 1: IO port D clock on.
4	IOPCEN	IOPCEN: I/O portC clock enable Set '1' or clear '0' by software 0: IO port C clock off; 1: IO port C clock on.
3	IOPBEN	IOPBEN: I/O portB clock enable Set '1' or clear '0' by software 0: IO port B clock off; 1: IO port B clock on.
2	IOPAEN	IOPAEN: I/O portA clock enable Set '1' or clear '0' by software 0: IO port A clock off; 1: IO port A clock on.
1	Reserved	Reserved, always reads 0.
0	AFIOEN	AFIOEN: Auxiliary function IO clock enable (Alternate function I/O clock enable) Set '1' or clear '0' by software 0: Auxiliary function IO clock off; 1: The auxiliary function IO clock is turned on.

### 6.3.8 APB1 Peripheral Clock Enable Register (RCC\_APB1ENR)

Offset address: 0x1C

Reset value: 0x0000 0000

Access: word, half-word and byte access

Normally there are no access wait cycles. However, when a peripheral on the APB1 bus is accessed, a wait state is inserted until the APB1 peripheral access ends.

*Notes: When the peripheral clock is not enabled, the software cannot read the value of the peripheral register and the value returned is always 0x0.*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	DAC EN	PWR EN	BKP EN	Reserved	CAN EN	Reserved	USB EN	I2C2 EN	I2C1 EN	UART5 EN	UART4 EN	USART3 EN	USART2 EN	Reserved	Reserved
	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
SPI3 EN	Reserved	Reserved	WWDG EN	Reserved	TIM14 EN	TIM13 EN	TIM12 EN	TIM7 EN	TIM6 EN	TIM5 EN	TIM4 EN	TIM3 EN	TIM2 EN		
rw			rw		rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bit	notation	clarification
31:30	Reserved	Reserved, always reads 0.
29	DACEN	DACEN: DAC interface clock enable (DAC interface clock enable) by software set '1' or clear '0' 0: DAC interface clock off; 1: DAC interface clock on.
28	PWREN	PWREN: Power interface clock enable is set to '1' or cleared to '0' by software. 0: Power interface clock off; 1: Power interface clock on.
27	BKPEN	BKPEN: Backup interface clock enable is set to '1' or cleared to '0' by software. 0: Backup interface clocked off; 1: The backup interface clock is turned on.
26	Reserved	Reserved, always reads 0.
25	CANEN	CANEN: CAN clock enable is set to '1' or cleared to '0' by software. 0: CAN clock off; 1: CAN clock on.
24	Reserved	Reserved, always reads 0.
23	USBEN	USBEN: USB clock enable is set to '1' or '0' by software. 0: USB clock off; 1: USB clock on.
22	I2C2EN	I2C2EN: I2C2 clock enable is set to '1' or cleared to '0' by software. 0: I2C2 clock off; 1: I2C2 clock on.
21	I2C1EN	I2C1EN: I2C1 clock enable is set to '1' or cleared to '0' by software. 0: I2C1 clock off; 1: I2C1 clock on.

20	UART5EN	<b>UART5EN:</b> UART5 clock enable is set to '1' or cleared to '0' by software. 0: UART5 clock off; 1: UART5 clock on.
19	UART4EN	<b>UART4EN:</b> UART4 clock enable is set to '1' or cleared to '0' by software. 0: UART4 clock off; 1: UART4 clock on.
18	USART3EN	<b>USART3EN:</b> USART3 clock enable is set to '1' or cleared to '0' by software. 0: USART3 clock off; 1: USART3 clock is turned on.
17	USART2EN	<b>USART2EN:</b> USART2 clock enable is set to '1' or cleared to '0' by software. 0: USART2 clock off; 1: USART2 clock on.
16	Reserved	Reserved, always reads 0.
15	SPI3EN	<b>SPI3EN:</b> SPI3 clock enable is set to '1' or cleared to '0' by software. 0: SPI3 clock off; 1: SPI3 clock on.
14	Reversed	Reserved, always reads 0.
13:12	Reserved	Reserved, always reads 0.
11	WWDGEN	<b>WWDGEN:</b> Window watchdog clock enable (Window watchdog clock enable) set to '1' or clear '0' by the software. 0: Window watchdog clock off; 1: Window watchdog clock on.
10:6	Reserved	Reserved, always reads 0.
5	TIM7EN	<b>TIM7EN:</b> Timer7 clock enable is set to '1' or cleared to '0' by software. 0: Timer 7 clock off; 1: Timer 7 clock on.
4	TIM6EN	<b>TIM6EN:</b> Timer6 clock enable is set to '1' or cleared to '0' by software. 0: Timer 6 clock off; 1: Timer 6 clock on.
3	TIM5EN	<b>TIM5EN:</b> Timer5 clock enable is set to '1' or cleared to '0' by software. 0: Timer 5 clock off; 1: Timer 5 clock on.
2	TIM4EN	<b>TIM4EN:</b> Timer4 clock enable is set to '1' or cleared to '0' by software. 0: Timer 4 clock off; 1: Timer 4 clock on.
1	TIM3EN	<b>TIM3EN:</b> Timer3 clock enable is set to '1' or cleared to '0' by software. 0: Timer 3 clock off; 1: Timer 3 clock on.
0	TIM2EN	<b>TIM2EN:</b> Timer2 clock enable is set to '1' or cleared to '0' by software. 0: Timer 2 clock off; 1: Timer 2 clock on.

### 6.3.9 Backup Domain Control Register (RCC\_BDCR)

Offset address: 0x20

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

Accesses: 0 to 3 wait cycles, word, half-word, and byte accesses

When successive accesses are made to this register, a wait state is inserted.

**Notes:** The LSEON, LSEBYP, RTCSEL and RTCEN bits in the Backup Domain Control Register (RCC\_BDCR) are in the backup domain. Therefore, these bits are write-protected after a reset, and changes to these bits can only be made after the DBP position '1' in the Power Control Register (PWR\_CR). Refer to section 5.1 for further information. These bits can only be cleared by a backup domain reset (see section 6.1.3). Any internal or external reset will not affect these bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16				
Reserved															BDRST				
rw																			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
RTCEN	Reserved					RTCSEL[1:0]		Reserved					LSEBYP	LSERDY	LSEON				
rw							rw		rw							rw		r	rw
Bit	notation		clarification																
31:17		Reserved		Reserved, always reads 0.															
16		BDRST		<b>BDRST:</b> Backup domain software reset Set '1' or clear '0' by software 0: Reset is not activated; 1: Reset the entire backup domain.															

15	RTCEN	<b>RTCEN:</b> RTC clock enable Set '1' or clear '0' by software 0: RTC clock off; 1: RTC clock on.
14:10	Reserved	Reserved, always reads 0.
9:8	RTCSEL [1:0]	<b>RTCSEL[1:0]:</b> RTC clock source selection (RTC clock source selection) The RTC clock source is selected by a software setting. Once the RTC clock source is selected, it cannot be changed until the next time the backup domain is reset. It can be cleared by setting the BDRST bit. 00: No clock; 01: LSE oscillator as RTC clock; 10: LSI oscillator as RTC clock; 11: The HSE oscillator is used as the RTC clock after 128 divisions.
7:3	Reserved	Reserved, always reads 0.
2	LSEBYP	<b>LSEBYP:</b> External low-speed clock oscillator bypass (External low-speed oscillator bypass) LSE is bypassed by software in debug mode by setting '1' or clearing '0'. this bit can only be written when the external 32kHz oscillator is off 0: The LSE clock is not bypassed; 1: The LSE clock is bypassed.
1	LSERDY	<b>LSERDY:</b> external low-speed LSE ready (External low-speed oscillator ready) Set '1' or cleared '0' by hardware to indicate if the external 32kHz oscillator is ready. After LSEON is cleared, it takes 6 cycles of the external low-speed oscillator for this bit to be cleared. 0: External 32kHz oscillator not ready; 1: External 32kHz oscillator ready.
0	LSEON	<b>LSEON:</b> External low-speed oscillator enable Set '1' or clear '0' by software 0: External 32kHz oscillator off; 1: External 32kHz oscillator on.

### 6.3.10 Control/Status Register (RCC\_CSR)

Offset address: 0x24

Reset value: 0x0C00 0000, cleared by system reset except for the reset flag, which can only be cleared by a power reset.

Accesses: 0 to 3 wait cycles, word, half-word, and byte accesses

When successive accesses are made to this register, a wait state is inserted.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPWR RSTF	WWDG RSTF	IWDG RSTF	SFT RSTF	POR RSTF	PIN RSTF	Reserv ed	RMVF	Reserved							
rw	rw	rw	rw	rw	rw		rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													LSIRDY	LSION	
													r	rw	

Bit	notation	clarification
31	LPWRRSTF	<b>LPWRRSTF:</b> Low-power reset flag (Low-power reset flag) Set '1' by hardware when a low-power management reset occurs; cleared by software by writing the RMVF bit. 0: no low-power management reset occurs ;) 1: Low power management reset occurs . For more information about Low Power Management Reset, refer to "Low Power Management Reset" at 6.1.1 .
30	WWDGRSTF	<b>WWDGRSTF:</b> Window watchdog reset flag (Window watchdog reset flag) is set to '1' by hardware when window watchdog reset occurs; cleared by software by writing the RMVF bit. 0: No window watchdog reset occurs; 1: A window watchdog reset occurs.
29	IWDGRSTF	<b>IWDGRSTF:</b> Independent watchdog reset flag (Independent watchdog reset flag) Set '1' by hardware when an independent watchdog reset occurs in the VDD region; cleared by software by writing the RMVF bit. 0: No independent watchdog reset occurs; 1: An independent watchdog reset occurs.
28	SFTRSTF	<b>SFTRSTF:</b> Software reset flag (Software reset flag) Set '1' by hardware when a software reset occurs; cleared by software by writing the RMVF bit. 0: No software reset occurred; 1: A software reset has occurred.
27	PORRSTF	<b>PORRSTF:</b> power-on/power-off reset flag (POR/PDR reset flag) Set '1' by hardware when power-up/power-down reset occurs; cleared by software by writing the RMVF bit. 0: No power-up/power-down reset occurs;

		1: A power-up/power-down reset occurs.
26	PINRSTF	<b>PINRSTF:</b> NRST pin reset flag (PIN reset flag) Set '1' by hardware when NRST pin reset occurs; cleared by software by writing the RMVF bit. 0: No NRST pin reset occurs; 1: An NRST pin reset occurs.
25	Reserved	Reserved, read operation returns 0
24	RMVF	<b>RMVF:</b> Remove reset flag is set to '1' by software to clear the reset flag. 0: No effect; 1: Clear the reset flag.
23:2	Reserved	Reserved, read operation returns 0
1	LSIRDY	<b>LSIRDY:</b> Internal low-speed oscillator ready (Internal low-speed oscillator ready) Set '1' or cleared '0' by hardware to indicate the readiness of the internal 40kHz RC oscillator. LSIRDY is cleared after 3 cycles of the internal 40kHz RC oscillator after LSION is cleared. 0: The internal 40kHz RC oscillator clock is not ready; 1: Internal 40kHz RC oscillator clock ready.
0	LSION	<b>LSION:</b> Internal low-speed oscillator enable is set to '1' or cleared to '0' by software. 0: Internal 40kHz RC oscillator off; 1: The internal 40kHz RC oscillator is turned on.

### 6.3.11 Control/Status Register (RCC\_MCO\_VAL)

Offset address: 0x30

Reset value: 0x0000 0001, cleared by system reset except for the reset flag, which can only be cleared by a power reset.

Accesses: 0 to 3 wait cycles, word, half-word, and byte accesses

When successive accesses are made to this register, a wait state is inserted.

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

Bit	notation	clarification
31:4	Reserved	Reserved, read operation returns 0.
3:0	MCOPLLVAL	<b>MCOPLLVAL:</b> MCO(Microcontroller clock output), PLL clock frequency division extension bit. 0001: PLL clock 2-division output.                      1001: PLL clock 10-division output. 0010: PLL clock 3-division output.                    1010: PLL clock 11-division output. 0011: PLL clock 4-division output.                    1011: PLL clock 12-division output. 0100: PLL clock 5-division output.                    1100: PLL clock 13-division output. 0101: PLL clock 6-division output.                    1101: PLL clock 14-division output. 0110: PLL clock 7-division output.                    1110: PLL clock 15-division output. 0111: PLL clock 8-division output.                    1111: PLL clock 16-division output. 1000: PLL clock 9-division output. <b>Note:</b> You must first set the MCO bit of the RCC_CFGR register to 111 (that is, the PLL clock is output after being divided by 2), and then set the current register for the extended bit settings to take effect.

### 6.3.12 SYSCFG\_CONFIG Register (RCC\_SYSCFG\_CONFIG)

Offset address: 0xF0  
Reset value: 0x0000 0000

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

Bit	notation	clarification
31:9	Reserved	Reserved, read operation returns 0.
8	RST	RST: SYSCFG interface reset Set to '1' or cleared to '0' by software 0: No effect 1: Reset SYSCFG
7:1	Reserved	Reserved, read operation returns 0.
0	EN	EN: SYSCFG clock enable Set to '1' or cleared to '0' by software 0: SYSCFG clock is on 1: SYSCFG clock is off

### 6.3.13 RCC Register Address Image

The following table lists the RCC register image and Reset values.

Table12 RCC Register Address Map and Reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
000h	RCC_CR	Reserved								PLLRDY	PLLON	Reserved				CSSON	HSEBYP	HSERDY	HSEON	HSICAL[7:0]							HSITRIM[4:0]				Reserved	HSIRDY	HSION	
	Reset value	0								0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1
004h	RCC_CFGR	USBPRE[2]	ADCPRE[3:2]		PUUMUL[4]		Reserved	MCO [2:0]		Reserved	USBPRE	PLLMUL[3:0]			PLLXTPRE		PLLSRC	ADC PRE [1:0]	PRRE2 [2:0]		PRRE1 [2:0]		HPRE [3:0]			SWS [1:0]		SW [1:0]						
	Reset value	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
008h	RCC_CIR	Reserved								CSSC	Reserved	PLLRDYC	HSERDYC	HSIRDYC	LSERDYC	LSIRDYC	Reserved				PLLRDYIE	HSERDYIE	HSIRDYIE	LSERDYIE	LSIRDYIE	CSSF	Reserved		PLLRDYF	HSERDYF	HSIRDYF	LSERDYF	LSIRDYF	
	Reset value	0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
00Ch	RCC_APB2RSTR	Reserved										TIM11RST		TIM10RST	TIM9RST	Reserved			ADC3RST	USART1RST	TIM8RST	SPI1RST	TIM1RST	ADC2RST	ADC1RST	IOPGRST	IOPFRST	IOPERST	IOPDRST	IOPCRST	IOPBRST	IOPARST	Reserved	AFIORST
	Reset value	0										0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
010h	RCC_APB1RSTR	Reserved	DACRST	PWRRST	BKPRST	Reserved	CANRST	Reserved	USBRST	I2C2RST	I2C1RST	UART5RST	UART4RST	USART3RST	USART2RST	Reserved	SPI3RST	Reserved			WWDGRST	Reserved	TIM14RST	TIM13RST	TIM12RST	TIM7RST	TIM6RST	TIM5RST	TIM4RST	TIM3RST	TIM2RST			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																		
014h	RCC_AHBENR	Reserved																					SDIOEN	Reserved	FSMCEN	Reserved	CRCEN	Reserved	FLITFEN	Reserved	SRAMEN	DMA2EN	DMA1EN																		
	Reset value																						0		0		0		1		0		0		0		0		0		0		0		0		0		0		0
018h	RCC_APB2ENR	Reserved										TIM11EN	TIM10EN	TIM9EN	Reserved			ADC3EN	USART1EN	TIM8RST	SP11EN	TIM1EN	ADC2EN	ADC1EN	IOPGEN	IOPFEN	IOPEN	IPODEN	IOPDEN	IOPCEN	IOPBEN	Reserved	IOPAEN																		
	Reset value											0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0														
01Ch	RCC_APB1ENR	Reserved	DACRST	PWREN	BKPEN	Reserved	CANEN	Reserved	USBEN	I2C2EN	I2C1EN	UART5EN	UART4EN	USART3EN	USART2EN	Reserved	SPI3EN	Reserved			WWDGEN	Reserved					TIM7EN	TIM6EN	TIM5EN	TIM4EN	TIM3EN	TIM2EN																			
	Reset value		0	0	0		0		0	0	0	0	0	0	0		0				0						0	0	0	0	0	0	0	0	0	0	0														
020h	RCC_BDCR	Reserved															BDRST	RTCEN	Reserved			RTC SEL [1:0]		Reserved					LSEBYP	LSERDYF	LSEON																				
	Reset value																0	0				0	0						0	0	0	0	0	0																	
024h	RCC_CSR	LPWRRSTF	WWDGRSTF	IWDGRSTF	SFTRSTF	PORRSTF	PINRSTF	Reserved	RMVF	Reserved																			LSIRDY	LSION																					
	Reset value	0	0	0	0	1	1		0																				0	0																					
028h-02Fh	Reserved																																																		
030h	RCC_MCO_VAL	Reserved																								MCOPLLVAL																									
	Reset value																									0	0	0	1																						
034h-0Ef	Reserved																																																		
0F0h	RCC_SYSCFG_CONFIG	Reserved																							RST	Reserved					EN																				
	Reset value																								0						0																				

Refer to Table1 for the starting addresses of the registers.

## General Purpose and Multiplexed Function I/O (GPIO and AFIO)

### 7.1 GPIO Function Description

Each GPIO port has two 32-bit configuration registers (GPIOx\_CRL, GPIOx\_CRH), two 32-bit data registers (GPIOx\_IDR and GPIOx\_ODR), a 32-bit bit/reset register (GPIOx\_BSRR), a 16-bit reset register (GPIOx\_BRR), and a 32-bit lockout register (GPIOx\_LCKR).

Each bit of the GPIO port can be individually configured by software into a variety of modes, depending on the specific hardware characteristics of each I/O port listed in the datasheet.

- Input Float
- Input pull-up
- Input Dropdown
- analog input
- open-drain output
- push-pull output
- push-pull multiplexing
- open drain multiplexing (ODMP) function

Each I/O port bit is freely programmable; however, the I/O port registers must be accessed as 32-bit words (no half-word or byte accesses are allowed.) The GPIOx\_BSRR and GPIOx\_BRR registers allow independent read/change accesses to any GPIO 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 bit.

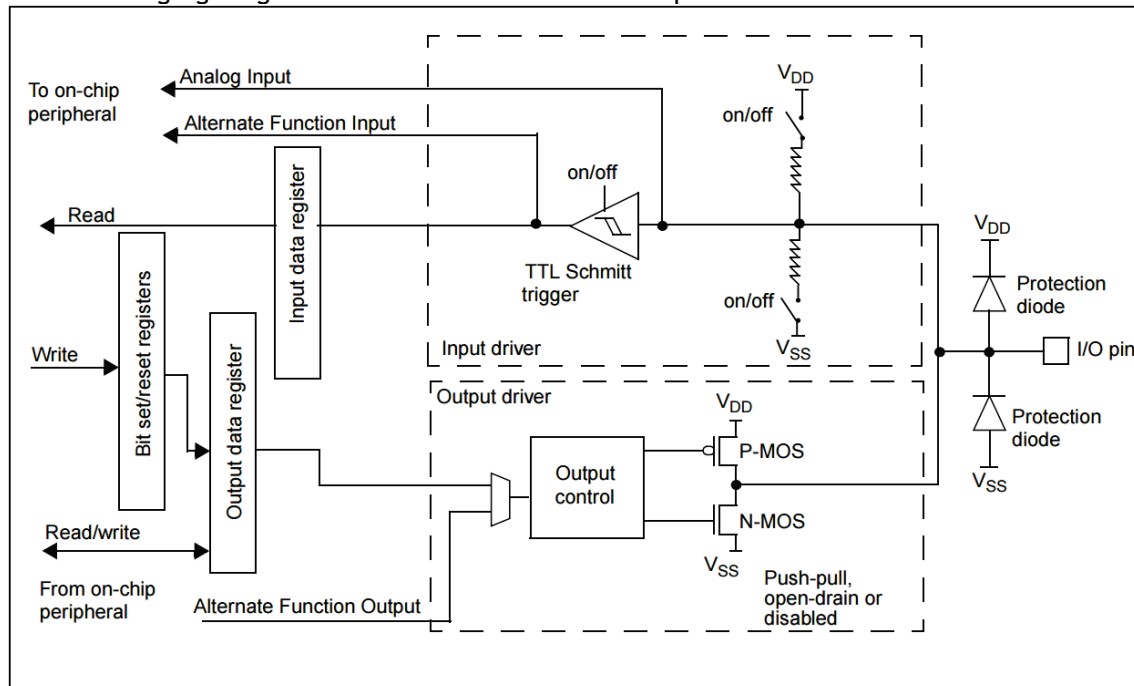


Figure9 Basic Structure of I/O Port Bits

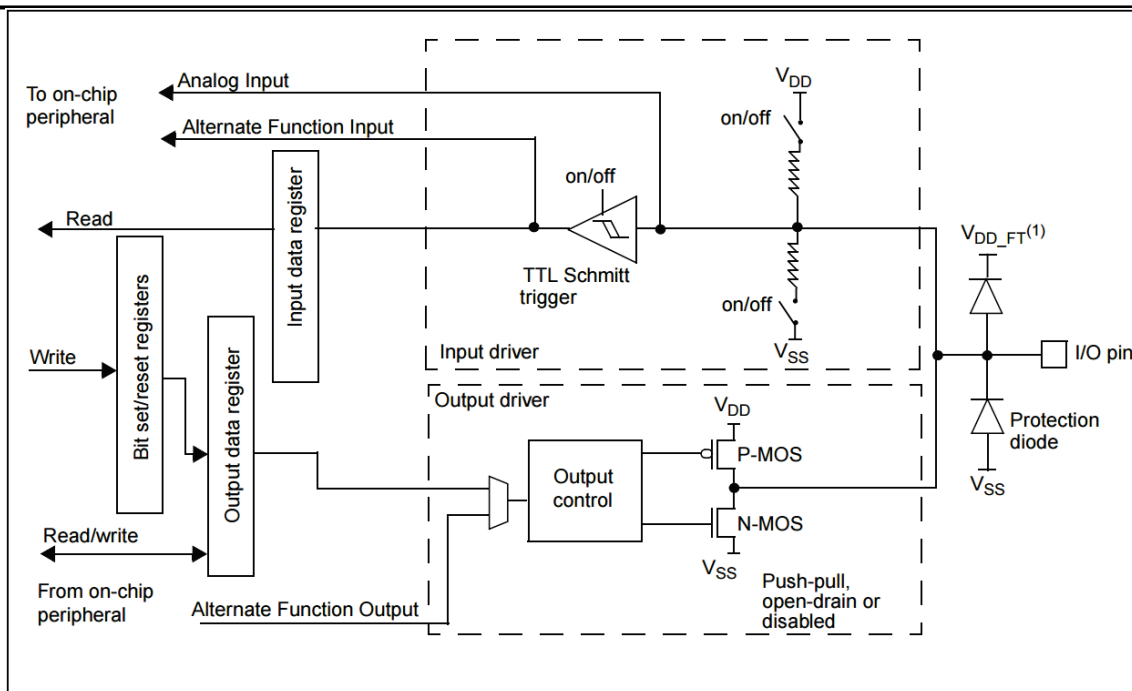


Figure10 Basic Structure of 5-Volt Compatible I/O Port Bits

(1)  $V_{DD\_FT}$  is special to the 5-volt tolerant I/O pin, which is different from VDD

Table13 Port Bit Configuration Table

Configuration Mode		CNF1	CNF0	MODE1	MODE0	PxODR register		
universal output	Push-Pull	0	0	01 10 11 SeeTable14		0 or 1		
	Open-Drain		1			0 or 1		
Multiplexed Function Outputs	Push-Pull	1	0				non-use	
	Open-Drain		1				non-use	
importation	analog input	0	0	00		non-use		
	float input		1			non-use		
	pull-down input	1	0			0		
	pull-up input					1		

Table14 Output Mode Bits

MODE[1:0]	significance
00	Reserved
01	Maximum output speed of 10MHz
10	Maximum output speed of 2MHz
11	Maximum output speed of 50MHz

## 7.1.1 General Purpose I/O (GPIO)

During and just after reset, the multiplexing function is not turned on and the I/O ports are configured in floating input mode (CNF<sub>x</sub>[1:0]=01b, MODE<sub>x</sub>[1:0]=00b).

After reset, the JTAG pins are placed in input pull-up or pull-down mode:

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

When configured as an output, the value written to the output data register (GPIO<sub>x</sub>\_ODR) is output to the appropriate I/O pin. The output drivers can be used in push-pull mode or open-drain mode (when output 0, only the N-MOS is turned on).

The input data register (GPIO<sub>x</sub>\_IDR) captures data on the I/O pins at each APB2 clock cycle.

All GPIO pins have an internal weak pull-up and weak pull-down that can be activated or disconnected when configured as inputs.



---

### 7.1.2 Individual Bit Set or Bit Clear

When programming individual bits of GPIOx\_ODR, software does not need to disable interrupts: only one or more bits can be changed in a single APB2 write operation. This is accomplished by writing a '1' to the bit you want to change in the Set/Reset Register (GPIOx\_BSRR, Reset is GPIOx\_BRR). Bits that are not selected will not be changed.

### 7.1.3 External Interrupt/Wakeup Line

All ports have external interrupt capability. In order to use the external interrupt line, the port must be configured in input mode. For more information on external interrupts, refer to:

- Section 8.2 : External Interrupt/Event Controller (EXTI);
- Section 8.2.3 : Wake-Up Event Management.

### 7.1.4 Multiplexing Function (AF)

The Port Bit Configuration Register must be programmed before using the default multiplexing feature.

- For multiplexed input functions, the port must be configured in input mode (float, pull-up or pull-down) and the input pin must be externally driven.

*Notes: It is also possible to emulate the multiplexed function input pins through software; this emulation can be achieved by programming the GPIO controller. In this case, the port should be set to multiplexed output mode. Obviously, the corresponding pins are no longer driven externally, but by software via the GPIO controller.*

- For the multiplexed output function, the port must be configured for the multiplexed function output mode (push-pull or open-drain).
- For bidirectional multiplexing, the port bits must be configured for the multiplexing function output mode (push-pull or open-drain). At this point, the input driver is configured for floating input mode.

If the port is configured for the multiplexed output function, the pin is disconnected from the output register and connected to the output signal of the on-chip peripheral.

If software configures a GPIO pin to function as a multiplexed output, but the peripheral is not activated, its output will be indeterminate.

### 7.1.5 Software Remapping of I/O Multiplexing Functions

In order to optimize the number of peripheral I/O functions for different device packages, it is possible to remap some of the multiplexed functions to some other pins. This can be done by configuring the appropriate registers in software (refer to the AFIO register description). At this point, the multiplexed functions are no longer mapped to their original pins.

### 7.1.6 GPIO Locking Mechanism

The locking mechanism allows the IO configuration to be frozen. When a lock (LOCK) procedure is performed on a port bit, the configuration of the port bit cannot be changed again until the next reset.

### 7.1.7 Input Configuration

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

- Output buffer disabled
- Schmidt trigger input activated
- Depending on the input configuration (pull-up, pull-down or float), weak pull-up and pull-down resistors are connected
- Data appearing on the I/O pins is sampled at each APB2 clock into the input data registers
- Read accesses to the input data registers yield I/O states

The following figure gives the input configuration of the I/O port bits

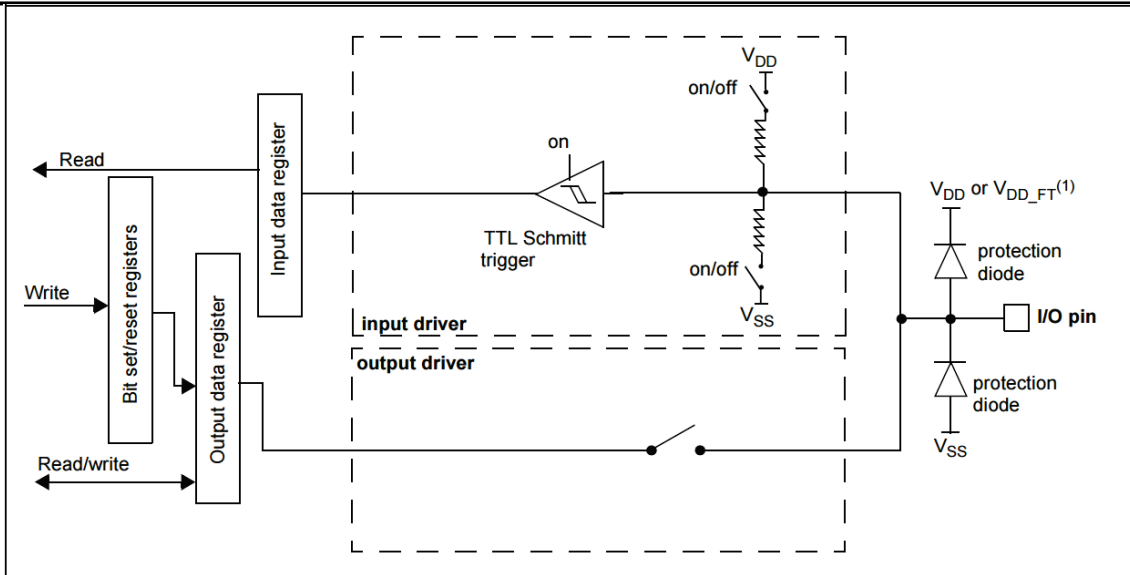


Figure11 Input Float/Pull-Up/Pull-Down Configuration

(1)  $V_{DD\_FT}$  is special for the 5-volt tolerant I/O pin, which is different from  $V_{DD}$

### 7.1.8 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
- Weak pull-up and pull-down resistors are disabled
- Data appearing on the I/O pins is sampled at each APB2 clock into the input data registers
- In open-drain mode, a read access to the input data register yields the I/O state
- In push-pull mode, a read access to the output data register gets the value of the last write.

The following figure gives the output configuration of the I/O port bits.

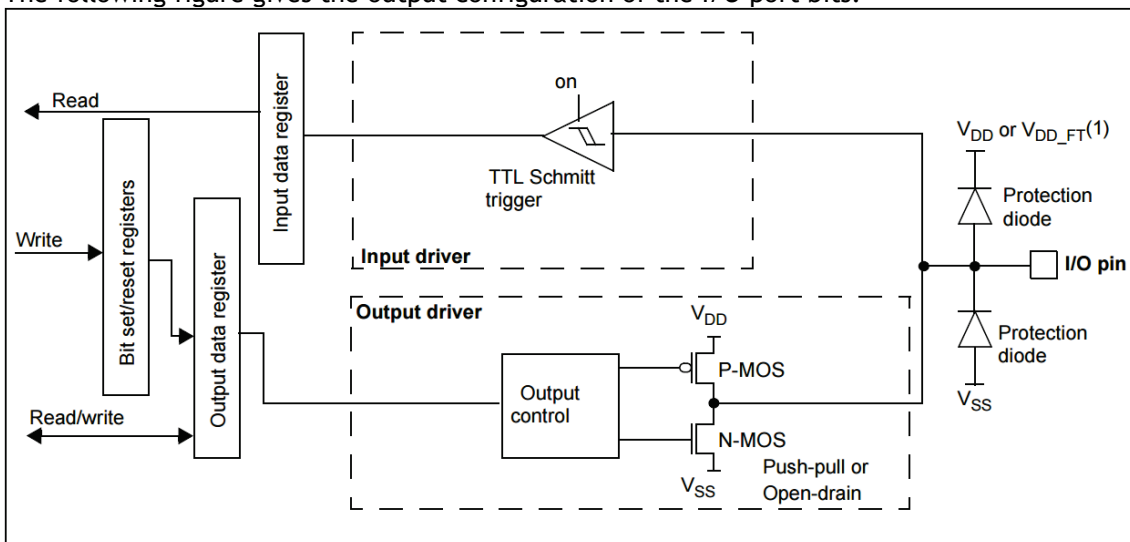


Figure12 Output Configuration

(1)  $V_{DD\_FT}$  is special for 5-volt-compatible I/O pins, which are different from  $V_{DD}$

### 7.1.9 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 driver output buffer with built-in peripheral (multiplexed function output)
- Schmidt trigger input activated

- Weak pull-up and pull-down resistors are disabled
- At each APB2 clock cycle, data appearing on the I/O pins is sampled into the input data registers
- In open-drain mode, the I/O port status can be obtained when reading the input data register.
- In push-pull mode, the value of the last write is available when the output data register is read.

The following figure illustrates the configuration of the multiplexing function for the I/O port bits. See Section 7.3.10 - AFIO Register Description for details.

A set of multiplexed function I/O registers allows the user to remap some multiplexed functions to different pins.

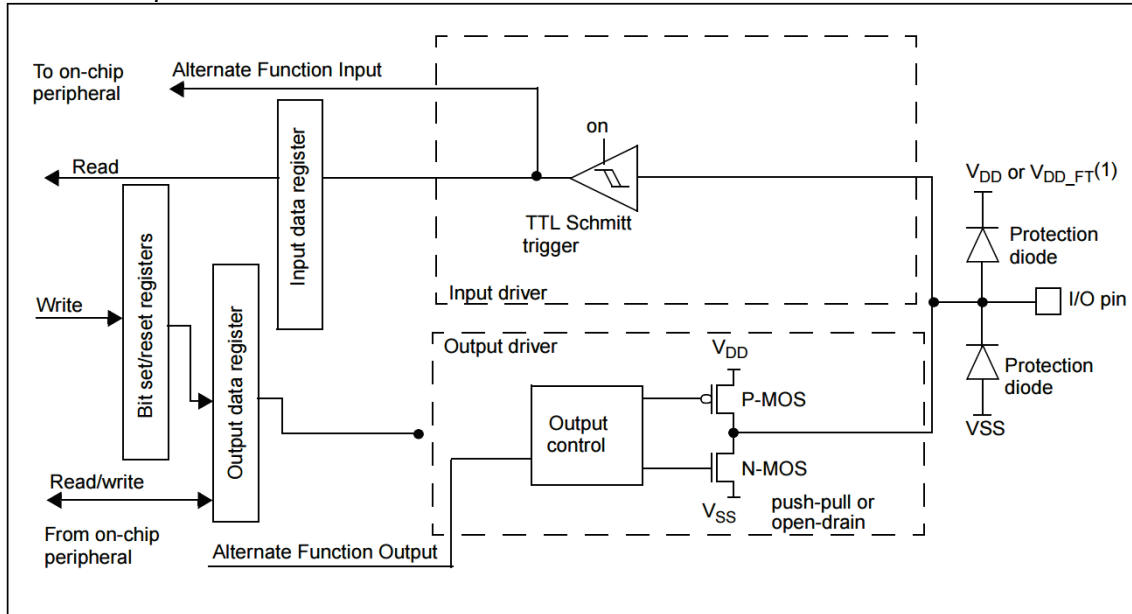


Figure 13 Multiplexing Function Configuration

(1)  $V_{DD\_FT}$  is special for 5-volt-compatible I/O pins, which are different from  $V_{DD}$

### 7.1.10 Analog Input Configuration

When the I/O port is configured for analog input configuration:

- The output buffer is disabled;
- Disabling the 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;
- The value is '0' when the input data register is read.

The following figure illustrates the high impedance analog input configuration for the I/O port bits:

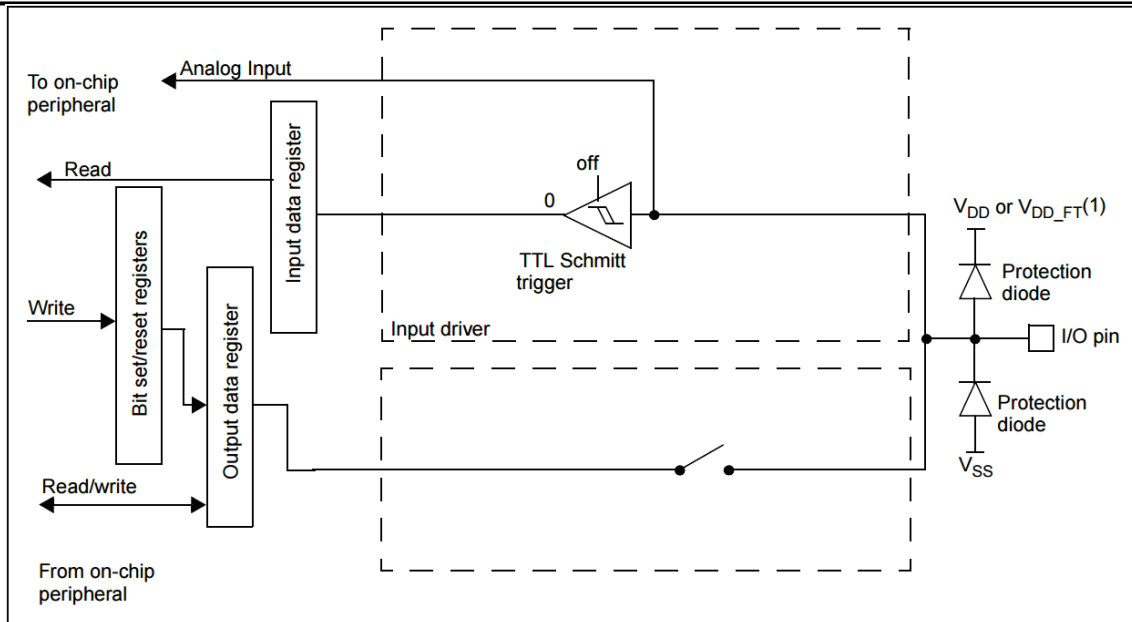


Figure14 High Impedance Analog Input Configuration

(1)  $V_{DD\_FT}$  is special for 5-volt-compatible I/O pins, which are different from  $V_{DD}$ .

### 7.1.11 GPIO Configuration for Peripherals

The following table lists the pinouts for each peripheral.

Table15 Advanced Timer TIM1/TIM8

TIM1/TIM8 pins	configure	GPIO Configuration
TIM1/8_CHx	Input capture channel x	float input
	Output comparison channel x	push-pull multiplexed output
TIM1/8_CHxN	Complementary output channels x	push-pull multiplexed output
TIM1/8_BKIN	Brake Input	float input
TIM1/8_ETR	External Trigger Clock Input	float input

Table16 General Purpose Timer TIM2/3/4/5

TIM2/3/4/5 pins	configure	GPIO Configuration
TIM2/3/4/5_CHx	Input capture channel x	float input
	Output comparison channel x	push-pull multiplexed output
TIM2/3/4/5_ETR	External Trigger Clock Input	float input

Table17 USART s

USART pin	configure	GPIO Configuration
USARTx_TX	full duplex mode	push-pull multiplexed output
	half-duplex synchronous mode	push-pull multiplexed output
USARTx_RX	full duplex mode	Float input or pull-up input
	half-duplex synchronous mode	Unused, can be used as general purpose I/O
USARTx_CK	synchronous mode	push-pull multiplexed output
USARTx_RTS	hardware flow control	push-pull multiplexed output
USARTx_CTS	hardware flow control	Float input or pull-up input

Table18 SPI

SPI Pin	configure	GPIO Configuration
SPIx_SCK	Master Mode	push-pull multiplexed output
	modal	float input
SPIx_MOSI	Full duplex mode/master mode	push-pull multiplexed output
	Full duplex mode/slave mode	Float input or pull-up input
	Simple two-way data line/master mode	push-pull multiplexed output
	Simple bi-directional data line/slave mode	Unused, can be used as

		general purpose I/O
SPIx_MISO	Full duplex mode/master mode	Float input or pull-up input
	Full duplex mode/slave mode	push-pull multiplexed output
	Simple two-way data line/master mode	Unused, can be used as general purpose I/O
	Simple bi-directional data line/slave mode	push-pull multiplexed output
SPIx_NSS	Hardware master/slave mode	Floating input or pull-up input or pull-down input
	Hardware master mode/NSS output enable	push-pull multiplexed output
	software model	Unused, can be used as general purpose I/O

Table19 I2S

I2S Pin	configure	GPIO Configuration
I2Sx_WS	Master Mode	push-pull multiplexed output
	modal	float input
I2Sx_CK	Master Mode	push-pull multiplexed output
	modal	float input
I2Sx_SD	transmitters	push-pull multiplexed output
	refraction	Floating input or pull-up input or pull-down input
I2Sx_MCK	Master Mode	push-pull multiplexed output
	modal	Unused, can be used as general purpose I/O

Table20 I2C Interface

I2C pin	configure	GPIO Configuration
I2Cx_SCL	I2C clock	open-drain multiplexed output
I2Cx_SDA	I2C data	open-drain multiplexed output

Table21 BxCAN

BxCAN pin	GPIO Configuration
CAN_TX	push-pull multiplexed output
CAN_RX	Float input or pull-up input

Table22 USB

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

Table23 SDIO

SDIO Pin	GPIO Configuration
SDIO_CK	push-pull multiplexed output
SDIO_CMD	push-pull multiplexed output
SDIO[D7:D0]	push-pull multiplexed output

ADC input pins must be configured as analog in

Table24 ADC/DAC

ADC/DAC Pins	GPIO Configuration
ADC/DAC	analog input

Table25 Other I/O Functions

pinout	multiplexing function	GPIO Configuration
TAMPER-RTC	RTC output	When configuring the BKP_CR and BKP_RTCCR registers, the setting is forced by hardware
	Intrusion Event Input	
MCO	clock output	push-pull multiplexed output
EXTI Input Line	External Interrupt Input	Floating input or pull-up input or pull-down input

## 7.2 GPIO Register Description

Refer to Chapter 1 for the abbreviations used in the register descriptions.  
These peripheral registers must be operated in word (32-bit) format.

### 7.2.1 Port Configuration Low Register (GPIOx\_CRL) (x=A..G)

Offset address: 0x00

Reset value: 0x4444 4444

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNF7[1:0]	MODE7[1:0]	CNF6[1:0]	MODE6[1:0]	CNF5[1:0]	MODE5[1:0]	CNF4[1:0]	MODE4[1:0]	CNF3[1:0]	MODE3[1:0]	CNF2[1:0]	MODE2[1:0]	CNF1[1:0]	MODE1[1:0]	CNF0[1:0]	MODE0[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
CNF3[1:0]	MODE3[1:0]	CNF2[1:0]	MODE2[1:0]	CNF1[1:0]	MODE1[1:0]	CNF0[1:0]	MODE0[1:0]	CNF0[1:0]	MODE0[1:0]	CNF0[1:0]	MODE0[1:0]	CNF0[1:0]	MODE0[1:0]	CNF0[1:0]	MODE0[1:0]
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	notation	clarification
31:30 27:26 23:22 19:18 15:14 11:10 7:6 3:2	CNFy[1:0]	<b>CNFy[1:0]: port x configuration bits (y=0..7) (Port x configuration bits)</b> The software configures the corresponding I/O port with these bits, refer toTable13 Port Bit Configuration Table. In input mode (MODE[1:0]=00): 00: Analog input mode 01: Float input mode (state after reset) 10: Pull-up/down input mode 11: Reserved In output mode (MODE[1:0]>00): 00: General-purpose push-pull output mode 01: Generalized open-drain output mode 10: Multiplexed function push-pull output mode 11: Multiplexed function open-drain output mode
29:28 25:24 21:20 17:16 13:12 9:8 5:4 1:0	MODEy[1:0]	<b>MODEy[1:0]: mode bits for port x (y=0..7) (Port x mode bits)</b> The software configures the corresponding I/O port with these bits, refer toTable13 Port Bit Configuration Table. 00: Input mode (state after reset) 01: Output mode, max. speed 10MHz 10: Output mode, max. speed 2MHz 11: Output mode, maximum speed 50MHz

### 7.2.2 Port Configuration High Register (GPIOx\_CRH) (x=A..G)

Offset address: 0x04

Reset value: 0x4444 4444

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNF15[1:0]	MODE15[1:0]	CNF14[1:0]	MODE14[1:0]	CNF13[1:0]	MODE13[1:0]	CNF12[1:0]	MODE12[1:0]	CNF11[1:0]	MODE11[1:0]	CNF10[1:0]	MODE10[1:0]	CNF9[1:0]	MODE9[1:0]	CNF8[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
CNF11[1:0]	MODE11[1:0]	CNF10[1:0]	MODE10[1:0]	CNF9[1:0]	MODE9[1:0]	CNF8[1:0]	MODE8[1:0]	CNF8[1:0]	MODE8[1:0]	CNF8[1:0]	MODE8[1:0]	CNF8[1:0]	MODE8[1:0]	CNF8[1:0]	MODE8[1:0]
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	notation	clarification
31:30 27:26 23:22 19:18 15:14 11:10 7:6 3:2	CNFy[1:0]	<b>CNFy[1:0]: port x configuration bits (y=8..15) (Port x configuration bits)</b> The software configures the corresponding I/O port with these bits, refer toTable13 Port Bit Configuration Table. In input mode (MODE[1:0]=00): 00: Analog input mode 01: Float input mode (state after reset) 10: Pull-up/down input mode 11: Reserved In output mode (MODE[1:0]>00): 00: General-purpose push-pull output mode 01: Generalized open-drain output mode 10: Multiplexed function push-pull output mode 11: Multiplexed function open-drain output mode
29:28 25:24 21:20 17:16 13:12 9:8 5:4 1:0	MODEy[1:0]	<b>MODEy[1:0]: mode bits for port x (y=8..15) (Port x mode bits)</b> The software configures the corresponding I/O port with these bits, refer toTable13 Port Bit Configuration Table. 00: Input mode (state after reset) 01: Output mode, max. speed 10MHz 10: Output mode, max. speed 2MHz 11: Output mode, maximum speed 50MHz

### 7.2.3 Port Input Data Register (GPIOx\_IDR) (x=A..G)

Address offset: 0x08

Reset value: 0x0000 XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDR15	IDR14	IDR13	IDR12	IDR11	IDR10	IDR9	IDR8	IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit	notation	clarification
31:16	Reserved	Reserved, always reads 0.
15:0	IDRy[15:0]	IDRy[15:0]: Port input data (y=0...15) (Port input data) These bits are read-only and can only be read out as words (16 bits). The value read out is the status of the corresponding I/O port.

### 7.2.4 Port Output Data Register (GPIOx\_ODR) (x=A..G)

Address Offset: 0x0Ch

Reset value: 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW

Bit	notation	clarification
31:16	Reserved	Reserved, always reads 0.
15:0	IDRy[15:0]	ODRy[15:0]: Port output data (y=0...15) (Port output data) These bits are readable and writable and can only be manipulated as words (16 bits). Note: For GPIOx_BSRR (x=A...E), each ODR bit can be set/cleared independently separately.

### 7.2.5 Port Bit Set/Clear Register (GPIOx\_BSRR) (x=A..G)

Address offset: 0x10

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	notation	clarification
31:16	BRy	BRy: clear port x bit y (y=0..15) (Port x Reset bit y) These bits can only be written to and can only be operated on as words (16 bits). 0: no effect on the corresponding ODRy bit 1: Clear the corresponding ODRy bit to 0 Note: The BSy bit works if the corresponding bits of BSy and BRy are set at the same time.
15:0	BSy	BSy: Set bit y of port x (y=0..15) (Port x Set bit y) These bits can only be written to and can only be operated on as words (16 bits). 0: no effect on the corresponding ODRy bit 1: Set the corresponding ODRy bit to 1

## 7.2.6 Port Bit Clear Register (GPIOx\_BRR) (x=A..G)

Address offset: 0x14

Reset value: 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
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	notation	clarification
31:16	Reserved	Reserved, always reads 0.
15:0	BRy	BRy: clear port x bit y (y=0..15) (Port x Reset bit y) These bits can only be written to and can only be operated on as words (16 bits). 0: no effect on the corresponding ODRy bit 1: Clear the corresponding ODRy bit to 0

## 7.2.7 Port Configuration Lock Register (GPIOx\_LCKR) (x=A..G)

This register is used to lock the configuration of the port bits when bit 16 (LCKK) is set by performing the correct write sequence. Bits [15:0] are used to lock the configuration of the GPIO port. LCKP[15:0] cannot be changed during a defined write operation. After a LOCK sequence has been performed on the corresponding port bit, the configuration of the port bit cannot be changed again until the next system reset.

Each lock bit locks the corresponding 4 bits in the control registers (CRL,CRH).

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															LCKK
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	notation	clarification
31:17	Reserved	Reserved, always reads 0.
16	LCKK	LCKK: Lock key This bit can be read out at any time; it can only be modified by a lock-key write sequence. 0: Port Configuration Lock keystone active 1: The port configuration lock key bit is activated and the GPIOx_LCKR register is locked before the next system reset. Lock key write sequence: Write 1 -> Write 0 -> Write 1 -> Read 0 -> Read 1 The last read can be omitted, but can be used to confirm that the lock key has been activated. Note: The value of LCK[15:0] cannot be changed while operating the lock key write sequence. Any error in the operation lock key write sequence will not activate the lock key.
15:0	LCKy	LCKy: Port x Lock bit y (y=0..15) (Port x Lock bit y) These bits are readable and writable but can only be written when the LCKK bit is zero. 0: Configuration without locking the port 1: Locked port configuration



## 7.2.8 Forced Pull-Up/Pull-Down Configuration Register (GPIOx\_PU\_PD\_EN) (x=A..G)

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PULL_DOWN_EN_15	PULL_DOWN_EN_14	PULL_DOWN_EN_13	PULL_DOWN_EN_12	PULL_DOWN_EN_11	PULL_DOWN_EN_10	PULL_DOWN_EN_9	PULL_DOWN_EN_8	PULL_DOWN_EN_7	PULL_DOWN_EN_6	PULL_DOWN_EN_5	PULL_DOWN_EN_4	PULL_DOWN_EN_3	PULL_DOWN_EN_2	PULL_DOWN_EN_1	PULL_DOWN_EN_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
PULL_UP_EN_15	PULL_UP_EN_14	PULL_UP_EN_13	PULL_UP_EN_12	PULL_UP_EN_11	PULL_UP_EN_10	PULL_UP_EN_9	PULL_UP_EN_8	PULL_UP_EN_7	PULL_UP_EN_6	PULL_UP_EN_5	PULL_UP_EN_4	PULL_UP_EN_3	PULL_UP_EN_2	PULL_UP_EN_1	PULL_UP_EN_0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	notation	clarification
31:16	PULL_DOWN_EN_x	PULL_DOWN_EN_x: Force enable pull-down (x=0...15) 0: Disable pull-down 1: Enable pull-down
15:0	PULL_UP_EN_x	PULL_UP_EN_y: Force enable pull-up (y=0...15) 0: Disable pull-up 1: Enable pull-up

## 7.2.9 Forced Pull-Up/Pull-Down Lock Register (GPIOx\_PU\_PD\_LCKR) (x=A..G)

Address offset: 0x34

Reset value: 0x0000 0000

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

Bit	notation	clarification
31:0	FORCE_PU_PD_LOCK	FORCE_PU_PD_LOCK: Force pull-up and pull-down lock register 0xa5a5a5a5: Enable the forced pull-up and pull-down function other: Disable the forced pull-up and pull-down function

## 7.3 Multiplexed Functional I/O and Debug Configuration (AFIO)

Setting the Multiplexing Remap and Debug I/O Configuration Register (AFIO\_MAPR) implements the remapping of pins. At this point, the multiplexed functions are no longer mapped to their original assignments.

### 7.3.1 OSC32\_IN/OSC32\_OUT as GPIO Port PC14/PC15

When the LSE oscillator is turned off, the LSE oscillator pins OSC32\_IN/OSC32\_OUT can be used as GPIO's PC14/PC15, respectively, and the LSE function always takes precedence over the function of the general-purpose I/O port.

- Notes:
1. The GPIO port function of PC14/PC15 cannot be used when the 1.8V voltage area is turned off (entering standby mode) or when the backup area is powered by VBAT (no more VDD power supply);
  2. See Section 4.1.2 for restrictions on I/O port usage.

### 7.3.2 Use OSC\_IN/OSC\_OUT Pins as GPIO Ports PD0/PD1

The external oscillator pins OSC\_IN/OSC\_OUT can be used as PD0/PD1 of the GPIO, which is realized by setting the Multiplexing Remap and Debug I/O Configuration Register (AFIO\_MAPR). This remapping only applies to 36, 48, and 64-pin packages (separate pins for PD0 and PD1 are available on 100-pin and 144-pin packages and do not need to be remapped)

*Notes: The external interrupt event function is not remapped. On 36, 48, and 64-pin packages, PD0 and PD1 cannot be used to generate external interrupt events.*

### 7.3.3 CAN1 Multiplexing Function Remapping

CAN signals can be mapped to Port A, Port B, or Port D as shown in the table below.

Table26 CAN1 Multiplexing Function Remapping

multiplexing function	CAN_REMAP[1:0]="00"	CAN_REMAP[1:0]="10"	CAN_REMAP[1:0]="11"
CAN1_RX or AN_RX	PA11	PB8	PD0
CAN1_TX or AN_TX	PA12	PB9	PD1

### 7.3.4 JTAG/SWD Multiplexing Function Remapping

The debug interface signals are mapped to GPIO ports as shown in the following table.

Table27 Debug Interface Signals

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

In order to be able to use more GPIOs during debugging, the above remapping configuration can be changed by setting the SWJ\_CFG[2:0] bits of the Multiplexed Remapping and Debugging I/O Configuration Register (AFIO\_MAPR). See the table below.

Table28 Debug Port Images

SWJ_CFG [2:0]	Possible debug ports	SWJ/I/O Pin Assignment				
		PA13/ JTMS/ SWDIO	PA14/ JTCK/ SWCLK	PA15/ JTDI	PB3/ JTDO/ TRACESWO	PB4/ JNTRST
000	Fully SWJ (JTAG-DP+SW-DP) (reset state)	I/O unavailable	I/O unavailable	I/O unavailable	I/O unavailable	I/O unavailable
001	Fully SWJ (JTAG-DP+SW-DP) But no JNTRST.	I/O unavailable	I/O unavailable	I/O unavailable	I/O unavailable	I/O available
010	Turn off the JTAG-DP. Enable SW-DP	I/O unavailable	I/O unavailable	I/O available	I/O available <sup>(1)</sup>	I/O available
100	Turn off the JTAG-DP. Turn off SW-DP	I/O available	I/O available	I/O available	I/O available	I/O available
other than	prohibit the use of sth.					

(1). I/O ports can be used only when asynchronous tracing is not used.

### 7.3.5 ADC multiplexing Function Remapping

Refer to the Multiplexing Remapping and Debugging I/O Configuration Register (AFIO\_MAPR).

Table29 ADC1 External Trigger Injection Conversion Multiplexing Function Remapping

multiplexing function	ADC1_ETRGINJ_REMAP=0	ADC1_ETRGINJ_REMAP=1
ADC1 externally triggered injection conversion	ADC1 externally triggered injection conversion connected to EXTI15	ADC1 externally triggered injection conversion connected to TIM8_CH4

Table30 ADC1 External Trigger Rule Conversion Multiplexing Function Remapping

multiplexing function	ADC1_ETRGREG_REMAP=0	ADC1_ETRGREG_REMAP=1
ADC1 externally triggered injection conversion	ADC1 externally triggered injection conversion connected to EXTI15	ADC1 externally triggered injection conversion connected to TIM8_CH4

Table31 ADC2 External Trigger Injection Conversion Multiplexing Function Remapping

multiplexing function	ADC2_ETRGINJ_REMAP=0	ADC2_ETRGINJ_REMAP=1
ADC2 externally triggered injection conversion	ADC2 externally triggered injection conversion connected to EXTI15	ADC2 externally triggered injection conversion connected to TIM8_CH4

Table32 ADC2 External Trigger Rule Conversion Multiplexing Function Remapping

multiplexing function	ADC2_ETRGREG_REMAP=0	ADC2_ETRGREG_REMAP=1
ADC2 externally triggered rule conversion	ADC2 externally triggered rule conversion connected to EXTI11	ADC2 externally triggered rule conversion connected to TIM8_TRGO

### 7.3.6 Timer Reuse Function Remapping

Channels 1 through 4 of Timer 4 can be remapped from Port B to Port D. Remappings for other timers are listed in Table33-Table37 . See Multiplexing Remapping and Debug I/O Configuration Registers (AFIO\_MAPR).

Table33 TIM5 Multiplexing Function Remap

multiplexing function	TIM5CH4_IEMAP=0	TIM5CH4_IEMAP=1
TIM5_CH4	Channel 4 of TIM5 is connected to PA3.	The LSI internal clock is connected to the input of TIM5_CH4 for calibration.

Table34 TIM4 Multiplexing Function Reimage

multiplexing function	TIM4_REMAP=0	TIM4_REMAP=1
TIM4_CH1	PB6	PD12
TIM4_CH2	PB7	PD13
TIM4_CH3	PB8	PD14
TIM4_CH4	PB9	PD15

Table35 TIM3 Multiplexing Functionality Remap

multiplexing function	TIM3_REMAP[1:0]=00 (No reimaging)	TIM3_REMAP[1:0]=10 (partial re-imaging)	TIM3_REMAP[1:0]=11 (full reimage)
TIM3_CH1	PA6	PB4	PC6
TIM3_CH2	PA7	PB5	PC7
TIM3_CH3	PB0		PC8
TIM3_CH4	PB1		PC9

Table36 TIM2 Multiplexing Functional Remapping

multiplexing function	TIM2_REMAP[1:0]=00 (No reimaging)	TIM2_REMAP[1:0]=01 (partial re-imaging)	TIM2_REMAP[1:0]=10 (partial re-imaging) <sup>(1)</sup>	TIM2_REMAP[1:0]=11 (full reimage)
TIM2_CH1_ETR <sup>(1)</sup>	PA0	PA15	PA0	PA15
TIM2_CH2	PA1	PB3	PA1	PB3
TIM2_CH3	PA2		PB10	
TIM2_CH4	PA3		PB11	

(1). TIM2\_CH1 and TIM2\_ETR share a common pin, but cannot be used at the same time (hence the use of this labeling here: TIM2\_CH1\_ETR)

Table37 TIM1 Multiplexing Function Reimage

multiplexed functional image	TIM1_REMAP[1:0]=00 (No reimaging)	TIM1_REMAP[1:0]=01 (partial re-imaging)	TIM1_REMAP[1:0]=11 (full reimage)
TIM1ETR	PA12		PE7
TIM1_CH1	PA8		PE9
TIM1_CH2	PA9		PE11
TIM1_CH3	PA10		PE13
TIM1_CH4	PA11		PE14
TIM1BKIN	PB12	PA6	PE15
TIM1_CH1N	PB13	PA7	PE8
TIM1_CH2N	PB14	PB0	PE10

TIM1_CH3N	PB15	PB1	PE12
-----------	------	-----	------

Table38 TIM9 Remapping <sup>(1)</sup>

multiplexed functional image	TIM9_REMAP = 0	TIM9_REMAP = 1
TIM9_CH1	PA2	PE5
TIM9_CH2	PA3	PE6

(1). See AF Remapping and Debug I/O Configuration Registers, Section xx AF Remapping and Debug I/O Configuration Register 2 (AFIO\_MAPR2)

Table39 TIM10 Remapping <sup>(1)</sup>

multiplexed functional image	TIM9_REMAP = 0	TIM9_REMAP = 1
TIM10_CH1	PB8	PF6

(1). See AF Remapping and Debug I/O Configuration Registers, Section xx AF Remapping and Debug I/O Configuration Register 2 (AFIO\_MAPR2)

Table40 TIM11 Remapping <sup>(1)</sup>

multiplexed functional image	TIM9_REMAP = 0	TIM9_REMAP = 1
TIM11_CH1	PB9	PF7

(1). See AF Remapping and Debug I/O Configuration Registers, Section xx AF Remapping and Debug I/O Configuration Register 2 (AFIO\_MAPR2)

Table41 TIM13 Remapping <sup>(1)</sup>

multiplexed functional image	TIM9_REMAP = 0	TIM9_REMAP = 1
TIM13_CH1	PA6	PF8

(1). See AF Remapping and Debug I/O Configuration Registers, Section xx AF Remapping and Debug I/O Configuration Register 2 (AFIO\_MAPR2)

Table42 TIM14 Remap <sup>(1)</sup>

multiplexed functional image	TIM9_REMAP = 0	TIM9_REMAP = 1
TIM14_CH1	PA7	PF9

(1). See AF Remapping and Debug I/O Configuration Registers, Section xx AF Remapping and Debug I/O Configuration Register 2 (AFIO\_MAPR2)

## 7.3.7 USART Multiplexing Function Remapping

See Multiplexing Remapping and Debugging I/O Configuration Register (AFIO\_MAPR)

Table43 USART3 Reimage

multiplexing function	usart3_remap[1:0]=00 (No reimaging)	usart3_remap[1:0]=01 (partial re-imaging)	usart3_remap[1:0]=11 (full reimage)
USART3_TX	PB10	PC10	PD8
USART3_RX	PB11	PC11	PD9
USART3_CK	PB12	PC12	PD10
USART3_CTS	PB13		PD11
USART3_RTS	PB14		PD12

Table44 USART2 Reimage

multiplexing function	USART2_REMAP=0	USART2_REMAP=1
USART2_CTS	PA0	PD3
USART2_RTS	PA1	PD4
USART2_TX	PA2	PD5
USART2_RX	PA3	PD6
USART2_CK	PA4	PD7

Table45 USART1 Reimage

multiplexing function	USART1_REMAP=0	USART1_REMAP=1
USART1_TX	PA9	PB6
USART1_RX	PA10	PB7

### 7.3.8 I2C1 Multiplexing Function Remapping

See Multiplexing Remapping and Debugging I/O Configuration Register (AFIO\_MAPR).

Table46 I2C1 Remap

multiplexing function	I2C1_REMAP=0	I2C1_REMAP=1
I2C1_SCL	PB6	PB8
I2C1_SDA	PB7	PB9

### 7.3.9 SPI1 Multiplexing Function Remapping

See Multiplexing Remapping and Debugging I/O Configuration Register (AFIO\_MAPR).

Table47 SPI1 Reimage

multiplexing function	SPI1_REMAP=0	SPI1_REMAP=1
SPI1_NSS	PA4	PA15
SPI1_SCK	PA5	PB3
SPI1_MISO	PA6	PB4
SPI1_MOSI	PA7	PB5

### 7.3.10 SPI3 Multiplexing Function Remapping

Table48 SPI3 Remapping

multiplexing function	SPI3_REMAP=0	SPI3_REMAP=1
SPI1_NSS	PA15	PA4
SPI1_SCK	PB3	PC10
SPI1_MISO	PB4	PC11
SPI1_MOSI	PB5	PC12

## 7.4 AFIO Register Description

Refer to Chapter 1 for the abbreviations used in the register descriptions.

*Notes:* The AFIO clock . Refer to Section0 APB2 Peripheral Clock Enable Register (RCC\_APB2ENR) should be turned on first before performing read or write operations to registers AFIO\_EVCR, AFIO\_MAPR and AFIO\_EXTICRX.

These peripheral registers must be operated in word (32-bit) format.

### 7.4.1 Event Control Register (AFIO\_EVCR)

Address offset: 0x00

Reset value: 0x00000000

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

Bit	notation	instructions
31:8	Reserved	Reserved, always reads 0.
7	EVOE	<b>EVOE:</b> allow event output (Event output enable) This bit can be read and written by software. When this bit is set, the Cortex's EVENTOUT will connect to the I/O port selected by PORT[2:0] and PIN[3:0].
6:4	PORT[2:0]	<b>PORT[2:0]:</b> Port selection Select the port used to output the EVENTOUT signal from the Cortex: 000: Select PA 001: Select PB 010: Select PC 011: Select PD 100: Select PE
3:0	PIN[3:0]	<b>PIN[3:0]:</b> Pin selection (x=A...E) (Pin selection) Select the pin used to output the EVENTOUT signal from the Cortex: 0000: Select Px0    0001: Select Px1    0010: Select Px2    0011: Select Px3 0100: Select Px4    0101: Select Px5    0110: Select of Px6    0111: Select Px7

		1000: Select Px8 1100: Select Px12	1001: Select Px9 1101: Select Px13	1010: Select Px10 1110: Select Px14	1011: Select Px11 1111: Select Px15
--	--	---------------------------------------	---------------------------------------	--	--

## 7.4.2 Multiplexed Remap and Debug I/O Configuration Register (AFIO\_MAPR)

Address offset: 0x04

Reset value: 0x0000 0000

Register images and bit definitions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				SWJ_CFG[2:0]			Reserved				ADC2_ETRGREG_REMAP	ADC2_ETRGINJ_REMAP	ADC1_ETRGREG_REMAP	ADC1_ETRGINJ_REMAP	TIM5CH4_IEMAP
				W											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PD01_REMAP	CAN_REMAP [1:0]	TIM4_REMAP	TIM3_REMAP [1:0]	TIM2_REMAP [1:0]	TIM1_REMAP [1:0]	USART3_REMAP [1:0]		USART2_REMAP		USART1_REMAP	ZC1_REMAP		SPI1_REMAP		
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	notation	clarification
31:27	Reserved	Reserved, always reads 0.
26:24	SWJ_CFG [2:0]	<b>SWJ_CFG[2:0]:</b> serial wire JTAG configuration (Serial wire JTAG configuration) These bits are writable by software only (reading these bits will return undefined values) and are used to configure the I/O ports for the SWJ and trace multiplexing functions. SWJ (Serial Wire JTAG) supports JTAG or SWD access to the Cortex's debug ports. The default state after system reset is SWJ enabled but no trace functionality. This state allows you to select either JTAG or SW (Serial Wire) mode via specific signals on the JTMS/JTCK pin. 000: Fully SWJ (JTAG-DP+SW-DP): reset state; 001: Fully SWJ (JTAG-DP + SW-DP) but no NJTRST; 010: Disable JTAG-DP and enable SW-DP; 100: turn off JTAG-DP, turn off SW-DP; other combinations: no effect.
23:21	Reserved	Reserved, always reads 0.
20	ADC2_ETRGREG_REMAP	<b>ADC2_ETRGREG_REMAP:</b> ADC2 external trigger regular conversion remapping This bit can be set '1' or set '0' by software. It controls the trigger input that is connected to the ADC2 rule-conversion external trigger. When this bit is '0', the ADC2 rule-conversion external trigger is connected to EXTI11; when this bit is '1', the ADC2 rule-conversion external trigger is connected to TIM8_TRGO.
19	ADC2_ETRGINJ_REMAP	<b>ADC2_ETRGINJ_REMAP:</b> ADC2 external trigger injected conversion remapping This bit can be set '1' or set '0' by software. It controls the trigger input that is connected to the ADC2 injection conversion external trigger. When this bit is '0', the ADC2 injection conversion external trigger is connected to EXTI15; when this bit is '1', the ADC2 injection conversion external trigger is connected to TIM8 channel 4.
18	ADC1_ETRGREG_REMAP	<b>ADC1_ETRGREG_REMAP:</b> ADC1 external trigger regular conversion remapping This bit can be set '1' or set '0' by software. It controls the trigger input connected to the ADC2 rule-converting external trigger. When this bit is '0', the ADC1 rule-conversion external trigger is connected to EXTI11; when this bit is '1', the ADC1 rule-conversion external trigger is connected to TIM8_TRGO.
17	ADC1_ETRGINJ_REMAP	<b>ADC1_ETRGINJ_REMAP:</b> ADC1 External trigger injected conversion remapping This bit can be set '1' or set '0' by software. It controls the trigger input that is connected to the ADC2 injection conversion external trigger. When this bit is '0', the ADC2 injection conversion external trigger is connected to EXTI15; when this bit is '1', the ADC1 injection conversion external trigger is connected to TIM8 channel 4.
16	TIM5CH4_IEMAP	<b>TIM5CH4_IEMAP:</b> TIM5 channel4 internal remap (TIM5 channel4 internal remap) This bit can be set '1' or set '0' by software. It controls the TIM5 channel 4 internal image. When this bit is '0', TIM5_CH4 is connected to PA3; when this bit is '1', the LSI internal oscillator is connected to TIM5_CH4 for the purpose of calibrating the LSI.
15	PD01_REMAP	<b>PD01_REMAP:</b> port D0/port D1 mapping to OSC_IN/OSC_OUT (PortD0/PortD1 mapping on OSC_IN/OSC_OUT) This bit can be set '1' or set '0' by software. It controls the GPIO function images for PD0 and PD1. When the main oscillator HSE is not used (the system runs on the internal 8MHz resistive oscillator), PD0 and PD1 can be mapped to the OSC_IN and OSC_OUT pins. 0: No reimagining of PD0 and PD1; 1: PD0 is mapped to OSC_IN and PD1 is mapped to OSC_OUT.
14:13	CAN_REMAP [1:0]	<b>CAN_REMAP[1:0]:</b> CAN alternate function remapping These bits can be set '1' or '0' by software to control the reimagining of the multiplexing functions CAN_RX and CAN_TX. 00: CAN_RX is mapped to PA11 and CAN_TX is mapped to PA12;

		01: Unused combinations; 10: CAN_RX is mapped to PB8 and CAN_TX is mapped to PB9; 11: CAN_RX is mapped to PD0 and CAN_TX is mapped to PD1.
12	TIM4_REMAP	<b>TIM4_REMAP:</b> Timer 4 remapping (TIM4 remapping) This bit can be set '1' or '0' by software to control the mapping of channels 1-4 of the TIM4 to the GPIO port. 0: No reimaging (TIM4_CH1/PB6, TIM4_CH2/PB7, TIM4_CH3/PB8, TIM4_CH4/PB9); 1: Full image (TIM4_CH1/PD12, TIM4_CH2/PD13, TIM4_CH3/PD14, TIM4_CH4/PD15). Note: Reimaging does not affect TIM4_ETR on PE0.
11:10	TIM3_REMAP [1:0]	<b>TIM3_REMAP[1:0]:</b> timer 3 remapping (TIM3 remapping) These bits can be set '1' or set '0' by software to control the image of channels 1 through 4 of Timer 3 on the GPIO port. 00: no reimaging (CH1/PA6, CH2/PA7, CH3/PB0, CH4/PB1); 01: unused combination; 10: Partial image (CH1/PB4, CH2/PB5, CH3/PB0, CH4/PB1); 11: Full image (CH1/PC6, CH2/PC7, CH3/PC8, CH4/PC9). Note: Re-imaging does not affect TIM3_ETR on PD2.
9:8	TIM2_REMAP [1:0]	<b>TIM2_REMAP[1:0]:</b> timer 2 remapping (TIM2 remapping) These bits, which can be set '1' or '0' by software, control the mapping of channels 1 through 4 of Timer 2 and external triggering (ETR) on the GPIO ports. 00: No re-imaging (CH1/ETR/PA0, CH2/PA1, CH3/PA2, CH4/PA3); 01: Partial images (CH1/ETR/PA15, CH2/PB3, CH3/PA2, CH4/PA3); 10: Partial images (CH1/ETR/PA0, CH2/PA1, CH3/PB10, CH4/PB11); 11: Fully imaged (CH1/ETR/PA15, CH2/PB3, CH3/PB10, CH4/PB11).
7:6	TIM1_REMAP [1:0]	<b>TIM1_REMAP[1:0]:</b> timer 1 remapping (TIM1 remapping) These bits can be set '1' or '0' by software to control the image of channels 1 through 4, 1N through 3N, External Trigger (ETR) and Brake Input (BKIN) of Timer 1 on the GPIO port. 00: No reimaging (ETR/PA12, CH1/PA8, CH2/PA9, CH3/PA10, CH4/PA11, BKIN/PB12, CH1N/PB13, CH2N/PB14, CH3N/PB15); 01: Partial images (ETR/PA12, CH1/PA8, CH2/PA9, CH3/PA10, CH4/PA11, BKIN/PA6, CH1N/PA7, CH2N/PB0, CH3N/PB1); 10: Unused combinations; 11: Full Image (ETR/PE7, CH1/PE9, CH2/PE11, CH3/PE13, CH4/PE14, BKIN/PE15, CH1N/PE8, CH2N/PE10, CH3N/PE12).
5:4	USART3_REMAP[1:0]	<b>USART3_REMAP[1:0]:</b> USART3 remapping These bits can be set '1' or '0' by software to control the image of the USART3's CTS, RTS, CK, TX and RX multiplexing functions on the GPIO port. 00: No reimage (TX/PB10, RX/PB11, CK/PB12, CTS/PB13, RTS/PB14); 01: Partial images (TX/PC10, RX/PC11, CK/PC12, CTS/PB13, RTS/PB14); 10: Unused combinations; 11: Full Image (TX/PD8, RX/PD9, CK/PD10, CTS/PD11, RTS/PD12).
3	USART2_REMAP	<b>USART2_REMAP:</b> USART2 remapping These bits can be set '1' or '0' by software to control the image of the USART2's CTS, RTS, CK, TX and RX multiplexing functions on the GPIO port. 0: No reimaging (CTS/PA0, RTS/PA1, TX/PA2, RX/PA3, CK/PA4); 1: Reimage (CTS/PD3, RTS/PD4, TX/PD5, RX/PD6, CK/PD7);
2	USART1_REMAP	<b>USART1_REMAP:</b> USART1 remapping This bit can be set '1' or '0' by software to control the image of USART1's TX and RX multiplexing functions on the GPIO ports. 0: No reimaging (TX/PA9, RX/PA10); 1: Reimage (TX/PB6, RX/PB7).
1	I2C1_REMAP	<b>I2C1_REMAP:</b> I2C1 remapping This bit can be set '1' or '0' by software to control the mapping of I2C1's SCL and SDA multiplexing functions on the GPIO ports. 0: No reimaging (SCL/PB6, SDA/PB7); 1: Reimage (SCL/PB8, SDA/PB9).
0	SPI1_REMAP	<b>SPI1_REMAP:</b> reimage of SPI1 This bit can be set '1' or '0' by software to control the image of SPI1's NSS, SCK, MISO and MOSI multiplexing functions on the GPIO port. 0: No reimaging (NSS/PA4, SCK/PA5, MISO/PA6, MOSI/PA7); 1: Reimage (NSS/PA15, SCK/PB3, MISO/PB4, MOSI/PB5).

### 7.4.3 External Interrupt Configuration Register 1 (AFIO\_EXTICR1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI3[3:0]				EXTI2[3:0]				EXTI1[3:0]				EXTI0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit	notation		clarification												
31:16	Reserved		Reserved, always reads 0.												
15:0	EXTI		EXTIx[3:0]: EXTIx configuration (x=0..3) (EXTI x configuration) These bits can be read and written by software to select the input source for the EXTIx external interrupt. Refer to the 7.2.5 section.												



		0000: PA[x] pin 0001: PB[x] pins 0010: PC[x] pin 0011: PD[x] pin	0100: PE[x] pin 0101: PF[x] pin 0110: PG[x] pin
--	--	---	---

#### 7.4.4 External Interrupt Configuration Register 2 (AFIO\_EXTICR2)

Address offset: 0x0C

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI7 [3:0]				EXTI6 [3:0]				EXTI5 [3:0]				EXTI4 [3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit	notation	clarification													
31:16	Reserved	Reserved, always reads 0.													
15:0	EXTI	<b>EXTIx[3:0]:</b> EXTIx configuration (x=4..7) (EXTI x configuration) These bits can be read and written by software to select the input source for the EXTIx external interrupt. 0000: PA[x] pin 0001: PB[x] pins 0010: PC[x] pin 0011: PD[x] pin 0100: PE[x] pin 0101: PF[x] pin 0110: PG[x] pin													

#### 7.4.5 External Interrupt Configuration Register 3 (AFIO\_EXTICR3)

Address offset: 0x10

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI11 [3:0]				EXTI10 [3:0]				EXTI9 [3:0]				EXTI8 [3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit	notation	clarification													
31:16	Reserved	Reserved, always reads 0.													
15:0	EXTI	<b>EXTIx[3:0]:</b> EXTIx configuration (x=8..11) (EXTI x configuration) These bits can be read and written by software to select the input source for the EXTIx external interrupt. 0000: PA[x] pin 0001: PB[x] pins 0010: PC[x] pin 0011: PD[x] pin 0100: PE[x] pin 0101: PF[x] pin 0110: PG[x] pin													

#### 7.4.6 External Interrupt Configuration Register 4 (AFIO\_EXTICR4)

Address offset: 0x14

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI15 [3:0]				EXTI14 [3:0]				EXTI13 [3:0]				EXTI12 [3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit	notation	clarification													
31:16	Reserved	Reserved, always reads 0.													
15:0	EXTI	<b>EXTIx[3:0]:</b> EXTIx configuration (x=12..15) (EXTI x configuration) These bits can be read and written by software to select the input source for the EXTIx external interrupt. 0000: PA[x] pin 0001: PB[x] pins 0010: PC[x] pin 0011: PD[x] pin 0100: PE[x] pin 0101: PF[x] pin 0110: PG[x] pin													



## 7.4.7 AF Remaps and Debugs I/O Configuration Register 2 (AFIO\_MAPR2)

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						TIM14_REMAP	TIM13_REMAP	TIM11_REMAP	TIM10_REMAP	TIM9_REMAP	Reserved				
						rw	rw	rw	rw	rw					

Bit	notation	clarification
31:10	Reserved	Reserved, always reads 0.
9	TIM14_REMAP	<b>TIM14_REMAP:</b> TIM14 Remapping This bit is set and cleared by software. It controls the mapping of the TIM14CH1 alternative function to the GPIO port. 0: No remapping (PA7) 1: Remapping (PF9)
8	TIM13_REMAP	<b>TIM13_REMAP:</b> TIM13 Remapping This bit is set and cleared by software. It controls the mapping of the TIM13_CH1 alternative function to the GPIO port. 0: No remapping (PA6) 1: Remapping (PF8)
7	TIM11_REMAP	<b>TIM11_REMAP:</b> TIM11 Remapping This bit is set and cleared by software. It controls the mapping of the TIM11_CH1 alternative function to the GPIO port. 0: No remapping (PB9) 1: Remapping (PF7)
6	TIM10_REMAP	<b>TIM10_REMAP:</b> TIM10 Remapping This bit is set and cleared by software. It controls the mapping of the TIM10_CH1 alternative function to the GPIO port. 0: No remapping (PB8) 1: Remapping (PF6)
5	TIM9_REMAP	<b>TIM9_REMAP:</b> TIM9 Remapping This bit is set and cleared by software. It controls the mapping of the TIM9_CH1 alternative function to the GPIO port. 0: No remapping (TIM9_CH1 on PA2, TIM9_CH2 on PA3) 1: Remapping (TIM9_CH1 on PE5, TIM9_CH2 on PE6)
4:0	Reserved	Reserved, always reads 0.



Table50 AFIO Register Address Map and Reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																			
000h	AFIO_EVCR	Reserved																										EVOE		PORT [2:0]		PIN[3:0]																				
	Reset value																											0	0	0	0	0	0	0	0																	
004h	AFIO_MAPR	Reserved				SWJ_CFG [2:0]			Reserved				ADC2_ETRGREG_REMA		ADC2_ETRGINJ_REMA		ADC1_ETRGREG_REMA		ADC1_ETRGINJ_REMA		TIM5CH4_IEMAP		PD01_REMAP		CAN_REMAP[1:0]		TIM4_REMAP		TIM3_REMAP[1:0]		TIM2_REMAP[1:0]		TIM1_REMAP[1:0]		USART3_REMAP[1:0]		USART2_REMAP		USART1_REMAP		I2C1_REMAP		SPI1_REMAP									
	Reset value					0	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0										
008h	AFIO_EXTICR1	Reserved																EXTI3 [3:0]			EXTI2 [3:0]			EXTI1 [3:0]			EXTI0 [3:0]																									
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							
00Ch	AFIO_EXTICR2	Reserved																EXTI7 [3:0]			EXTI6 [3:0]			EXTI5 [3:0]			EXTI4 [3:0]																									
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							
010h	AFIO_EXTICR3	Reserved																EXTI11 [3:0]			EXTI10 [3:0]			EXTI9 [3:0]			EXTI8 [3:0]																									
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							
014h	AFIO_EXTICR4	Reserved																EXTI15 [3:0]			EXTI14 [3:0]			EXTI13 [3:0]			EXTI12 [3:0]																									
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							
01Ch	AFIO_EXTICR2	Reserved																					FSMC_NADV			TIM14_REMAP		TIM13_REMAP		TIM11_REMAP		TIM10_REMAP		TIM9_REMAP		Reserved																
	Reset value																						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

## 8 Interrupts and Events

### 8.1 Nested Vector Interrupt Controller

#### Characterization

- 67 maskable interrupt channels (16 Cortex™ -M3 interrupt lines not included);
- 8 programmable priority levels (3-bit interrupt priority is used);
- Low latency exception and interrupt handling;
- Power management control;
- System control register implementation;

The Nested Vector Interrupt Controller (NVIC) interfaces closely with the processor core to enable low-latency interrupt handling and efficient handling of late arriving interrupts.

The nested vector interrupt controller manages interrupts including kernel exceptions.

#### 8.1.1 SysTick Calibration Value Registers

The system tick calibration value is fixed at 9000, and when the system tick clock is set to 9MHz (the maximum value of HCLK/8), a 1ms time reference is generated.

#### 8.1.2 Interrupt and Exception Vector

Table51 W55MH32 Product Vector Table

placement	prioritization	Priority type	name (of a thing)	clarification	address
	-	-	-	Reserved	0x0000 0000
	-3	set rigidly in place	Reset	reset (a dislocated joint, an electronic device etc)	0x0000 0004
	-2	set rigidly in place	NMI	unmaskable interrupt RCC Clock Security System (CSS) Link to NMI Vector	0x0000 0008
	-1	set rigidly in place	Hard Fault	All types of failures	0x0000 000C
	0	configurable	Storage Management (Mem Manage)	memory management	0x0000 0010
	1	configurable	Bus Fault	Prefetch failure, memory access failure	0x0000 0014
	2	configurable	Usage Fault	Undefined instructions or illegal states	0x0000 0018
	-	-	-	Reserved	0x0000 001c-0x0000 002b
	3	configurable	SVCall	System service calls via SWI instructions	0x0000 002C
	4	configurable	Debug Monitor	Debugging Monitor	0x0000 0030
	-	-	-	Reserved	0x0000 0034
	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	Connected to EXTI's Power Voltage Detection (PVD) interrupt	0x0000 0044
2	9	configurable	TAMPER	Intrusion detection interruption	0x0000 0048
3	10	configurable	RTC	Real Time Clock (RTC) Global Interrupt	0x0000 004C
4	11	configurable	flash	Flash Global Interrupt	0x0000 0050
5	12	configurable	RCC	Reset and Clock Control (RCC) interrupts	0x0000 0054
6	13	configurable	EXTI0	EXTI line 0 interrupt	0x0000 0058
7	14	configurable	EXTI1	EXTI line 1 interrupt	0x0000 005C

placement	prioritization	Priority type	name (of a thing)	clarification	address
8	15	configurable	EXTI2	EXTI line 2 interrupt	0x0000 0060
9	16	configurable	EXTI3	EXTI line 3 interruptions	0x0000 0064
10	17	configurable	EXTI4	EXTI line 4 interrupt	0x0000 0068
11	18	configurable	DMA1 Channel 1	DMA1 channel 1 global interrupt	0x0000 006C
12	19	configurable	DMA1 Channel 2	DMA1 channel 2 global interrupt	0x0000 0070
13	20	configurable	DMA1 channel 3	DMA1 channel 3 global interrupt	0x0000 0074
14	21	configurable	DMA1 channel 4	DMA1 channel 4 global interrupt	0x0000 0078
15	22	configurable	DMA1 channel 5	DMA1 channel 5 global interrupt	0x0000 007C
16	23	configurable	DMA1 channel 6	DMA1 channel 6 global interrupt	0x0000 0080
17	24	configurable	DMA1 channel 7	DMA1 channel 7 global interrupt	0x0000 0084
18	25	configurable	ADC1_2	Global interrupts for ADC1 and ADC2	0x0000 0088
19	26	configurable	USB_HP_CAN_TX	USB high priority or CAN transmit interrupt	0x0000 008C
20	27	configurable	USB_LP_CAN_RX0	USB low priority or CAN receive 0 interrupt	0x0000 0090
21	28	configurable	CAN_RX1	CAN receive 1 interrupt	0x0000 0094
22	29	configurable	CAN_SCE	CANSCE interrupt	0x0000 0098
23	30	configurable	EXTI9_5	EXTI line [9:5] interruptions	0x0000 009C
24	31	configurable	TIM1BRK	TIM1 brake interrupt	0x0000 00A0
25	32	configurable	TIM1_UP	TIM1 update interrupt	0x0000 00A4
26	33	configurable	TIM1_TRG_COM	TIM1 trigger and communication interrupt	0x0000 00A8
27	34	configurable	TIM1_CC	TIM1 Capture Compare Interrupt	0x0000 00AC
28	35	configurable	TIM2	TIM2 global interrupt	0x0000 00B0
29	36	configurable	TIM3	TIM3 Global Interrupt	0x0000 00B4
30	37	configurable	TIM4	TIM4 Global Interrupt	0x0000 00B8
31	38	configurable	I2C1_EV	I2C1 event interrupt	0x0000 00BC
32	39	configurable	I2C1_ER	I2C1 error interrupt	0x0000 00C0
33	40	configurable	I2C2_EV	I2C2 event interrupt	0x0000 00C4
34	41	configurable	I2C2_ER	I2C2 error interrupt	0x0000 00C8
35	42	configurable	SPI1	SPI1 global interrupt	0x0000 00CC
36	43	-	Reserved	Reserved	0x0000 00D0
37	44	configurable	USART1	USART1 global interrupt	0x0000 00D4
38	45	configurable	USART2	USART2 Global Interrupt	0x0000 00D8
39	46	configurable	USART3	USART3 Global Interrupt	0x0000 00DC
40	47	configurable	EXTI15_10	EXTI line [15:10] interruptions	0x0000 00E0
41	48	configurable	RTCAlarm	RTC Alarm Clock Interrupt Connected to EXTI	0x0000 00E4
42	49	configurable	USB Wakeup	Wake from USB standby interrupt connected to EXTI	0x0000 00E8
43	50	configurable	TIM8_BRK	TIM8 brake interrupt	0x0000 00EC
44	51	configurable	TIM8_UP	TIM8 update interrupt	0x0000 00F0
45	52	configurable	TIM8_TRG_COM	TIM8 trigger and communication interrupt	0x0000 00F4
46	53	configurable	TIM8_CC	TIM8 Capture Compare	0x0000 00F8

placement	prioritization	Priority type	name (of a thing)	clarification	address
				Interrupt	
47	54	configurable	ADC3	ADC3 Global Interrupt	0x0000 00FC
48	55	-	Reserved	Reserved	0x0000 0100
49	56	configurable	SDIO	SDIO Global Interrupt	0x0000 0104
50	57	configurable	TIM5	TIM5 Global Interrupt	0x0000 0108
51	58	configurable	SPI3	SPI3 Global Interrupt	0x0000 010C
52	59	configurable	UART4	UART4 Global Interrupt	0x0000 0110
53	60	configurable	UART5	UART5 global interrupt	0x0000 0114
54	61	configurable	TIM6	TIM6 Global Interrupt	0x0000 0118
55	62	configurable	TIM7	TIM7 Global Interrupt	0x0000 011C
56	63	configurable	DMA2 Channel 1	DMA2 channel 1 global interrupt	0x0000 0120
57	64	configurable	DMA2 Channel 2	DMA2 channel 2 global interrupt	0x0000 0124
58	65	configurable	DMA2 Channel 3	DMA2 channel 3 global interrupt	0x0000 0128
59	66	configurable	DMA2 channels 4_5	DMA2 channel 4 and DMA2 channel 5 global interrupts	0x0000 012C
60	-	-	Reserved	Reserved	0x0000 0130 ~ 0x0000 0153
61	69	configurable	RNG	RNG global interrupts	0x0000 0154

## 8.2 External Interrupt/Event Controller (EXTI)

The W55MH32 has 19 edge detectors capable of generating event/interrupt requests. Each input line can be independently configured with the input type (pulse or hang) and the corresponding trigger event (rising or falling edge or both edges triggered). Each input line can be independently masked. The pending register holds the interrupt request for the status line.

### 8.2.1 Main Characteristics

The main features of the EXTI controller are as follows:

- Each interrupt/event is independently triggered and masked
- Each interrupt line has dedicated status bits
- Supports up to 19 software interrupt/event requests
- Detects external signals with pulse widths lower than the APB2 clock width. See the relevant parameters in the Electrical Characteristics section of the datasheet.

## 8.2.2 Flowchart

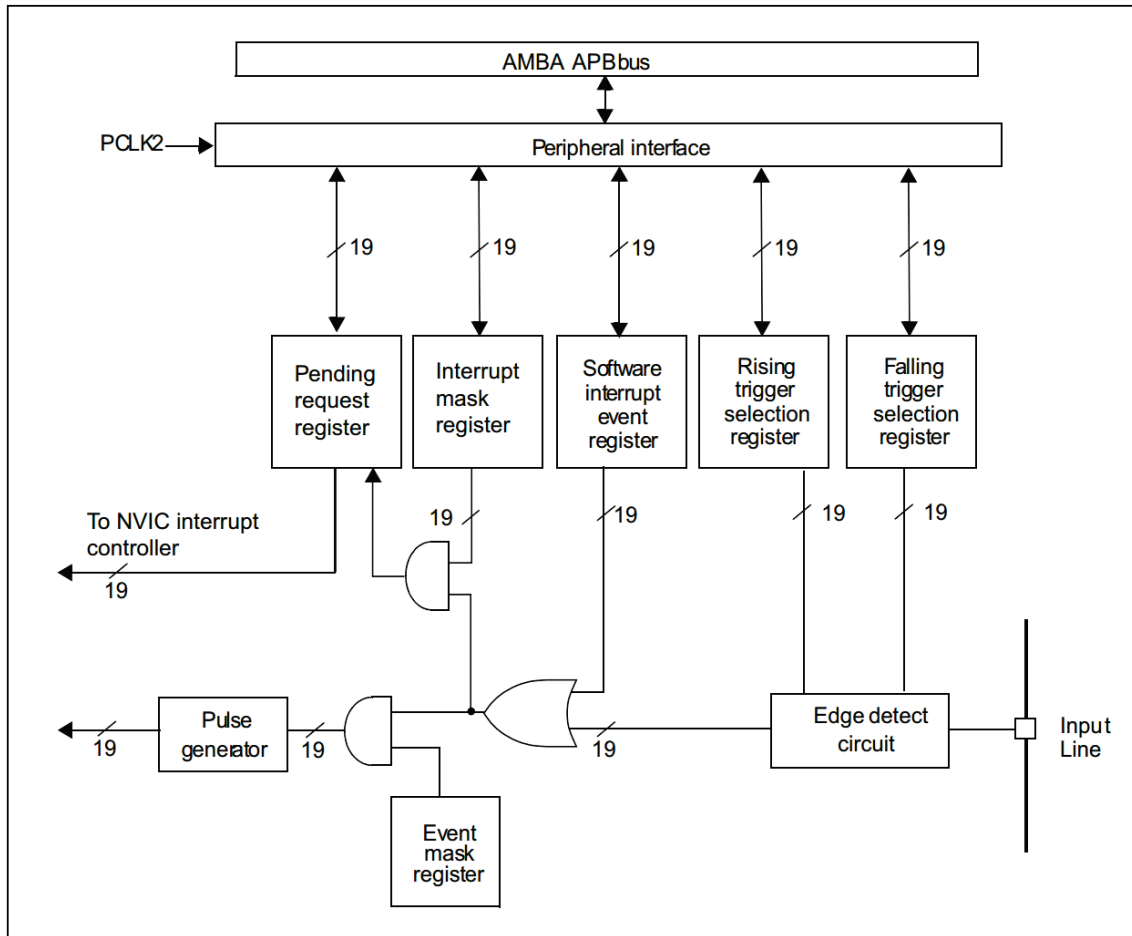


Figure15 External Interrupt/Event Controller Block Diagram

## 8.2.3 Wake-up Event Management

The W55MH32 can handle external or internal events to wake up the core (WFE). Wake-up events can be generated by the following configuration:

- Enable an interrupt in the peripheral's control register, but not in the NVIC, and enable the SEVONPEND bit in the Cortex-M3's system control register. When the CPU recovers from WFE, it needs to clear the interrupt pending bit of the corresponding peripheral and the peripheral NVIC interrupt channel pending bit (in the NVIC interrupt clear pending register).
- Configure an external or internal EXTI line to event mode, and when the CPU recovers from WFE, it does not have to clear the interrupt pending bit of the corresponding peripheral or the NVIC interrupt channel pending bit because the pending bit of the corresponding event line is not set.

To use an external I/O port as a wake-up event, see the function description in the section 7.2.4

## 8.2.4 Functional Description

To generate an interrupt, the interrupt line must first be configured and enabled. Set the 2 trigger registers according to the desired edge detection and also write '1' to the corresponding bit in the interrupt mask register to allow an interrupt request. When an expected edge occurs on the external interrupt line, an interrupt request is generated and the corresponding pending bit is set '1'. Writing '1' to the corresponding bit in the pending register will clear the interrupt request.

If an event needs to be generated, the event line must be configured and enabled first. Edge detection as required is allowed by setting the 2 trigger registers and also writing a '1' to the corresponding bit in the event mask register to allow an event request. When the desired edge occurs on the event line, an event request pulse is generated and the corresponding pending bit is not set '1'.

---

Interrupt/event requests can also be generated by software by writing '1' to the software interrupt/event register.

#### **Hardware Interrupt Selection**

Configure 19 lines as interrupt sources by following the procedure below:

- Configuration of 19 mask bits for interrupt lines (EXTI\_IMR)
- Configure the trigger selection bits (EXTI\_RTISR and EXTI\_FTSR) for the selected breakout line;
- Configuring the enable and mask bits of the NVIC interrupt channel corresponding to the external interrupt controller (EXTI) allows requests in the 19 interrupt lines to be responded to correctly.

#### **Hardware Event Selection**

Through the following process, 19 lines can be configured as event sources

- Configure the mask bits of the 19 event lines (EXTI\_EMR)
- Configure the trigger selection bits (EXTI\_RTISR and EXTI\_FTSR) for the event line

#### **Selection of Software Interrupts/Events**

19 lines can be configured as software interrupt/event lines. The following is the procedure for generating a software interrupt:

- Configure 19 interrupt/event line mask bits (EXTI\_IMR,EXTI\_EMR)
- Setting the request bit of the software interrupt register (EXTI\_SWIER)



## 8.2.5 External Interrupt/Event Line Image

The 61 general-purpose I/O ports are connected to 16 external interrupt/event lines in the manner shown below:

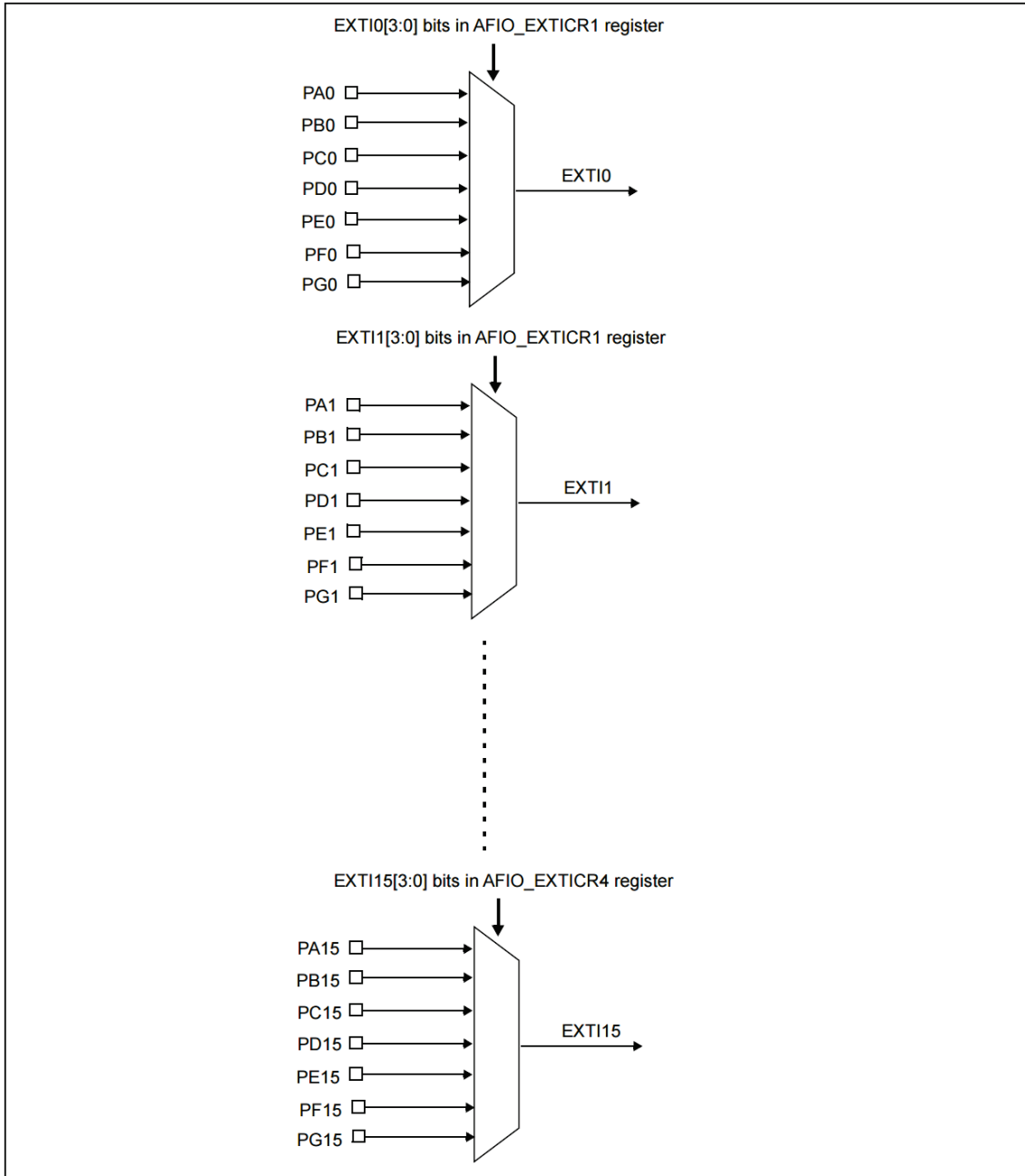


Figure16 External Interrupt General Purpose I/O Image

1. To configure external interrupts/events on the GPIO line via AFIO\_EXTICRx, the AFIO clock must be enabled first.

The other four EXTI lines are connected as follows:

- EXTI line 16 connected to PVD outputs
- EXTI line 17 connected to RTC alarm clock event
- EXTI line 18 connected to USB wake-up event

## 8.3 EXTI Register Description

For abbreviations in register descriptions, refer to Section1 .

These peripheral registers must be operated in word (32-bit) format.

### 8.3.1 Interrupt Mask Register (EXTI\_IMR)

Offset address: 0x00  
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													MR18	MR17	MR16
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MR15	MR14	MR13	MR12	MR11	MR10	MR9	MR8	MR7	MR6	MR4	MR4	MR3	MR2	MR1	MR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit	notation	clarification													
31:19	Reserved	Reserved, always reads 0.													
18:0	MRx	MRx: Interrupt Mask on line x. 0: Block interrupt requests from line x; 1: Open interrupt requests from line x.													

### 8.3.2 Event Mask Register (EXTI\_EMR)

Offset address: 0x04  
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													MR18	MR17	MR16
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MR15	MR14	MR13	MR12	MR11	MR10	MR9	MR8	MR7	MR6	MR4	MR4	MR3	MR2	MR1	MR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit	notation	clarification													
31:19	Reserved	Reserved, always reads 0.													
18:0	MRx	MRx: Event Mask on line x. 0: Block interrupt requests from line x; 1: Open interrupt requests from line x.													

### 8.3.3 Rising Edge Trigger Select Register (EXTI\_RTSTR)

Offset address: 0x08  
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													TR18	TR17	TR16
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR15	TR14	TR13	TR12	TR11	TR10	TR9	TR8	TR7	TR6	TR4	TR4	TR3	TR2	TR1	TR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit	notation	clarification													
31:19	Reserved	Reserved, always reads 0.													
18:0	TRx	TRx: Rising trigger event configuration bit of line x 0: Block interrupt requests from line x; 1: Open interrupt requests from line x.													

Notes: External wake-up lines are edge-triggered, and no burr signals can appear on these lines. When the EXTI\_RTSTR register is written, the rising edge signal on the external interrupt line is not recognized and the hang bit is not set. On the same interrupt line, both rising edge and falling edge triggering can be set. That is, either edge can trigger an interrupt.

### 8.3.4 Falling Edge Trigger Select Register (EXTI\_FTSR)

Offset address: 0x0C  
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													TR18	TR17	TR16
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR15	TR14	TR13	TR12	TR11	TR10	TR9	TR8	TR7	TR6	TR5	TR4	TR3	TR2	TR1	TR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit	notation	clarification													
31:19	Reserved	Reserved, always reads 0.													
18:0	TRx	TRx: Falling trigger event configuration bit of line x 0: Block interrupt requests from line x; 1: Open interrupt requests from line x.													

Notes: External wake-up lines are edge-triggered, and no burr signals can appear on these lines. When the EXTI\_FTSR register is written, the falling edge signal on the external interrupt line is not recognized and the hang bit will not be set. On the same interrupt line, both rising edge and falling edge triggering can be set. That is, either edge can trigger an interrupt.

### 8.3.5 Software Interrupt Event Register (EXTI\_SWIER)

Offset address: 0x10  
Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													WIER18	WIER17	WIER16
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WIER15	WIER14	WIER13	WIER12	WIER11	WIER10	WIER9	WIER8	WIER7	WIER6	WIER5	WIER4	WIER3	WIER2	WIER1	WIER0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit	notation	clarification													
31:19	Reserved	Reserved, always reads 0.													
18:0	SWIERx	SWIERx: Software interrupt on line x. When this bit is '0', writing '1' will set the corresponding pending bit in EXTI_PR. An interrupt will be generated at this point if it is allowed to be generated in EXTI_IMR and EXTI_EMR. Note: The bit can be cleared to '0' by clearing the corresponding bit of EXTI_PR (writing '1').													

### 8.3.6 Pending Register (EXTI\_PR)

Offset address: 0x14  
Reset value: 0xFFFF XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													WIER18	WIER17	WIER16
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WIER15	WIER14	WIER13	WIER12	WIER11	WIER10	WIER9	WIER8	WIER7	WIER6	WIER5	WIER4	WIER3	WIER2	WIER1	WIER0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit	notation	clarification													
31:19	Reserved	Reserved, always reads 0.													
18:0	PRx	PRx: Pending bit 0: No trigger request occurred 1: A trigger request for selection has occurred This bit is set to '1' when a selected edge event occurs on the external interrupt line. Writing a '1' to this bit clears it, or it can be cleared by changing the polarity of the edge detection.													

### 8.3.7 External Interrupt/Event Register Image

The following table lists the EXTI register image and Reset values.

Table52 External Interrupt/Event Controller Register Images and Reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h	EXTI_IMR	Reserved														MR [18:0]																	
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
004h	EXTI_EMR	Reserved														MR [18:0]																	
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
008h	EXTI_RTSTR	Reserved														TR[18:0]																	
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
00Ch	EXTI_FTSR	Reserved														TR[18:0]																	
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
010h	EXTI_SWIER	Reserved														SWIER[18:0]																	
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
014h	EXTI_PR	Reserved														PR[18:0]																	
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SeeTable1 for register start addresses.

## 9 TCP/IP Offload Engine (TOE)

---

### 9.1 TOE Introduction

The TCP/IP Offload Engine (TOE) is an embedded, all-hardware TCP/IP Ethernet controller that provides a simpler, embedded network access solution. 10/100M Ethernet Data Link Layer (MAC) and Physical Layer (PHY), enabling users to extend network connectivity in their applications using a single chip.

The market-proven WIZnet full hardware TCP/IP stack supports TCP, UDP, IPv4, ICMP, ARP, IGMP and PPPoE protocols. A 32K byte on-chip cache is embedded for Ethernet packet processing. If you use the TCP/IP Offload Engine (TOE), you only need some simple socket programming to implement Ethernet applications. This is faster and easier than other embedded Ethernet solutions. Users can use up to 8 hardware sockets to communicate independently at the same time. In order to minimize system power consumption, Wake-on-LAN (WOL) and Power-Off modes are available for customers to choose.

### 9.2 TOE Key Features

- Full hardware TCP/IP protocol support: TCP, UDP, ICMP, IPv4, ARP, IGMP, PPPoE
- Supports 8 independent Sockets
- Supports hibernation mode
- Supports UDP Wake-on-Demand
- Independent 32KB TX/RX cache
- Supports power-down mode
- 10BaseT/100BaseTX Ethernet Physical Layer (PHY)
- Supports auto-negotiation (10/100-Based full/half-duplex)
- LED status display (full/half-duplex, network connection, network speed, activity status)

### 9.3 Register and Memory Composition

The TCP/IP Offload Engine (TOE) has one general-purpose register, eight socket register areas, and a send/receive buffer for each socket. The transmit buffer for each socket is in a 16KB physical transmit memory, initially allocated as 2KB, and the receive buffer for each socket is in a 16KB physical receive memory, initially allocated as 2KB.

Regardless of the size of the receive/transmit cache assigned to each Socket, it must be within the 16-bit offset address range (from 0x0000 to 0xFFFF). For more information on what constitutes 16KB of receive/send memory and how it is accessed, refer to the 9.4 section.

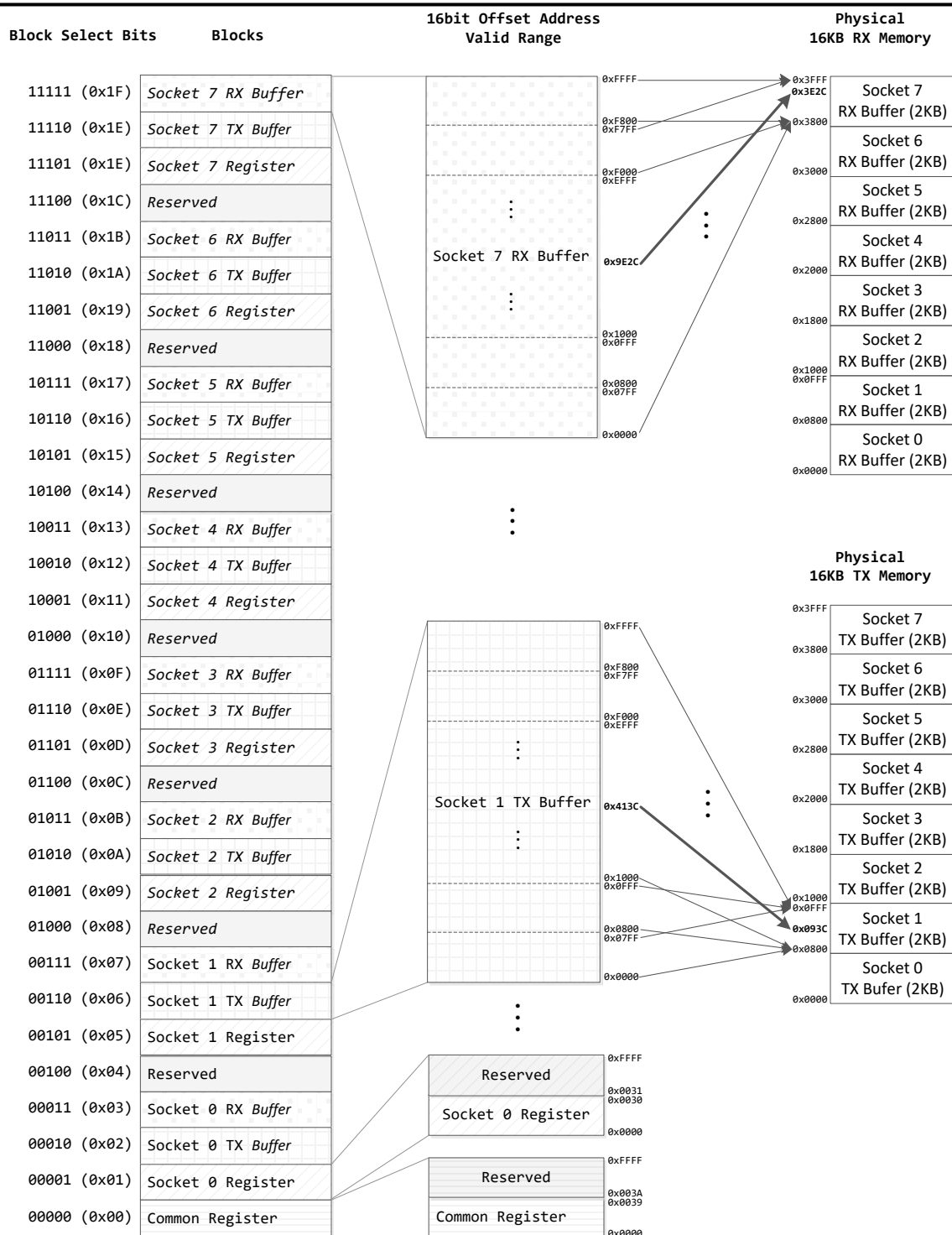


Figure17 Register and Memory Composition

### 9.3.1 General Register Area (GRA)

The General Purpose Register area configures basic information about the TOE, such as IP and MAC addresses. Table53 describes the offset addresses of the general purpose registers. For more information on each register, refer to the 9.5.1 section.

Table53 Offset Addresses for General Purpose Registers

Offset	register	Offset	register	Offset	register
0x0000	paradigm (MR)	0x0013	Interrupt Low Level Timer (INTLEVEL0)	0x0021	(PHAR3)
0x0001	Gateway Address (GAR0)	0x0014	(INTLEVEL1)	0x0022	(PHAR4)
0x0002				0x0023	(PHAR5)
0x0003	Gateway Address (GAR1)	0x0015	interrupt (IR)	0x0024	PPP Session Identification (PSID0)
				0x0025	

0x0004	(GAR2) (GAR3)	0x0016	Interrupt Mask (IMR)		(PSID1)
0x0005 0x0006 0x0007 0x0008	Subnet Mask Address (SUBR0) (SUBR1) (SUBR2) (SUBR3)	0x0017	Socket Interrupt (SIR)	0x0026 0x0027	PPP Maximum Segment Size (PMRU0) (PMRU1)
0x0009 0x000A 0x000B 0x000C 0x000D 0x000E	Source Hardware Address (SHAR0) (SHAR1) (SHAR2) (SHAR3) (SHAR4) (SHAR5)	0x0018	Socket Interrupt Mask (SIMR)	0x0028 0x0029 0x002A 0x002B	Unreachable IP address (UIPR0) (UIPR1) (UIPR2) (UIPR3)
		0x0019 0x001A	Retry Time (RTR0) (RTR1)		
		0x001B	Retry Count (RCR)	0x002C 0x002D	Unreachable Port (UPORTR0) (UPORTR1)
		0x001C	PPP LCP Request Timer (PTIMER)	0x002E	PHY Configuration (PHYCFGR)
		0x001D	PPP LCP Magic number (PMAGIC)	0x002F ~ 0x0038	Reserved
0x000F 0x0010 0x0011 0x0012	Source IP Address (SIPR0) (SIPR1) (SIPR2) (SIPR3)	0x001E 0x001F 0x0020	PPP Destination MAC Address (PHAR0) (PHAR1) (PHAR2)	0x0039	Chip version (VERSIONR)
0x003A ~ 0xFFFF		Reserved			

### 9.3.2 Socket Register Area

The TOE supports 8 sockets as communication channels. Each Socket is controlled through the Socket n register area ( $0 \leq n \leq 7$ ).

Table54 defines the 16-bit offset address corresponding to the Socket n register area. Refer to the 9.5.2 section for details on each register.

Table54 Socket n Offset addresses in registers ( $0 \leq n \leq 7$ )

Offset	register	Offset	register	Offset	register
0x0000	Socket n (Sn_MR)	0x0010 0x0011	Socket n Destination Port (Sn_DPORT0) (Sn_DPORT1)	0x0024 0x0025	Socket n TX Write Pointer (Sn_TX_WR0) (Sn_TX_WR1)
0x0001	Socket n Command (Sn_CR)			0x0026 0x0027	Socket n RX Received Size (Sn_RX_RSR0) (Sn_RX_RSR1)
0x0002	Socket n Interrupt (Sn_IR)	0x0012 0x0013	Socket n Maximum Segment Size (Sn_MSSR0) (Sn_MSSR1)	0x0028 0x0029	Socket n RX Read Pointer (Sn_RX_RD0) (Sn_RX_RD1)
0x0003	Socket n Status (Sn_SR)	0x0014	Reserved	0x002A 0x002B	Socket n RX Write Pointer (Sn_RX_WR0) (Sn_RX_WR1)
0x0004 0x0005	Socket n Source Port (Sn_PORT0) (Sn_PORT1)	0x0015	Socket n IP TOS (Sn_TOS)	0x002C	Socket n Interrupt Mask (Sn_IMR)
0x0006 0x0007 0x0008 0x0009 0x000A 0x000B	Socket n Destination Hardware Address (Sn_DHAR0) (Sn_DHAR1) (Sn_DHAR2) (Sn_DHAR3) (Sn_DHAR4) (Sn_DHAR5)	0x0016	Socket n IP TTL (Sn_TTL)	0x002D 0x002E	Socket n Fragment Offset in IP header (Sn_FRAG0) (Sn_FRAG1)
		0x0017 ~ 0x001D	Reserved	0x002F	Keep alive timer (Sn_KPALVTR)
		0x001E	Socket n Receive Buffer Size (Sn_RXBUF_SIZE)	0x0030 ~ 0xFFFF	Reserved
		0x001F	Socket n Transmit Buffer Size (Sn_TXBUF_SIZE)		
0x000C 0x000D 0x000E 0x000F	Socket n Destination IP Address (Sn_DIPR0) (Sn_DIPR1) (Sn_DIPR2) (Sn_DIPR3)	0x0020 0x0021	Socket n TX Free Size (Sn_TX_FSR0) (Sn_TX_FSR1)		
		0x0022 0x0023	Socket n TX Read Pointer (Sn_TX_RD0) (Sn_TX_RD1)		

## 9.4 Memory

The TOE has a 16KB send memory for Socket n's send buffer and a 16KB receive memory for Socket n's receive buffer.

16KB of transmit memory is initially allocated as a 2KB transmit buffer per socket (2KB X 8 = 16KB). The initially allocated 2KB Socket Transmit Buffer can be reallocated by using the Socket Transmit Buffer Size Register (Sn\_TXBUF\_SIZE).

Once all the Socket Transmit Buffer Size Registers (Sn\_TXBUF\_SIZE) are configured, 16KB of transmit memory is allocated to each Socket's transmit buffer as configured and in order from Socket 0 to 7. The addresses of the 16KB of physical memory are self-addressable. However, to avoid data transfer errors, it is necessary to avoid the sum of the Send Cache Size Register (Sn\_TXBUF\_SIZE) exceeding 16.

The 16KB of read memory is allocated in the same manner as the 16KB of transmit memory. 16KB of receive memory is initially allocated as a 2KB receive buffer per Socket (2KB\*8=16KB). The initially allocated 2KB Socket receive buffer can be reallocated by using the Socket receive buffer size register (Sn\_XBUF\_SIZE).

Once all the Socket transmit buffer size registers (Sn\_TXBUF\_SIZE) have been configured, 16KB of transmit memory is allocated to each Socket's transmit buffer as configured and in order from Socket 0 to 7. The addresses of the 16KB of physical memory are self-incrementing. However, to avoid data transfer errors, it is necessary to avoid the sum of the Send Cache Size Register (Sn\_TXBUF\_SIZE) exceeding 16.

For 16-byte receive/transmit memory allocation, please refer to the description of Sn\_TXBUF\_SIZE and Sn\_RXBUF\_SIZE in Section 9.5.2.

The 16KB of transmit memory is allocated to the corresponding Socket n transmit buffer, which is used to cache data transmitted from the host. the 16-bit offset address of the Socket n transmit buffer supports an addressing range of 64KB (from 0x0000 to 0xFFFF), for its configuration please refer to 'Socket n Transmit Write Pointer Register (Sn\_TX\_WR)' and Socket n Transmit Read Pointer Register (Sn\_RX\_WR). However, this 16-bit offset address is automatically translated into the physical address of the specified 16KB of transmit memory, as shown in Figure 17. Please refer to section 9.5.2 for Sn\_TX\_WR & Sn\_TX\_RD.

The 16KB of receive memory is allocated to the receive buffer corresponding to Socket n, which is used to cache data transmitted from the network. receive buffer of Socket n. The 16-bit offset address of the Socket n receive buffer supports an addressing range of 64KB (from 0x0000 to 0xFFFF), please refer to 'Socket n Receive Read Pointer Register (Sn\_RX\_RD)' and 'Socket n Receive Write Pointer Register (Sn\_RX\_WR)' for more details about the configurations. Socket n accepts read pointer registers (Sn\_RX\_RD) and Socket n accepts write pointer registers (Sn\_RX\_WR). However, this 16-bit offset address is automatically translated into the physical address of the specified 16KB of receive memory as shown in Figure 17. Please refer to the 9.5.2 section for Sn\_RX\_RD & Sn\_RX\_WR.

## 9.5 Register Description

### 9.5.1 General-Purpose Register

**MR (Mode Register - Mode Register) [R/W] [0x0000] [0x00]<sup>1</sup>**

This register is used for S/W reset, ping block mode and PPPoE mode.

7	6	5	4	3	2	1	0
RST	Reserved	WOL	PB	PPPoE	Reserved	FARP	Reserved

Bit	notation	show
7	RST	If this bit (bit) is '1', the internal register will be initialized and it will be cleared automatically after reset.
6	Reserved	reserved bit
5	WOL	Wake on LAN 0: Turn off network wakeup; 1: Turn on Wake on Network; If Wake-on-Network is enabled, normal reception of a Magic Packet from UDP pulls the

<sup>1</sup> Footnote symbols: [Readable/writable] [Memory address] [Default]



		interrupt (INTn) pin low. When using Wake-on-Network, the UDP Socket port needs to be turned on regardless of any source port number. (Refer to the Socket n Mode Register (Sn_MR) Enabling Sockets section for details.) Note: Magic Packet supported by TOE is transported via UDP. The UDP load consists of a 6-byte synchronization stream ('0xFFFF FFFF FFFF') and a 16 destination MAC address stream. Settings regarding passwords are ignored. You can use any UDP source port as a network wakeup.
4	PB	Ping Block Mode 0: Disables the Ping block; 1: Enable the Ping block; If this bit is set to '1', there is no response to a ping request.
3	PPPoE	PPPoE mode 0 : Disable PPPoE mode 1 : Enable PPPoE mode If you want to use ADSL, it needs to be set to '1'.
2	Reserved	Reserved
1	FARP	Forced ARP mode 0 : Disable forced ARP mode; 1 : Enable forced ARP mode; In Forced ARP mode, an ARP request is forced whether or not data is sent.
0	Reserved	Reserved

#### **GAR (Gateway IP Address Register) [R/W] [0x0001 - 0x0004] [0x00]**

This register is used to set the default gateway address.

Example: "192.168.0.1"

0x0001	0x0002	0x0003	0x0004
192 (0xC0)	168 (0xA8)	0 (0x00)	1 (0x01)

#### **SUBR (Subnet Mask Register) [R/W] [0x0005 - 0x0008] [0x00]**

This register is used to set the subnet mask address.

Example: "255.255.255.0"

0x0005	0x0006	0x0007	0x0008
255 (0xFF)	255 (0xFF)	255 (0xFF)	0 (0x00)

#### **SHAR (Source MAC Address Register) [R/W] [0x0009 - 0x000E] [0x00]**

This register is used to set the source MAC address.

Example: "00.08.DC.01.02.03"

0x0009	0x000A	0x000B	0x000C	0x000D	0x000E
0x00	0x08	0xDC	0x01	0x02	0x03

#### **SIPR (Source IP Address Register) [R/W] [0x000F - 0x0012] [0x00]**

This register is used to set the source IP address.

Example: "192.168.0.2"

0x000F	0x0010	0x0011	0x0012
192 (0xC0)	168 (0xA8)	0 (0x00)	2 (0x02)

#### **INTLEVEL (Low Level Interrupt Timer Register) [R/W] [0x0013 - 0x0014] [0x0000]**

This register is used to set the interrupt active wait time (IAWT). When the next interrupt is triggered, the interrupt pin will pull down the interrupt pin (INTn) after the INTLEVEL time.

$$I_{AWT} = (INTLEVEL + 1) \times PLL_{CLK} \times 4 \text{ (when INTLEVEL > 0)}$$

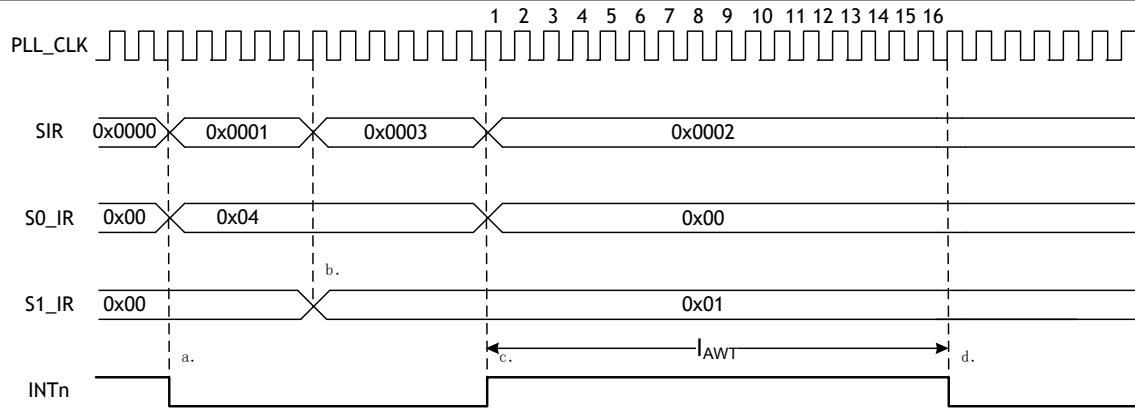


Figure18 INTLEVEL Timing Sequence

1. When Socket 0's timeout interrupt is triggered, the S1\_IR[0] & SIR[1] bits are set to '1', at which time the INTn pin is pulled low.
2. When the connection interrupt for Socket 1 is triggered before the previous interrupt has been processed, the S1\_IR[0] & SIR[1] bits are set to '1', while the INTn pin remains low.
3. If the host completes the previous interrupt entirely by clearing the S0\_IR[3] bit, the INTn pin is pulled high, but S1\_IR[0] & SIR[1] remain '1'.
4. Even though the S1\_IR[0] & SIR[1] bits are set to '1', INTn cannot be pulled down during the INTLEVEL time. Only after the INTLEVEL time, INTn can be pulled down.

#### IR (Interrupt Register) [R/W] [0x0015] [0x00]

The Interrupt Register (IR) specifies the status of the interrupt. A bit of '1' indicates that an interrupt has occurred, and when the host writes a '1' to that bit, that bit of the IR is cleared to '0'. If IR is not equal to '0x00', the INTn pin will be pulled low. INTn will not be pulled high until it becomes '0x00'.

7	6	5	4	3	2	1	0
CONFLICT	UNREACH	PPPoE	MP	Reserved	Reserved	Reserved	Reserved

Bit	notation	clarification
7	CONFLICT	IP conflict This bit will be set to '1' if the sender IP is found to be duplicated with the local IP when the APR request is received
6	UNREACH	The target is unreachable. When an ICMP (destination port unreachable) packet is received, this position '1' When this bit is '1', the target information may be queried through the corresponding UIPR & UPORTR. For example, IP address and port number.
5	PPPoE	PPPoE connection shutdown This bit takes effect when the PPPoE connection is disconnected in PPPoE mode
4	MP	Magic Packet When Wake-on-LAN mode is enabled and Magic Packet Wake-on-LAN packets are received over UDP The bit takes effect
3-0	Reserved	reserved bit

#### IMR (Interrupt Mask Register) [R/W][0x0016][0x00]

The Interrupt Mask Register (IMR) is used to mask the interrupt source. Each interrupt mask bit corresponds to a bit in the Interrupt Register (IR).

An interrupt is generated when a bit in the IMR is '1' and the corresponding bit in the IR is also '1'. In other words, when the mask bit in IMR is cleared '0', no interrupt is generated even if the corresponding IR interrupt bit is '1'.

7	6	5	4	3	2	1	0
IM_IR7	IM_IR6	IM_IR5	IM_IR4	Reserved	Reserved	Reserved	Reserved

Bit	notation	clarification
7	IM_IR7	IP Conflict Interrupt Masking 0: Turn off IP conflict interrupt 1: Enable IP Conflict Interrupt

6	IM_IR6	The destination address cannot reach the interrupt mask 0: Closed destinations cannot be reached 1: Turn on destination not reachable
5	IM_IR5	PPPoE Disable Interrupt Masking 0: Turn off PPPoE Turn off interrupts 1: Turn on PPPoE Turn off interrupts
4	IM_IR4	Magic Packet Interrupt Blocking 0: Turn off Magic Packet interrupt 1: Enable Magic Packet interrupt
3:0	Reserved	reserved bit

#### SIR (Socket Interrupt Register) [R/W] [0x0017] [0x00]

SIR specifies the interrupt status of the Socket, and after SIR is set to 1, it will remain at '1' until Sn\_IR is cleared by the host writing '1'. If Sn\_IR is not equal to '0x00', then it means that the nth bit corresponding to SIR is '1'. INTn can only be pulled down when SIR is '0x00'.

7	6	5	4	3	2	1	0
S7_INT	S6_INT	S5_INT	S4_INT	S3_INT	S2_INT	S1_INT	S0_INT

Bit	notation	clarification
7:0	Sn_INT	When the interrupt of Socket n is triggered, the corresponding bit of the SIR register changes to '1'.

#### SIMR (Socket Interrupt Mask Register) [R/W] [0x0018] [0x00]

Each bit in the SIMR register corresponds to the corresponding bit of the SIR. When one bit of SIMR is '1' and the corresponding bit of SIR is '1', the interrupt will be triggered. In other words, if a bit of SIMR is '0', the interrupt will not be triggered even if the corresponding bit of SIR is '1'.

7	6	5	4	3	2	1	0
S7_IMR	S6_IMR	S5_IMR	S4_IMR	S3_IMR	S2_IMR	S1_IMR	S0_IMR

Bit	notation	clarification
7:0	Sn_IMR	Socket n Interrupt Mask 0: Close Socket n interrupt 1: Enable Socket n interrupts

#### RTR (Retry Time Value Register) [R/W] [0x0019 - 0x001A] [0x07D0]

RTR configures the time value for the retransmission timeout. The value is 100 microseconds per unit. The value is set to 2000 (0x07D0) when initialized, which is equivalent to 200 milliseconds (100us X 2000).

During the RTR configured time period, the TOE waits for a response from the other party after the Sn-CR(CONNECT, DISCON, CLOSE, SEND, SEND\_MAC, SEND\_KEEP command) is transmitted. If there is no response within the RTR time period, the TOE performs a packet retransmission or triggers a timeout interrupt.

For example, when the timeout period is set to 400ms,  $RTR = (400ms / 1ms) \times 10 = 4000$  (0x0FA0)

0x0019	0x001A
0x0F	0xA0

#### RCR (Retry Count Register) [R/W] [0x001B] [0x08]

This register is to set the number of retransmissions. The timeout interrupt is set to '1' when the 'RCR+1th' retransmission occurs. (The 'interrupt' bit ('TIMEOUT' bit) of the interrupt register (Sn\_IR) is set to '1').

Example: RCR = 0x0007

0x001B
0x07

The TOE's timeout can be configured using RTR and RCR. The TOE's timeout includes Address Resolution Protocol (ARP) and TCP retransmission timeout. During the ARP retransmission timeout (see RFC 826 <http://www.ietf.org/rfc.html>), the TOE automatically sends an ARP

request to the peer's IP address to obtain MAC address information (IP, UDP, or TCP is used for communication). As for waiting for the ARP response from the other party (peer), if there no ARP response from the other party (peer) when the retransmission time is set in the RTR, the timeout occurs and the ARP will request a retransmission. Repeat this step up to 'RCR+1' times. If there is still no ARP response when the number of times ARP repeats the request for retransmission reaches 'RCR+1' times, a final timeout interrupt will be triggered and Sn\_IR(TIMEOUT) will change to '1'.

The final timeout value (ARPTO) for ARP requests is as follows.

$$ARP_{TO} = (RTR \times 0.1ms) \times (RCR + 1)$$

When a TCP packet retransmission timeout occurs while RTR and RCR are configured, the TOE sends TCP packets (SYN, FIN, RST, packets) and waits for an acknowledgement (ACK). If there is no ACK response from the peer, a temporary timeout interrupt is triggered and the TCP packets (transmitted earlier) are retransmitted. The TCP packet is retransmitted until it is retransmitted 'RCR+1' times. Even if the TCP packet is retransmitted 'RCR+1' times, if there is still no ACK response from the peer, a final timeout interrupt will be triggered and Sn\_IR(TIMEOUT)='1' will be triggered at the same time.

$$TCP_{TO} = \left( \sum_{N=0}^M (RTR \times 2^N) + ((RCR - M) \times RTR_{MAX}) \right) \times 0.1ms$$

N : Retransmission count,  $0 \leq N \leq M$

M : Minimum value when  $RTR \times 2^{(M+1)} > 65535$  and  $0 \leq M \leq RCR$

RTRMAX :  $RTR \times 2^M$

Example:

When RTR = 2000(0x07D0), RCR = 8(0x0008), the

ARPTO =  $2000 \times 0.1ms \times 9 = 1800ms = 1.8s$

TCPTO =  $(0x07D0 + 0x0FA0 + 0x1F40 + 0x3E80 + 0x7D00 + 0xFA00 + 0xFA00 + 0xFA00 + 0xFA00) \times 0.1ms$   
 $= (2000 + 4000 + 8000 + 16000 + 32000 + ((8 - 4) \times 64000)) \times 0.1ms$   
 $= 318000 \times 0.1ms = 31.8s$

PTIMER (PPP Connection Control Protocol Request Timing Register) [R/W] [0x001C] [0x0028]

PTIMER Sets the time in 25 milliseconds (ms) for sending LCP Echo requests.

Example: if PTIMER is 200,  $200 \times 25(ms) = 5000(ms) = 5 \text{ seconds}$

**PMAGIC (PPP Connection Control Protocol Phantom Register) [R/W] [0x001D] [0x00]**

This register configures the 4-byte Magic number to be used for LCP response requests.

Example: PMAGIC = 0x01, LCP Magic number = 0x01010101

0x001D

0x01

**PHAR (Destination MAC register in PPPoE mode) [R/W] [0x001E-0x0023] [0x0000]**

The PHAR needs to write the MAC address required by the PPPoE server during the PPPoE connection.

Example: Destination MAC address is 00:08:DC:12:34:56

0x001E	0x001F	0x0020	0x0021	0x0022	0x0023
0x00	0x08	0xDC	0x12	0x34	0x56

### PSID (Session ID Register in PPPoE Mode) [R/W] [0x0024-0x0025] [0x0000]

PSID needs to be filled in with the PPPoE server session ID that is required during the PPPoE connection.

Example: Session ID 0x1234

0x0024	0025
18 (0x12)	52(0x34)

### PMRU (Maximum Receiving Unit in PPPoE Mode) [R/W] [0x0026-0x0027] [0xFFFF]

The PMRU specifies the maximum receive unit in PPPoE mode.

For example, the maximum receive unit in PPPoE mode is 0x1234

0x0026	0027
18 (0x12)	52 (0x34)

### UIPR (Unable to Reach IP Address Register) [R] [0x0028-0x002B] [0x00000000]

### UPORTR (Unable to Reach Port Register) [R] [0x002C-0x002D] [0x0000]

When the TOE sends data to an unopened or unreachable port number, an ICMP packet is received (destination unreachable), the IR changes to '1', and the UIPR & UPORTR record the destination IP address and port number respectively.

Example: "192.168.0.11"

0x0028	0x0029	0x002A	0x002B
192 (0xC0)	168 (0xA8)	0 (0x00)	11 (0x0E)

Example: "0x1234"

0x002C	002D
18 (0x12)	52(0x34)

### PHYCFGR (TOE PHY Configuration Register) [R/W] [0x002E] [0b10111XXX]

PHYCFGR configures the operating mode of the PHY and PHY restart. In addition, PHYCFGR specifies the status of the PHY, e.g., Full/Office Duplex, Speed, Link Status.

Bit	notation	clarification																																				
7	RST	Reboot Reset [R/W] When this bit is '0', the internal PHY restarts. This bit shall be set to '1' after PHY reboot.																																				
6	OPMD	Configure the PHY operating mode 1: Configured via the OPMD[2:0] bits of the PHYCFGR; 0: Configured via hardware pins (PMODE[2:0]); This bit configures the operating mode of the PHY via the OPMD[2:0] bits or the PMODE[2:0] pins. When the TOE is restarted via the POR or RSTn pin, the PHY's operating mode defaults to the mode in which it is restarted via the PMODE[2:0] pin configuration. After a POR or RSTn reboot, the user can still reset the Use the OPMD[2:0] bits to configure the PHY operating mode. If the user wants to reset the mode of operation of the PHY via the OPMD[2:0] bits to configure the PHY operating mode. It is necessary for the user to configure the PHY operating mode in the PMODE[2:0] bits. After setting this bit to '1' in pin configuration mode and restarting the PHY with RST position '0'.																																				
5:3	OPMDC	Operating Mode Configuration Bit [R/W] The operating modes of these bit-selected PHYs are shown in the table below: <table><tr><th>5</th><th>4</th><th>3</th><th>clarification</th></tr><tr><td>0</td><td>0</td><td>0</td><td>10BT Half-Duplex, Auto-Negotiation Off</td></tr><tr><td>0</td><td>0</td><td>1</td><td>10BT full duplex, auto-negotiation off</td></tr><tr><td>0</td><td>1</td><td>0</td><td>100BT Half-Duplex, Auto-Negotiation Off</td></tr><tr><td>0</td><td>1</td><td>1</td><td>100BT full duplex, auto-negotiation off</td></tr><tr><td>1</td><td>0</td><td>0</td><td>100BT half-duplex with auto-negotiation enabled</td></tr><tr><td>1</td><td>0</td><td>1</td><td>not yet activated</td></tr><tr><td>1</td><td>1</td><td>0</td><td>power-down mode</td></tr><tr><td>1</td><td>1</td><td>1</td><td>Full-featured, auto-negotiation enabled</td></tr></table>	5	4	3	clarification	0	0	0	10BT Half-Duplex, Auto-Negotiation Off	0	0	1	10BT full duplex, auto-negotiation off	0	1	0	100BT Half-Duplex, Auto-Negotiation Off	0	1	1	100BT full duplex, auto-negotiation off	1	0	0	100BT half-duplex with auto-negotiation enabled	1	0	1	not yet activated	1	1	0	power-down mode	1	1	1	Full-featured, auto-negotiation enabled
5	4	3	clarification																																			
0	0	0	10BT Half-Duplex, Auto-Negotiation Off																																			
0	0	1	10BT full duplex, auto-negotiation off																																			
0	1	0	100BT Half-Duplex, Auto-Negotiation Off																																			
0	1	1	100BT full duplex, auto-negotiation off																																			
1	0	0	100BT half-duplex with auto-negotiation enabled																																			
1	0	1	not yet activated																																			
1	1	0	power-down mode																																			
1	1	1	Full-featured, auto-negotiation enabled																																			

2	DPX	Duplex operating status [read-only] 1: Full duplex 0: Half-duplex
1	SPD	Speed status [read-only] 1: 100Mbps based 0: 10Mbps based
0	LNK	Connection status [read-only] 1: Connected 0: Connection disconnected

#### VERSIONR (TOE Chip Version Register) [R] [0x0039] [0x04]

The TOE's version register defaults to 0x04.

## 9.5.2 Socket Register

#### Sn<sup>2</sup>\_MR (Socket n Mode Register) [R/W] [0x0000] [0x00]

This register is used to configure all SOCKET options or protocol types.

7	6	5	4	3	2	1	0
MULTI/ MFEN	BCASTB	ND / MC /MMB	UCASTB MIP6B	P3	P2	P1	P0

Bit	notation	clarification
7	MULTI/ MFEN	<p>UDP multicast mode 0: Turn off multicasting 1: Enable Multicast This bit is only valid if UDP mode (P[3:0] = '0010'). To use multicast mode, Sn_DIPR &amp; Sn_DPORT are required on Socket n through the opening of Sn_CR. Configure the multicast IP address and port number separately before the configuration command opens.</p> <p>Enable MAC address filtering in MACRAW mode 0 : Disable MAC Filtering 1 : Enable MAC Filtering 0: Disable MAC address filtering 1: Enable MAC address filtering This bit is only valid during the MACRAW(P[3:0] = '0100') mode. If set to '1', TOE accepts only broadcast packets as well as packets sent to himself. When this bit is set to '0', the TOE can receive all packets from the network. If the user wants to implement mixed TCP and IP protocols stack, it is recommended to set this bit to '1' to reduce the load on the host to process all incoming packets.</p>
6	BCASTB	<p>Network Blocking in MACRAW and UDP Mode 0: Turn off broadcast blocking 1: Enable broadcast blocking This bit in UDP mode (P[3:0] = '0010') blocks the reception of broadcast packets. In addition, this bit is also valid in MACRAW mode (P[3:0] = '0100').</p>
5	ND/MC/MMB	<p>Use non-delayed ACK 0: Turn off the no delay ACK option 1: Enable the no delay ACK option This bit is only valid in TCP mode (P[3:0] = '0001'). When this bit is set to '1', the TOE will reply to the ACK packet as soon as possible after receiving the packet from the opposite end without any delay. When this bit is '0', the TOE sends ACK packets that require the timeout set by the RTR for delay.</p> <p>multicast 0: Use IGMP version 2 1: Using IGMP Version 1 This bit only takes effect in UDP mode (P[3:0] = '0010') with MULTI='1'. He configured the version of IGMP messages (join/leave/report).</p> <p>Multicast Blocking in MACRAW Mode 0: Turn off multicast blocking 1: Enable multicast blocking This bit is only effective in MACRAW(P[3:0] = '0100') mode. He can block multicast</p>

<sup>2</sup>is the Socket number (0, 1, 2, 3, 4, 5, 6, 7). n The SNUM[2:0] control bit set is set.

		MAC address for packet transmission.
4	UCASTB MIP6B	<p>Unicast Blocking in UDP Mode 0: Disable Unicast Blocking 1: Enable unicast In UDP mode (P[3:0] = '0010') and with MULTI='1', this bit masks the reception of unicast packets;</p> <p>IPv6 packet blocking in MACRAW mode 0: Turn off IPv6 packet blocking 1: Enable IPv6 packet blocking This bit is only effective in MACRAW mode (P[3:0]='0100'). It will block the reception of IPv6 Package.</p>
3	P3	Protocol
2	P2	This bit configures the protocol mode used by Socket n
1	P1	
0	P0	

P3	P2	P1	P0	hidden meaning
0	0	0	0	Closed
0	0	0	1	TCP
0	0	1	0	UDP
0	1	0	0	MACRAW

\* MACRAW is only available under Socket 0.

### Sn\_CR (Socket n Configuration Register) [R/W] [0x0001] [0x00]

(be) worth	notation	clarification	
0x01	OPEN	Initialize and open (open) Socket n according to the protocol selection of Sn_MR(P3:P0). The following table shows the corresponding values of Sn_SR and Sn_MR.	
		Sn_MR (P[3:0])	Sn_SR
		Sn_MR_CLOSE ('0000')	-
		Sn_MR_TCP ('0001')	SOCK_INIT (0x13)
		Sn_MR_UDP ('0010')	SOCK_UDP (0x22)
		s0_mr_macraw ('0100')	SOCK_MACRAW (0x02)
0x02	LISTEN	This bit takes effect only in TCP mode (Sn_MR(P3:P0) = Sn_MR_TCP). In this mode, Socket n is configured as a TCP server, which is waiting for the "TCP Client" connection request (SYN packet). This Sn_SR register is defined by SOCK_INIT Change to SOCK_LISTEN. When a TCP client's connection request is successful, this Sn_SR register is changed by the SOCK_LISTEN changes to SOCK_ESTABLISHED with Sn_IR(0) will change to '1'. On the other hand, when the connection fails, Sn_IR(3) is set to '1', the Sn_SR changed to SOCK_CLOSED.	
0x04	CONNECT	This mode is only available in TCP mode and running Socket n as a TCP client. This is accomplished by connecting to the stored in Destination Address Register (Sn_DIPR) and Port Number Register (Sn_DPORT) IP address and port number to connect, a connection request is sent to the TCP server. When a client's connection request is successful, the Sn_SR register changes to the SOCK_ESTABLISHED, Sn_IR(0) will change to '1'. The following three conditions mean that the connection request failed: ARPT0 timeout occurred (Sn_IR(s)='1'). Because the MAC address of the destination Cannot be obtained through the ARP process. When no SYN/ACK packet is received and TCPT0(Sn_IR(3)) is set to '1', the TCPT0(Sn_IR(3)) is set to '1'. Hours. When an RST packet is received instead of a SYN/ACK packet. In the above three cases, Sn_SR will be changed to SOCK_CLOSED.	
0x08	DISCON	Valid only in TCP mode; Use DISCON to disconnect either "TCP server" or "TCP client". Active shutdown: It transmits a disconnect request (FIN packet) to the connected peer. Passive shutdown: reply with a FIN packet when a FIN packet is received from the other party (peer) to the other person (peer). When a successful disconnect operation is performed (FIN/ACK packet is received), Sn_SR is changed to SOCK_CLOSED. The TCPT0 timeout interrupt occurs when the disconnect request does not receive an ACK (Sn_IR(3)= '1'), Sn_SR changed to SOCK_CLOSED. cf.> If the CLOSE command is used instead of the disconnect command, it means that directly the Sn_SR becomes SOCK_CLOSED, the disconnect mechanism FIN/ACK is no longer implemented.	

		Thus there is no disconnection process. If, when a communication is received during the communication from an opposing party (peer) RST packets, Sn_SR will unconditionally change to SOCK_CLOSED.
0x10	CLOSE	Close Socket n. Sn_SR changed to SOCK_CLOSED.
0x20	SEND	Send (SEND) Socket n Sends (TX) all buffered data in memory. For more information, see Socket n Transmit (TX) Free Size Registers (Sn_TX_FSR), Socket n Transmit (TX) Write Pointer Register (Sn_TX_WR) and Socket n Transmit (TX) Read Pointer Register (Sn_TX_RD).
0x21	SEND_MAC	Valid only in UDP mode; The basic operation is the same as sending (SEND). Generally speaking, sending (SEND) data usually requires the The destination MAC address must be obtained through an automatic ARP (Address Resolution Protocol) request in order to The transfer is performed. SEND_MAC, on the other hand, does not require the use of an automatic ARP request. In this case, the destination The MAC address is set using the host Sn_DHAR, not through the ARP process. Accessed.
0x22	SEND_KEEP	Valid only in TCP mode; The connection status is checked by sending a 1-byte online heartbeat packet. If the other party fails to return an online heartbeat packet within the timeout count period, the connection will be closed and the Triggers a timeout interrupt.
0x40	RCV	The RCV command accomplishes this by using the Receive Read Pointer Register (Sn_RX_RD) to set the RCV command in the Socket n The process of receiving data in the receive cache. For details, refer to, Socket n Receive Read Size Register (Sn_RX_RSR), Socket n Receive The pointer register (Sn_RX_WR) and Socket n receive read pointer registers. (Sn_RX_RD).

This register is used to set configuration commands such as OPEN, CLOSE, CONNECT, LISTEN, END, and RECEIVE for Socket n. Upon recognition of this command by the TOE, the Sn\_CR register is automatically cleared to 0x00. The command is still being processed even though Sn\_CR is cleared to 0x00. To verify that the command has completed, check the Sn\_IR or Sn\_SR registers.

#### Sn\_IR (Socket n Interrupt Register) [RCW1] [0x0002] [0x00]

The Sn\_IR register is used to provide the Socket n interrupt type information such as Establishment, Termination, Receiving data and Timeout. When an interrupt is triggered and the corresponding bit of Sn\_IMR is '1', the corresponding bit of Sn\_IR will also be set to '1'. The host should '1' the Sn\_IR bit if it wants to clear it to zero.

7	6	5	4	3	2	1	0
Reserved	Reserved	Reserved	SEND_OK	TIMEOUT	RCV	DISCON	CON

Bit	notation	clarification
7-5	Reserved	reserved bit
4	SEND_OK	Sn_IR(SENDOK) This bit takes effect when the SEND command completes.
3	TIMEOUT	Sn_IR(TIMEOUT) Interrupts This bit takes effect when the ARPT0 or TCPT0 timeout is triggered.
2	RCV	Sn_IR(RCV) Interrupt This bit takes effect whenever data is received from the other side.
1	DISCON	Sn_IR(DISCON) Interrupt This bit takes effect when a FIN or FIN/ACK packet is received from the other party.
0	CON	Sn_IR(CON) Interrupt When a connection is successfully established with the other party and Sn_SR changes to the SOCK_ESTABLISHED state, the This bit takes effect.



## Sn\_SR (Socket n Status Register) [R] [0x0003] [0x00]

Sn\_SR indicates the state of Socket n and changes according to Sn\_CR or some special control packets in TCP mode, such as SYN, FIN packets.

(be) worth	notation	clarification
0x00	SOCK_CLOSED	This bit indicates that Socket n is in the shutdown state and resources are released. When the DISCON, CLOSE command takes effect or when a timeout interrupt is triggered, the TOE The corresponding socket n ignores the previous state and becomes SOCK_CLOSED.
0x13	SOCK_INIT	This bit indicates that the Socket n port is open and in TCP operating mode. When Sn_MR (P[3:0]) = '0001' and the OPEN command is in effect, Sn_SR The user can only use LISTEN CONNECT command.
0x14	SOCK_LISTEN	This bit indicates that Socket n is operating in TCP server mode. and waiting for a connection request (SYN Packet) from the other party (TCP client). When the connection request is successfully accepted, Socket_SR changes to SOCK_ESTABLISHED state. Otherwise it will be triggered when TCPT0 is triggered. After a timeout interrupt, it changes to the SOCK_CLOSED state.
0x17	SOCK_ESTABLISHED	Indicates the connection status of Socket n. SOCK_LISTEN state, when the TCP server processes TCP client SYN request packet or when the CONNECT command is successfully configured, becomes SOCK_ESTABLISHED. In this state, you can use the SEND or RECV commands to perform packet Transmission.
0x1C	SOCK_CLOSE_WAIT	Indicates that Socket n received a disconnect request from the other side of the connection. (FIN packet). This is a semi-closed state that allows for data transfer. To turn it all off, you need to use the DISCON command. And if it is to turn off the Socket, the CLOSE command is required.
0x22	SOCK_UDP	Indicates that Socket n is in UDP mode (Sn_MR(P[3:0]) = '0010'). When Sn_MR(P[3:0]) = '0010' and the OPEN command is in effect, Sn_SR Change to SOCK_UDP. Unlike TCP mode, in this mode, packets can be sent in a connectionless process. transmission in the case of the
0x42	SOCK_MACRAW	Indicates that Socket 0 is operating in MACRAW mode (S0_MR(P[3:0]) = '0100').MACRAW mode is effective only under Socket 0. When S0_MR(P[3:0]) = '0100' and the OPEN command is in effect, Sn_SR Change to SOCK_MACRAW. As in UDP mode, Socket 0, when operating in MACRAW mode, also MAC packets (Ethernet frames) can be realized without a connection process. Transmission.

The following table shows the temporary state when the state of Socket n changes.

(be) worth	notation	clarification
0x15	SOCK_SYSENT	Indicates that Socket n has sent a connection request (SYN Packet) to the other side. He shows that after sending the CONNECT command, the sn_SR is removed from the SOCK_INIT to the temporary state of SOCK_ESTABLISHED. If, at this point, a request to accept the connection is received from the other party (SYN/ACKpac), the connection is not accepted. ket) then, becomes SOCK_ESTABLISHED. Otherwise, after a TCPT0 timeout (Sn_IR[TIMEOUT] = '1') interrupt, the Transforms to SOCK_CLOSED.
0x16	SOCK_SYNRCV	Indicates that Socket n successfully received a connection request packet from the other side (SYN). packet). If Socket n successfully sends a connection answer (SYN) to the other party, it is not possible for Socket n to send a connection answer (SYN) to the other party.

		/ACK packet), will transition to the SOCK_ESTABLISHED state. Otherwise, after triggering a timeout interrupt (Sn_IR[TIMEOUT] = '1'), the changed to SOCK_CLOSED.
0x18	SOCK_FIN_WAIT	These conditions indicate that SOCKET n is shutting down. This shows the process of disconnecting (active or passive shutdown). When the disconnect procedure completes successfully or TCPT0(Sn_IR(timeout)='1') occurs, the It then changes to SOCK_CLOSED.
0x1A	SOCK_CLOSING	
0x1B	SOCK_TIME_WAIT	
0x1D	SOCK_LAST_ACK	indicates that Socket n is in the passive shutdown state. Waiting for a response to a disconnect request (FIN packet) (FIN/ACK packet). When Socket n successfully receives a response to a disconnect request or a departure timeout is reached interrupt, it changes to the SOCK_CLOSED state.

### Sn\_PORT (Socket n Source Port Register) [R/W] [0x0004-0x0005] [0x0000]

This register configures Socket n's source port number. This register takes effect when Socket n is operating in TCP or UDP mode.

**Notes:** *Setting of this register must be completed before the OPEN command takes effect.*  
For example, port = 5000 (0x1388) for SOCKET 0, the configuration should be as follows:

0x0004	0x0005
0x13	0x88

### Sn\_DHAR (Socket n Destination MAC Address Register) [R/W] [0x0006-0x000B]

#### [0xFFFFFFFFFFFFFFFF]

The Sn\_DHAR register indicates the MAC address of the destination host for Socket n in UDP mode using the Send\_MAC configuration command, or the MAC address obtained by the ARP process using the CONNECT/SEND configuration command.

For example, Socket 0's destination MAC address = 08.DC.00.01.02.10 should be configured as follows:

0x0006	0x0007	0x0008	0x0009	0x000A	0x000B
0x08	0xDC	0x00	0x01	0x02	0x0A

### Sn\_DIPR (Socket n Destination IP Address Register) [R/W] [0x000C-0x000F] [0x00000000]

Sn\_DIPR Configured or indicated as the destination host IP address for Socket n, effective in TCP/UDP mode.

In TCP client mode, this register sets the IP address of the TCP server before the CONNECT configuration command.

In TCP server mode, he shows the IP address of the TCP client after a successful connection is established; in UDP mode, he configures the IP address of the other host to receive UDP packets after the SEND or SEND\_MAC configuration command.

For example, if the destination IP address of Socket 0 = 192.168.0.11, the configuration should be as follows:

0x000C	0x000D	0x000E	0x000F
192 (0xC0)	168 (0xA8)	0 (0x00)	11 (0x0B)

### Sn\_DPORT (Socket n Destination Port Register) [R/W] [0x0010-0x0011] [0x00]

Sn\_DPORT Configures or indicates the destination host port number for Socket n, effective in TCP/UDP mode.

In TCP client mode, this register configures the port number on which TCP Server listens before the CONNET configuration command.

In TCP server mode, he shows the port number of the TCP client after a successful connection has been established;

In UDP mode, he receives UDP packets after configuring the port number of the other host for the SEND or SEND\_MAC configuration command.

For example, if the destination port number for Socket 0 = 5000 (0x1388), the configuration should be as follows:

0x0010	0x0011
0x13	0x88

#### **Sn\_MSSR (Socket n Maximum Segmentation Register) [R/W] [0x0012-0x0013] [0x0000]**

This register configures or displays the Maximum Transfer Unit (MTU) of Socket n. The register is used to configure or display the MTU of Socket n. The register is used to configure or display the MTU of Socket n.

In TCP/UDP mode, the default maximum transmission unit set by this register takes effect.

However, in PPPoE mode (MR[PPPoE] = '1'), this register will depend on the maximum transmission unit for PPPoE.

Mode	Normal (MR[PPPoE]='0')		PPPoE (MR[PPPoE]='1')	
	Default MTU	Range	Default MTU	Range
TCP	1460	1 ~ 1460	1452	1 ~ 1452
UDP	1472	1 ~ 1472	1464	1 ~ 1464
MACRAW	1514			

When Socket n is in MACRAW mode, the default MTU will be in effect because the MTU is not processed internally. Therefore, when transferring data larger than the default MTU, the host needs to manually divide the data into the default MTU size units for transmission.

When Socket n is in TCP/UDP mode and the data being transferred is larger than the MTU, the data will be automatically divided into the default MTU unit size for transmission.

In UDP mode, MTU configuration is used because there is no connection process involved as in TCP mode. When MTU data of different sizes is transmitted to the other side, ICMP packets (MTU fragmentation) may be received. In this case, IR(FMTU) is set to '1' and the other party's information such as MTU size and IP address will be specified by FMTUR and UIPR respectively. If IR[MTU] = '1', the user cannot send data to the other party. To resume communication with the other party, you can do the following:

1. Shut down the Socket with the CLOSED configuration command.
2. Set Sn\_MSS to specify the MTU in FMTUR.
3. Open Socket n with the OPEN configuration command.
4. Re-communicate with each other.

For example, Socket 0 with MSS = 1460(0x05B4) should be configured as follows:

0x0012	0x0013
0x05	0xB4

#### **Sn\_TOS (Socket n IP Type of Service Register) [R/W] [0x0015] [0x00]**

This register is set in the TOS (Type of Service) field of the IP header at the IP level. It should be set before the OPEN command is executed.

Please refer to <http://www.iana.org/assignments/ip-parameters>.

#### **Sn\_TTL (Socket n Survival Time Register) [R/W] [0x0016] [0x80]**

This register is set in the TTL (Time-To-Live) field of the IP header in the IP layer. It should be set before the OPEN command is executed.

Please refer to <http://www.iana.org/assignments/ip-parameters>.

#### **Sn\_RXBUF\_SIZE (Socket n Receive Buffer Size Register) [R/W] [0x001E] [0x02]**

Sn\_RXBUF\_SIZE configures the receive buffer size for Socket n. The Socket n receive buffer size can be configured as 1, 2, 4, 8, and 16 Kbytes. if configured to any other size, the TOE will not be able to receive data from the other host properly.

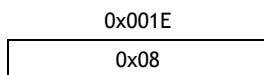
Even though the receive buffer size of Socket n is initially defaulted to 2Kbytes, the user can still redefine it using Sn\_RXBUF\_SIZE. However, the total size of the receive buffer (Sn\_RXBUF\_SIZE) for all sockets must not exceed 16Kbytes; otherwise, a receive exception will occur.

When all Sn\_RXBUF\_SIZE are configured, 16Kbytes of receive memory will be allocated to each socket in order from socket 0 to socket 7 to be used as receive cache.

Regardless of the size of Socket n's receive buffer configuration, it can be addressed by a 16-bit offset address. (Addressing range: 0x0000 to 0xFFFF)

Value (dec)	0	1	2	4	8	16
Buffer size	0KB	1KB	2KB	4KB	8KB	16KB

Example: Socket 0 RX Buffer Size = 8KB



### Sn\_TXBUF\_SIZE (Socket n Transmit Buffer Size Register) [R/W] [0x001F] [0x02]

Sn\_TXBUF\_SIZE configures the size of the transmit buffer for Socket n. Socket n transmit buffer size can be configured as 1, 2, 4, 8, and 16Kbytes. If configured to any other size, the TOE will not be able to send data to the other host properly.

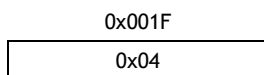
Even though the transmit buffer size of Socket n is initially defaulted to 2Kbytes, the user can still redefine it using Sn\_RXBUF\_SIZE. However, the total size of the transmit buffer for all sockets must not exceed 16Kbytes, otherwise, a transmit exception will occur.

Once all Sn\_TXBUF\_SIZE are configured, 16Kbytes of transmit memory will be allocated to each socket in the order of socket 0 to 7 to be used as transmit cache.

Socket n can be addressed with a 16-bit offset address, regardless of the size of its receiving memory configuration. (Addressing range: 0x0000 to 0xFFFF)

Value (dec)	0	1	2	4	8	16
Buffer size	0KB	1KB	2KB	4KB	8KB	16KB

Example: Socket 0 TX Buffer Size = 4KB

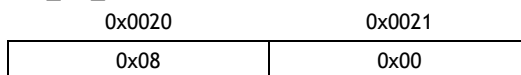


### Sn\_TX\_FSR (Socket n Idle Transmit Cache Register) [R] [0x0020-0x0021] [0x0800]

Sn\_TX\_FSR shows the size of the free space in the Socket n transmit buffer. This register is initially configured to the size of Sn\_TXBUF\_SIZE. If the length of the data to be sent is greater than the value of Sn\_TX\_FSR, the data in the transmit cache that has not been sent will be overwritten, so the TOE will not be allowed to write to the transmit cache if the data length is greater than the value of Sn\_TX\_FSR. Therefore, before sending the cached data to Socket n, you need to check if the data size is equal to or less than the remaining space before saving the data to the send cache and sending it via the SEND/SEND\_MAC configuration command. If the data is larger than the checked remaining space, you need to divide the data into sizes less than or equal to the remaining space before saving the data to the Socket n send cache. If Sn\_MR(P[3:0]) is not in TCP mode ('0001'), TOE calculates the space between the send write pointer (Sn\_TX\_WR) and the Socket n send read pointer, and automatically divides the data into the appropriate size.

If Sn\_MR(P[3:0]) is in TCP mode ('0001'), the TOE calculates the space between the transmit write pointer (Sn\_TX\_WR) and the internal ACK pointer (indicating the location of the node that has already received data from the connecting party).

Example: 2048(0x0800) at S0\_TX\_FSR



### Sn\_TX\_RD (Socket n Transmit Read Pointer Register) [R] [0x0022-0x0023] [0x0000]

The Sn\_TX\_RD register can be initialized by the OPEN configuration command. However, if Sn\_MR(P[3:0]) is in TCP mode ('0001'), this register will be re-initialized during TCP connection. After initialization, this register increments itself according to the SEND configuration command, which transfers the data currently stored in the Socket n transmit buffer between Sn\_TX\_RD and Sn\_TX\_WR. After transferring the saved data, the SEND configuration command causes Sn\_TX\_RD to equal Sn\_TX\_WR. When Sn\_TX\_RD is incremented beyond the maximum value of 0xFFFF (greater than 0x10000 and generates a rounding error), Sn\_TX\_RD ignores the rounding error and saves the value as a 16-bit lower value automatically.

### Sn\_TX\_WR (Socket n Transmit Write Pointer Register) [R/W] [0x0024-0x0025] [0x0000]

The Sn\_TX\_WR register can be initialized by the OPEN configuration command. However, if Sn\_MR(P[3:0]) is in TCP mode ('0001'), this register will be re-initialized during TCP connection. This register needs to be read or updated as follows:

1. Read the first address in the transmit cache where the transmitted data will be stored.
2. Starting from the first address corresponding to the transmit buffer of Socket n, save the data to be transmitted.
3. After saving the transmitted data, increase the value of Sn\_TX\_WR to the transmitted data size. If the increase exceeds the maximum value 0xFFFF (which is larger than 0x10000 and generates a rounding), then the rounding is automatically ignored and the value is automatically updated to the lower 16 bits.
4. Send the data stored in the Socket n transmit buffer by using the SEND command.

### Sn\_RX\_RSR (Socket n Idle Receive Cache Register) [R] [0x0026-0x0027] [0x0000]

Sn\_RX\_RSR shows the size of received and saved data in the Socket n receive cache.

Sn\_RX\_RSR does not exceed the size of n\_RXBUF\_SIZE and is calculated as the size of the space between the Socket n receive write pointer (Sn\_RX\_WR) and the Socket n receive read pointer.

Example: 2048(0x0800) at S0\_RX\_RSR

0x0026	0x0027
0x08	0x00

### Sn\_RX\_RD (Socket n Receive Read Pointer Register) [R/W] [0x0028-0x0029] [0x0000]

The Sn\_RX\_RD register can be initialized with the OPEN configuration command. Make sure that this register is read and updated according to the following procedure:

1. Read the first address of the data stored in the receive cache.
2. Start reading data from the first address of the data stored in the Socket n receive buffer.
3. After reading finished receiving data, update the value of Sn\_RX\_RD to the size of the read data. If the increased value exceeds the maximum value 0xFFFF, i.e., exceeds 0x10000 and generates a rounding, the rounding will be ignored and only the lower 16 bits will be taken.
4. After receiving the RECV command, inform the TOE of the updated Sn\_RX\_RD value.

Example: 2048(0x0800) at S0\_RX\_RD

0x0028	0x0029
0x08	0x00

### Sn\_RX\_WR (Socket n Receive Write Pointer Register) [R] [0x002A-0x002B] [0x0000]

The Sn\_RX\_WR register can be initialized with the OPEN configuration command. and is automatically incremented as data is received.

If the value of Sn\_RX\_WR grows beyond the maximum value of 0xFFFF (i.e., exceeds 0x10000 and generates a rounding), then the rounding is automatically ignored and the value is automatically updated to the lower 16 bits.

Example: 2048 (0x0800) at S0\_RX\_WR

0x002A	0x002B
0x08	0x00

### Sn\_IMR (Socket n Interrupt Mask Register) [R/W] [0x002C] [0xFF]

Sn\_IMR is responsible for blocking the interrupt of Socket n. Each bit corresponds to the corresponding bit in the Sn\_IR register. Each bit corresponds to the corresponding bit in the Sn\_IR register, and when the interrupt of Socket n is triggered and the corresponding bit of Sn\_IMR is '1', the corresponding bit of Sn\_IR becomes '1'. If the corresponding bits of Sn\_IMR and Sn\_IR are both '1' and the corresponding bit of the IR register is '1', the INTn pin is pulled low and the host generates an interrupt.

7	6	5	4	3	2	1	0
Reserved	Reserved	Reserved	SEND_OK	TIMEOUT	RECV	DISCON	CON

Bit	notation	clarification
7:5	Reserved	reserved bit
4	SENDOK	Sn_IR(SENDOK) Interrupt Masking
3	TIMEOUT	Sn_IR(TIMEOUT) Interrupt Masking
2	RECV	Sn_IR(RECV) Interrupt Masking
1	DISCON	Sn_IR(DISCON) Interrupt Masking
0	CON	Sn_IR(CON) Interrupt Mask

### Sn\_FRAG (Socket n Segment Register) [R/W] [0x002D-0x002E] [0x4000]

It sets the segmentation field of the IP header at the IP layer.

Example: Sn\_FRAG0 = 0x4000 (do not segment)

0x002D	0x002E
0x00	0x00

### Sn\_KPALVTR (Socket n Online Time Register) [R/W] [0x002F] [0x00]

Sn\_KPALVTR configures the 'KEEP ALIVE(KA)' in-line verification of heartbeat packet transmission time for SOCKET n. This is the first time a heartbeat packet is transmitted. It only works in TCP mode and will be ignored in other modes. The unit time is 5 seconds.

KA packets are transmitted after Sn\_SR has changed to SOCK\_ESTABLISHED and there has been at least one incoming or outgoing communication with the other party. If 'Sn\_KPALVTR > 0', the TOE automatically transmits KA packets to check the connection status of TCP at a certain time period (automatic online verification). If 'Sn\_KPALVTR = 0', automatic online verification will not be initiated and the host can send KA packets via the SEND\_KEEP configuration command (manual online verification). With 'Sn\_KPALVTR > 0', manual online authentication will be ignored.

Example: Sn\_KPALVTR = 10 (will automatically send an online verification packet every 50 seconds)

0x002F
0x0A

---

## 10 Direct Memory Access controller (DMA)

### 10.1 Introduction to DMA

Direct memory access (DMA) is used to provide high-speed data transfers between peripherals and memory or between memory and memory. Without CPU intervention, data can be moved quickly through DMA, which saves CPU resources for other operations.

The two DMA controllers have 12 channels (7 channels for DMA1 and 5 channels for DMA2), 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.

### 10.2 DMA Key Features

- 12 independently configurable channels (requests): DMA1 with 7 channels, DMA2 with 5 channels
- Each channel is directly connected to a dedicated hardware DMA request, and each channel equally supports software triggering. These features are configured through software.
- The priority between multiple requests on the same DMA module can be set programmatically by software (there are four levels: very high, high, medium, and low), and the hardware determines when the priority settings are equal (request 0 is prioritized over request 1, and so on).
- The transmission widths (byte, half-word, full-word) of the independent data source and destination data areas simulate the process of packing and unpacking. Source and destination addresses must be aligned by data transfer width.
- Supports cyclic buffer management
- Each channel has 3 event flags (DMA Half Transfer, DMA Transfer Complete, and DMA Transfer Error) which logically or become a single interrupt request.
- Transfers between memories and memories
- Transfers between peripherals and memory, memory and peripherals
- Flash memory, SRAM, SRAM of peripherals, APB1, APB2 and AHB peripherals can be used as access sources and targets.
- Number of programmable data transfers: up to 65535

Below is a functional block diagram:



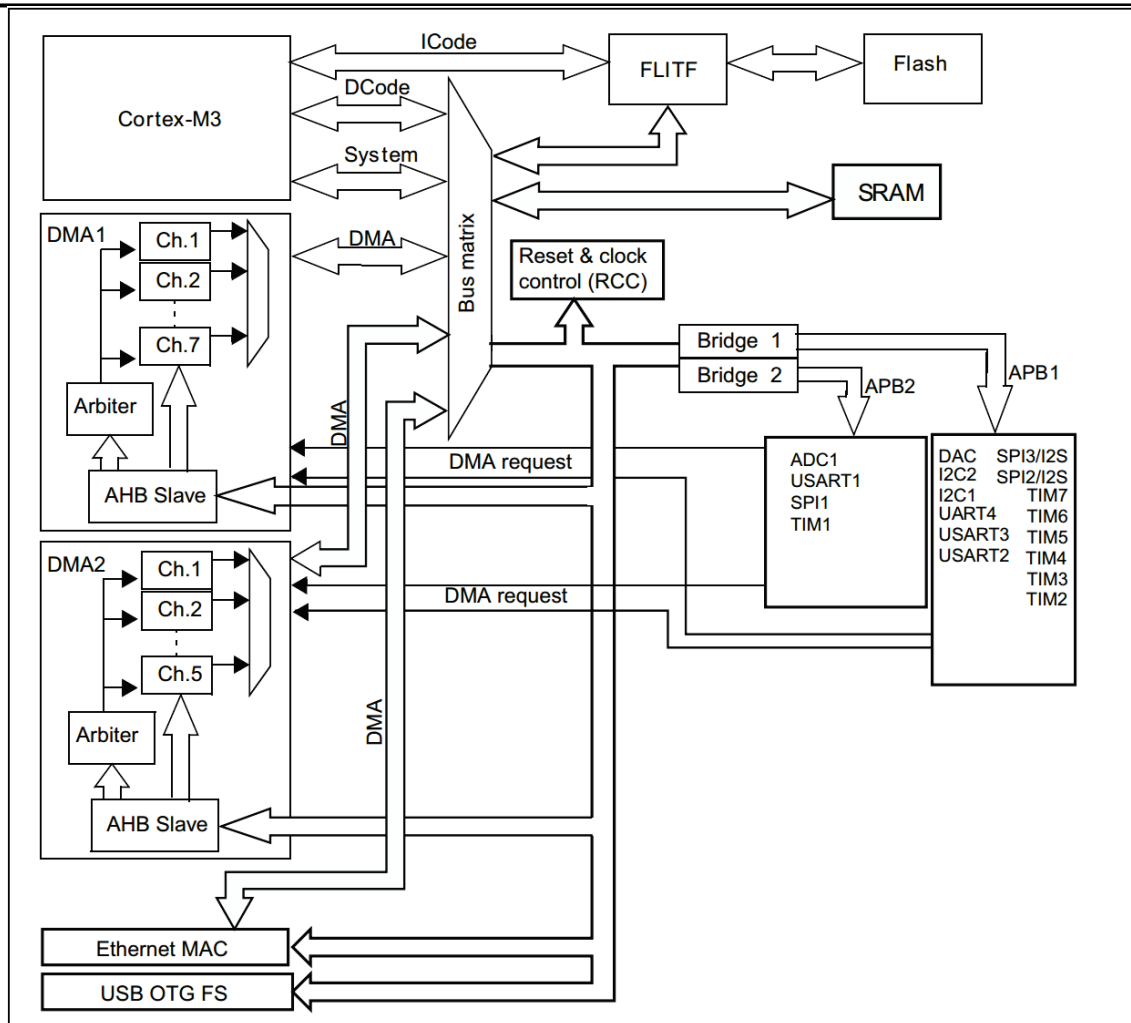


Figure19 DMA Block Diagram

## 10.3 Functional Description

The DMA controller shares the system data bus with the Cortex™ -M3 core to perform direct memory data transfers. When the CPU and DMA access the same target (RAM or peripheral) at the same time, the DMA request suspends the CPU from accessing the system bus for a number of cycles, and the bus arbiter performs round-robin scheduling to ensure that the CPU gets at least half of the system bus (memory or peripheral) bandwidth.

### 10.3.1 DMA Processing

After an event occurs, 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 requesting peripheral, the DMA controller immediately sends it an answer signal. When an answer signal is received from the DMA controller, the peripheral immediately releases its request. Once the peripheral releases the request, the DMA controller simultaneously revokes the answer signal. If there are more requests, the peripheral can start the next cycle.

In summary, each DMA transfer consists of 3 operations:

- Data is fetched from the peripheral data register or from the memory address indicated by the current peripheral/memory address register, and the start address for 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 data register or the memory address indicated by the current peripheral/memory address register, with the start address on the first transfer being the peripheral base 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.



### 10.3.2 Arbiter

The arbiter initiates peripheral/memory accesses based on the priority of the channel request. Priority is managed 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. As an example, channel 2 has priority over channel 4.

### 10.3.3 DMA channel

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

#### Programmable data volume

The amount of data transferred from peripherals and memory can be programmed with the PSIZE and MSIZE bits in the DMA\_CCRx register.

#### 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 increment mode, the next address to be transferred will be the previous address plus the increment value, which depends on the selected data width of 1, 2, or 4. The first address to be transferred is the address stored in the DMA\_CPARx/DMA\_CMARx registers. During the transfer, these registers maintain their initial values, and software cannot change and read out the address 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 has ended (i.e., the transfer count becomes 0). To start a new DMA transfer, the transfer count needs to be rewritten in the DMA\_CNDTRx register with the DMA channel turned off.

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

The following is the procedure for configuring DMA channel x (x represents the channel number):

1. 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.
2. The address of the data memory is set in the DMA\_CMARx register. When a peripheral data transfer request occurs, the transferred data will be read from or written to this address.
3. The amount of data to be transferred is set in the DMA\_CNDTRx register. This value is decremented after each data transfer.
4. Set the channel priority in the PL[1:0] bits of the DMA\_CCRx register.
5. Set the direction of data transfer, cyclic mode, incremental mode for peripheral and memory, data width for peripheral and memory, generate interrupt halfway through the transfer, or generate interrupt on completion of the transfer in the DMA\_CCRx register.
6. Setting the ENABLE bit of the DMA\_CCRx register starts the channel.

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

After half of the data has been transferred, the half-transfer flag (HTIF) is set to 1, and an interrupt request is generated when the allow half-transfer interrupt bit (HTIE) is set. After the

end of data transmission, the transmission completion flag (TCIF) is set to 1, and an interrupt request is generated when the Allow transmission completion interrupt bit (TCIE) is set.

#### recurrent 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, it will automatically be restored to the initial value set when the channel was configured and the DMA operation will continue.

#### Memory to memory mode

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

When the MEM2MEM bit in the DMA\_CCRx register is set, the DMA transfer will start immediately when the DMA channel is activated by software setting the EN bit in the DMA\_CCRx register. The DMA transfer ends when the DMA\_CNDTRx register becomes zero. Memory-to-memory mode cannot be used in conjunction with cyclic mode.

### 10.3.4 Programmable Data Transfer Width, Alignment and Data Size Terminals

When PSIZE and MSIZE are not the same, the DMA module aligns the data according to the following table.

Table55 Programmable Data Transfer Width and Size End Operation (when PINC=MINC=1)

source width	target width	transmission number	Source: Address/Data	transport operation	Target: Address/Data
8	8	4	0x0/B0 0x1/B1 0x2/B2 0x3/B3	1:Read B0[7:0] at 0x0, write B0[7:0] at 0x0 2:Read B1[7:0] at 0x1, write B1[7:0] at 0x1 3:Read B2[7:0] at 0x2, write B2[7:0] at 0x2 4:Read B3[7:0] at 0x3, write B3[7:0] at 0x3	0x0/B0 0x1/B1 0x2/B2 0x3/B3
8	16	4	0x0/B0 0x1/B1 0x2/B2 0x3/B3	1:Read B0[7:0] at 0x0, write 00B0[15:0] at 0x0 2:Read B1[7:0] at 0x1, write 00B1[15:0] at 0x2 3:Read B2[7:0] at 0x2 and write 00B2[15:0] at 0x4 4:Read B3[7:0] at 0x3 and write 00B3[15:0] at 0x6	0x0/00B0 0x2/00B1 0x4/00B2 0x6/00B3
8	32	4	0x0/B0 0x1/B1 0x2/B2 0x3/B3	1:Read B0[7:0] at 0x0, write 000000B0[31:0] at 0x0 2:Read B1[7:0] at 0x1, write 000000B1[31:0] at 0x4 3:Read B2[7:0] at 0x2, write 000000B2[31:0] at 0x8 4:Read B3[7:0] at 0x3, write 000000B3[31:0] at 0xC	0x0/000000B0 0x4/000000B1 0x8/000000B2 0xC/000000B3
16	8	4	0x0/B1B0 0x2/B3B2 0x4/B5B4 0x6/B7B6	1:Read B1B0[15:0] at 0x0, write B0[7:0] at 0x0 2: Read B3B2[15:0] at 0x2, write B2[7:0] at 0x1 3:Read B5B4[15:0] at 0x4 and write B4[7:0] at 0x2 4:Read B7B6[15:0] at 0x6 and write B6[7:0] at 0x3	0x0/B0 0x1/B2 0x2/B4 0x3/B6
16	16	4	0x0/B1B0 0x2/B3B2 0x4/B5B4 0x6/B7B6	1:Read B1B0[15:0] at 0x0, write B1B0[15:0] at 0x0 2:Read B3B2[15:0] at 0x2, write B3B2[15:0] at 0x2 3:Read B5B4[15:0] at 0x4, write B5B4[15:0] at 0x4 4:Read B7B6[15:0] at 0x6, write B7B6[15:0] at 0x6	0x0/B1B0 0x2/B3B2 0x4/B5B4 0x6/B7B6
16	32	4	0x0/B1B0 0x2/B3B2 0x4/B5B4 0x6/B7B6	1:Read B1B0[15:0] at 0x0, write 0000B1B0[31:0] at 0x0 2:Read B3B2[15:0] at 0x2, write 0000B3B2[31:0] at 0x4 3:Read B5B4[15:0] at 0x4, write 0000B5B4[31:0] at 0x8 4:Read B7B6[15:0] at 0x6, write 0000B7B6[31:0] at 0xC	0x0/0000B1B0 0x4/0000B3B2 0x8/0000B5B4 0xC/0000B7B6
32	8	4	0x0/B3B2B1B0 0x4/B7B6B5B4 0x8/BBBAB9B8 0xC/BFBEBDBC	1:Read B3B2B1B0[31:0] at 0x0, write B0[7:0] at 0x0 2:Read B7B6B5B4[31:0] at 0x4 and write B4[7:0] at 0x1 3:Read BBBAB9B8[31:0] at 0x8 and write B8[7:0] at 0x2 4:Read BFBEBDBC[31:0] at 0xC and write BC[7:0] at 0x3	0x0/B0 0x1/B4 0x2/B8 0x3/BC
32	16	4	0x0/B3B2B1B0 0x4/B7B6B5B4 0x8/BBBAB9B8 0xC/BFBEBDBC	1:Read B3B2B1B0[31:0] at 0x0, write B1B0[15:0] at 0x0 2:Read B7B6B5B4[31:0] at 0x4, write B5B4[15:0] at 0x2 3:Read BBBAB9B8[31:0] at 0x8 and write B9B8[15:0] at 0x4 4:Read BFBEBDBC[31:0] at 0xC and write BDBC[15:0] at 0x6	0x0/B1B0 0x2/B5B4 0x4/B9B8 0x6/BDBC
32	32	4	0x0/B3B2B1B0 0x4/B7B6B5B4 0x8/BBBAB9B8 0xC/BFBEBDBC	1:Read B3B2B1B0[31:0] at 0x0, write B3B2B1B0[31:0] at 0x0 2:Read B7B6B5B4[31:0] at 0x4, write B7B6B5B4[31:0] at 0x4 3:Read BBBAB9B8[31:0] at 0x8, write BBBAB9B8[31:0] at 0x8 4:Read BFBEBDBC[31:0] at 0xC, write BFBEBDBC[31:0] at 0xC	0x0/B3B2B1B0 0x4/B7B6B5B4 0x8/BBBAB9B8 0xC/BFBEBDBC

#### Operate an AHB device that does not support byte or halfword writing

When the DMA module begins an AHB byte or half-word write operation, the data will be duplicated in the unused portion of the HWDATA[31:0] bus. Therefore, 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 as in the following two examples:

- When HSIZE=Halfword, write the halfword '0xABCD' and the DMA will set the HWDATA bus to '0xABCD ABCD'.
- When HSIZE=byte, write byte '0xAB' and the DMA will set the HWDATA bus to '0xABAB ABAB'.
- 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) is converted to a write byte data '0xB0B0 B0B0' 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 '0xB1B0 B1B0' operation on an APB to address 0x0.

For example, to write to the APB back-up register (a 16-bit register aligned with a 32-bit address), you need to configure the memory data source width (MSIZE) to '16-bit' and the peripheral target data width (PSIZE) to '32-bit'.

### 10.3.5 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 in the Channel Configuration Register (DMA\_CCRx) corresponding to the channel on which the error occurred, and that channel operation is halted. At this time, the Transmission Error Interrupt Flag Bit (TEIF) corresponding to that channel in the DMA\_IFR register will be set, and an interrupt will be generated if the Transmission Error Interrupt Allow bit is set in the DMA\_CCRx register.

### 10.3.6 Disruptions

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 registers.

Table56 DMA Interrupt Requests

disruption event	event marker	Enable Control Bit
more than halfway through transmission	HTIF	HTIE
Transmission complete	TCIF	TCIE
transmission error	TEIF	TEIE

Notes: The interrupts for DMA2 channel 4 and DMA2 channel 5 are mapped on the same interrupt vector.

## 10.3.7 DMA Request Mapping

### DMA1 Controller

The seven requests generated from the peripherals (TIMx[x=1, 2, 3, 4], ADC1, SPI1, I2Cx[x=1, 2], and USARTx[x=1, 2, 3]) are input to the DMA1 controller via a logical or, which means that only one request can be valid at the same time. See the DMA1 request image below.

DMA requests from peripherals can be independently turned on or off by setting control bits in the corresponding peripheral registers.

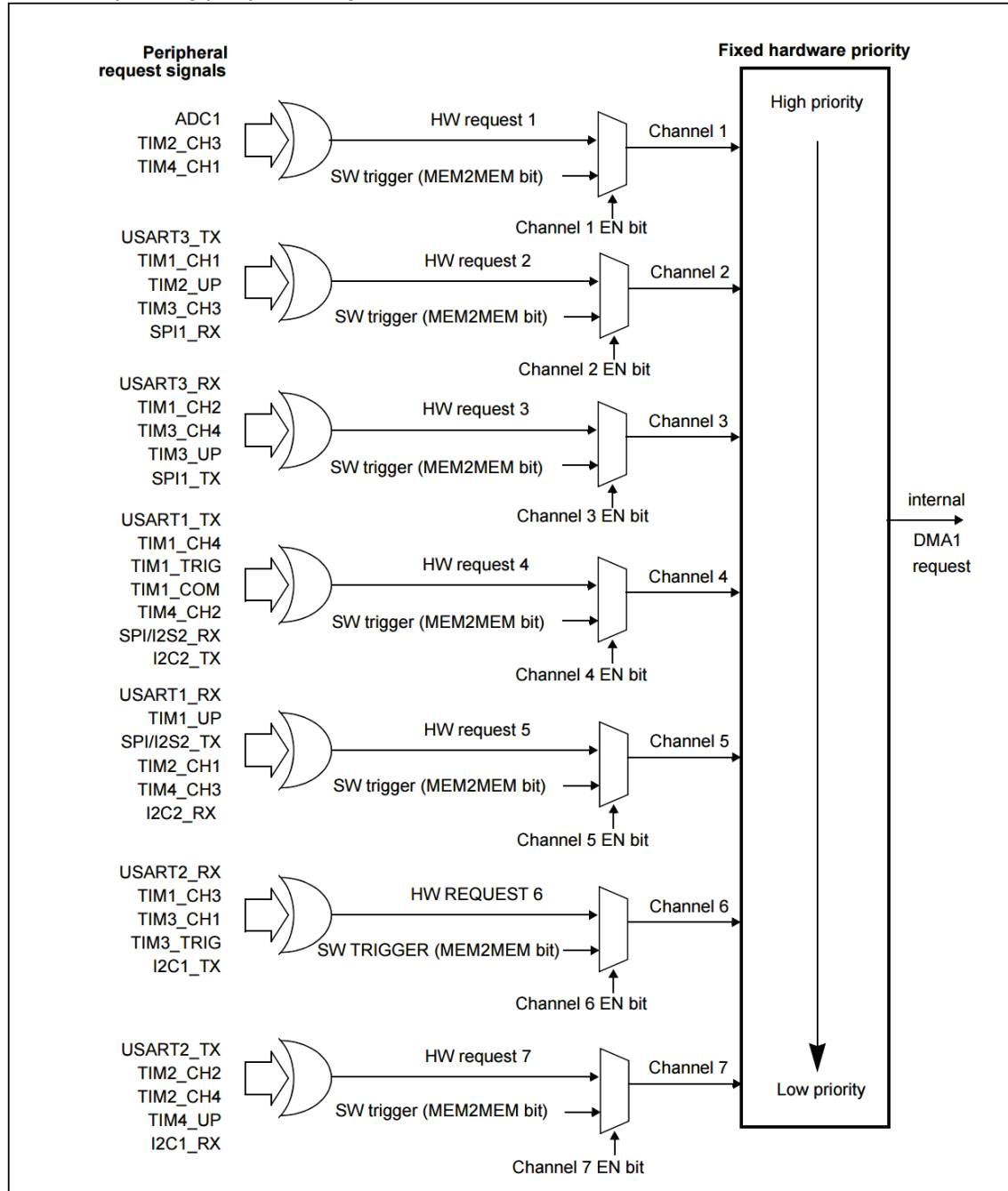


Figure20 DMA1 request mapping

Table57 List of DMA1 requests for each channel

peripherals	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5	Channel 6	Channel 7
ADC1	ADC1						
SPI/I2S		SPI1_RX	SPI1_TX				
USART		USART3_TX	USART3_RX	USART1_TX	USART1_RX	USART2_RX	USART2_TX
I2C				I2C2_TX	I2C2_RX	I2C1_TX	I2C1_RX
TIM1		TIM1_CH1	TIM1_CH2	TIM1_TX4TIM1_TRIGTIM1_COM	TIM1_UP	TIM1_CH3	
TIM2	TIM2_CH3	TIM2_UP			TIM2_CH1		TIM2_CH2TIM2_CH4
TIM3		TIM3_CH3	TIM3_CH4TIM3_UP			TIM3_CH1TIM3_TRIG	
TIM4	TIM4_CH1			TIM4_CH2	TIM4_CH3		TIM4_UP

## DMA2 Controller

The five requests generated from the peripherals (TIMx[5, 6, 7, 8], ADC3, SPI/I2S3, UART4, DAC channels 1 and 2, and SDIO) are input to the DMA2 controller via a logical or, which means that only one request can be valid at the same time. See the DMA2 request image below.

DMA requests from peripherals can be independently turned on or off by setting the DMA control bits in the corresponding peripheral registers.

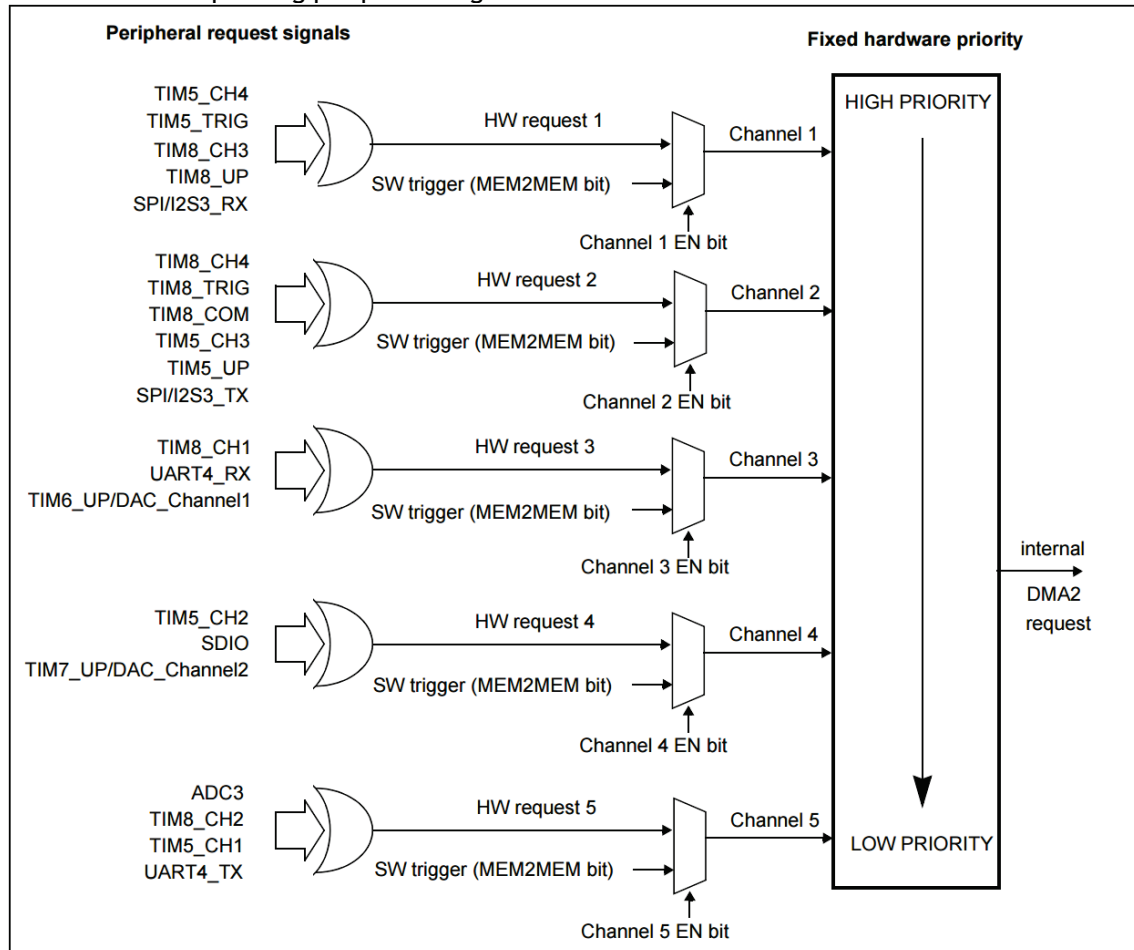


Figure21 DMA2 request mapping

Table58 List of DMA2 requests for each channel

peripherals	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5
ADC3					ADC3
SPI/I2S3	SPI/I2S3_RX	SPI/I2S3_TX			
UART4			UART4_RX		UART4_TX
SDIO				SDIO	
TIM5	TIM5_CH4TIM5_T RIG	TIM5_CH3TIM5_ UP		TIM5_CH2	TIM5_CH1
TIM6/DAC channel 1			TIM6_UP/DAC channel 1		
TIM7/DAC channel 2				TIM7_UP/DAC channel 2	
TIM8	TIM8_CH3TIM8_ UP	TIM8_CH4TIM8_T RIGTIM8_COM	TIM8_CH1		TIM8_CH2

## 10.4 DMA Registers

For abbreviations used in register descriptions, see Chapter 1.

*Notes: In all of the registers listed below, all of the bits associated with channels 6 and 7 do not apply to the DMA2, which has only five channels.*

### 10.4.1 DMA Interrupt Status Register (DMA\_ISR)

Offset address: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				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	notation	clarification
31:28	Reserved	Reserved, always reads 0.
27,23, 19,15, 11,7,3	TEIFx	TEIFx: Channel x transfer error flag (x=1..7) (Channel x transfer error flag) Hardware sets these bits. Writing a '1' to the corresponding bit in the DMA_IFCR register clears the corresponding flag bit here. 0: No transmission error (TE) at channel x; 1: A transmission error (TE) has occurred at channel x.
26, 22, 18, 14, 10, 6, 2	HTIFx	HTIFx: Channel x half transfer flag (x=1..7) (Channel x half transfer flag) Hardware sets these bits. Writing a '1' to the corresponding bit in the DMA_IFCR register clears the corresponding flag bit here. 0: No half-transmission event (HT) at channel x; 1: A half-transmission event (HT) is generated at channel x.
25, 21, 17, 13, 9, 5, 1	TCIFx	TCIFx: Channel x transfer complete flag (x=1..7) (Channel x transfer complete flag) Hardware sets these bits. Writing a '1' to the corresponding bit in the DMA_IFCR register clears the corresponding flag bit here. 0: There is no transmission completion event (TC) at channel x; 1: A transmission completion event (TC) is generated at channel x.
24,20, 16,12, 8,4,0	GIFx	GIFx: Channel x global interrupt flag (x=1..7) (Channel x global interrupt flag) Hardware sets these bits. Writing a '1' to the corresponding bit in the DMA_IFCR register clears the corresponding flag bit here. 0: No TE, HT or TC events at channel x; 1: A TE, HT, or TC event was generated at channel x.

### 10.4.2 DMA Interrupt Flag Clear Register (DMA\_IFCR)

Offset address: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				CTEIF7	CHTIF7	CTCIF7	CGIF7	CTEIF6	CHTIF6	CTCIF6	CGIF6	CTEIF5	CHTIF5	CTCIF5	CGIF5
				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
CTEIF4	CHTIF4	CTCIF4	CGIF4	CTEIF3	CHTIF3	CTCIF3	CGIF3	CTEIF2	CHTIF2	CTCIF2	CGIF2	CTEIF1	CHTIF1	CTCIF1	CGIF1
rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW

Bit	notation	clarification
31:28	Reserved	Reserved, always reads 0.
27,23,1 9,15, 11,7,3	CTEIFx	<b>CTEIFx</b> : clears the channel x transfer error flags (x=1..7) (Channel x transfer error clear) These bits are set and cleared by software. 0: not working 1: Clear the corresponding TEIF flag in the DMA_ISR register.
26, 22, 18, 14, 10, 6, 2	CHTIFx	<b>CHTIFx</b> : clears the half transfer flags (x=1..7) for channel x. (Channel x half transfer clear) These bits are set and cleared by software. 0: not working 0: Clear the corresponding HTIF flag in the DMA_ISR register.
25, 21, 17, 13, 9, 5, 1	CTCIF	<b>CTCIFx</b> : clears the channel x transfer complete flag (x=1..7) (Channel x transfer complete clear) These bits are set and cleared by software. 0: not working 0: Clear the corresponding TCIF flag in the DMA_ISR register.
24,20,1 6,12,8, 4,0	CGIFx	<b>CGIFx</b> : clears the global interrupt flags (x=1..7) for channel x (Channel x global interrupt clear) These bits are set and cleared by software. 0: not working 0: Clear the corresponding GIF, TEIF, HTIF, and TCIF flags in the DMA_ISR register.

### 10.4.3 DMA Channel x Configuration Register (DMA\_CCRx) (x=1..7)

Offset address: 0x08+0d20x (channel number-1)

Reset value: 0x0000 0000

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

Bit	notation	show
31:15	Reserved	Reserved, always reads 0.
14	MEM2MEM	<b>MEM2MEM</b> : Memory to memory mode This bit is set and cleared by software. 0: Non-memory to memory mode; 1: Initiate memory to memory mode.
13:12	PL[1:0]	<b>PL[1:0]</b> : Channel priority level (Channel priority level) These bits are set and cleared by software. 00: Low 01: Medium 10: High 11: Maximum
11:10	MSIZE[1:0]	<b>MSIZE[1:0]</b> : memory data width (Memory size) These bits are set and cleared by software. 00: 8 bits 01: 16 bits 10: 32 bits 11: Reserved
9:8	PSIZE[1:0]	<b>PSIZE[1:0]</b> : Peripheral data width (Peripheral size) These bits are set and cleared by software. 00: 8 bits 01: 16 bits 10: 32 bits 11: Reserved
7	MINC	<b>MINC</b> : Memory address increment mode (Memory increment mode) This bit is set and cleared by software. 0: Memory address increment operation is not performed 1: Perform memory address increment operation
6	PINC	<b>PINC</b> : Peripheral address increment mode (Peripheral increment mode) This bit is set and cleared by software. 0: Peripheral address increment operation is not performed 1: Perform peripheral address increment operation
5	CIRC	<b>CIRC</b> : Circular mode This bit is set and cleared by software. 0: No loop operation is performed 1: Perform cyclic operations
4	DIR	<b>DIR</b> : Data transfer direction This bit is set and cleared by software. 0: Read from peripheral 1: Read from memory
3	TEIE	<b>TEIE</b> : Transfer error interrupt enable This bit is set and cleared by software. 0: Disable TE interrupt 0: TE interrupt allowed
2	HTIE	<b>HTIE</b> : Half transfer interrupt enable This bit is set and cleared by software. 0: HT interrupt disabled



		0: HT interrupt allowed
1	TCIE	TCIE: Transfer complete interrupt enable This bit is set and cleared by software. 0: Disable TC interrupt 1: TC interrupt allowed
0	EN	EN: Channel enable This bit is set and cleared by software. 0: Channel not working 1: Channel opening

#### 10.4.4 DMA Channel x Transfer Count Register (DMA\_CNDTRx) (x=1..7)

Offset address: 0x0C+0d20x (channel number-1)

Reset value: 0x0000 0000

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

Bit	notation	clarification
31:16	Reserved	Reserved, always reads 0.
15:0	NDT [15:0]	NDT[15:0] : Number of data to transfer The number of data transfers is from 0 to 65535. This register can only be written when the channel is not operational (EN=0 for DMA_CCRx). When the channel is on this register becomes read-only and indicates the number of bytes remaining to be transferred. The register contents are decremented after each DMA transfer. At the end of the data transfer, the contents of the registers either change to 0; or when the channel is configured for auto-reload mode, the contents of the registers are automatically reloaded to the values they had when previously configured. When the contents of the register are 0, no data transfer occurs, regardless of whether the channel is on or off.

#### 10.4.5 DMA Channel x Peripheral Address Register (DMA\_CPARx) (x=1..7)

Offset address: 0x10+0d20x (channel number-1)

Reset value: 0x0000 0000

This register cannot be written when the channel is turned on (EN=1 for DMA\_CCRx).

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

Bit	notation	clarification
31:0	Reserved	PA[31:0]: Peripheral address (Peripheral address) The base address of the peripheral data register that serves as the source or destination of the data transfer. When PSIZE='01' (16 bits), the PA[0] bit is not used. The operation is automatically aligned to the halfword address. When PSIZE='10' (32 bits), the PA[1:0] bits are not used. The operation is automatically aligned to the word address.

#### 10.4.6 DMA Channel x Memory Address Register (DMA\_CMARx) (x=1..7)

Offset address: 0x14+20x (channel number-1)

Reset value: 0x0000 0000

This register cannot be written when the channel is turned on (EN=1 for DMA\_CCRx).

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

Bit	notation	clarification
31:0	MA[31:0]	MA[31:0]: memory address The memory address is used as the source or target of the data transfer. When MSIZE='01' (16 bits), the MA[0] bit is not used. The operation is automatically aligned to the halfword address. When MSIZE='10' (32 bits), the MA[1:0] bits are not used. The operation is automatically aligned to the word address.



## 10.4.7 DMA Register Image

Table59 DMA Register Images and Resets

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
000h	DMA_ISR	Reserved				TEIF7	HTIF7	TCIF7	GIF7	TEIF6	HTIF6	TCIF6	GIF6	TEIF5	HTIF5	TCIF5	GIF5	TEIF4	HTIF4	TCIF4	GIF4	TEIF3	HTIF3	TCIF3	GIF3	TEIF2	HTIF2	TCIF2	GIF2	TEIF1	HTIF1	TCIF1	GIF1		
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
004h	DMA_IFCR	Reserved				CTEIF7	CHTIF7	CTCIF7	CGIF7	CTEIF6	CHTIF6	CTCIF6	CGIF6	CTEIF5	CHTIF5	CTCIF5	CGIF5	CTEIF4	CHTIF4	CTCIF4	CGIF4	CTEIF3	CHTIF3	CTCIF3	CGIF3	CTEIF2	CHTIF2	CTCIF2	CGIF2	CTEIF1	CHTIF1	CTCIF1	CGIF1		
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
008h	DMA_CCR1	Reserved																	MEM2MEM	PL [1:0]	MSIZE [1:0]	PSIZE [1:0]	MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN					
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0					
00Ch	DMA_CNDTR1	Reserved																	NDT [15:0]																
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
010h	DMA_CPAR1	PA [31:0]																																	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
014h	DMA_CMAR1	MA[31:0]																																	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
018h	Reserved																																		
01Ch	DMA_CCR2	Reserved																	MEM2MEM	PL [1:0]	MSIZE [1:0]	PSIZE [1:0]	MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN					
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
020h	DMA_CNDTR2	Reserved																	NDT [15:0]																
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
024h	DMA_CPAR2	PA [31:0]																																	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
028h	DMA_CMAR2	MA[31:0]																																	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
02Ch	Reserved																																		
030h	DMA_CCR3	Reserved																	MEM2MEM	PL [1:0]	MSIZE [1:0]	PSIZE [1:0]	MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN					
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
034h	DMA_CNDTR3	Reserved																	NDT [15:0]																
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
038h	DMA_CPAR3	PA [31:0]																																	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
03Ch	DMA_CMAR3	MA[31:0]																																	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
040h	Reserved																																		
044h	DMA_CCR4	Reserved																	MEM2MEM	PL [1:0]	MSIZE [1:0]	PSIZE [1:0]	MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN					
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
048h	DMA_CNDTR4	Reserved																NDT [15:0]																
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
04Ch	DMA_CPAR4	PA [31:0]																																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
050h	DMA_CMAR4	MA[31:0]																																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
054h	Reserved																																	
058h	DMA_CCR5	Reserved																MEM2MEM	PL [1:0]	MSIZE [1:0]	PSIZE [1:0]	MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN					
	Reset value																													0	0	0		
05Ch	DMA_CNDTR5	Reserved																NDT [15:0]																
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
060h	DMA_CPAR5	PA [31:0]																																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
064h	DMA_CMAR5	MA[31:0]																																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
068h	Reserved																																	
06Ch	DMA_CCR6	Reserved																MEM2MEM	PL [1:0]	MSIZE [1:0]	PSIZE [1:0]	MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN					
	Reset value																													0	0	0		
070h	DMA_CNDTR6	Reserved																NDT [15:0]																
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
074h	DMA_CPAR6	PA [31:0]																																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
078h	DMA_CMAR6	MA[31:0]																																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
07Ch	Reserved																																	
080h	DMA_CCR7	Reserved																MEM2MEM	PL [1:0]	MSIZE [1:0]	PSIZE [1:0]	MINC	PINC	CIRC	DIR	TEIE	HTIE	TCIE	EN					
	Reset value																													0	0	0		
084h	DMA_CNDTR7	Reserved																NDT [15:0]																
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
088h	DMA_CPAR7	PA [31:0]																																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
08Ch	DMA_CMAR7	MA[31:0]																																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
090h	Reserved																																	

SeeTable1 for register start addresses.

---

## 11 Analog/Digital Conversion (ADC)

### 11.1 ADC Introduction

The 12-bit ADC is a successive approximation analog-to-digital converter. It has up to 18 channels and can measure 16 external and 2 internal signal sources. The A/D conversion of each channel can be performed in single, continuous, sweep or intermittent modes. The ADC results can be stored in 16-bit data registers in a left- or right-aligned manner.

The analog watchdog feature allows the application to detect if the input voltage exceeds a user-defined high/low threshold.

The input clock to the ADC must not exceed 14MHz, which is generated from PCLK2 by dividing.

### 11.2 ADC Key Features

- 12-bit resolution
- Interrupts are generated at the end of conversion, at the end of injection conversion, and when an analog watchdog event occurs
- Single and continuous conversion modes
- Auto-scan mode from channel 0 to channel n
- self-calibration
- Data alignment with embedded data consistency
- Sampling intervals can be programmed individually by channel
- External trigger options are available for both rule conversions and injection conversions
- intermittent mode
- Dual mode (devices with 2 or more ADCs)
- ADC conversion time:

1  $\mu$ s at a clock of 56 MHz (1.17  $\mu$ s at a clock of 72 MHz)

- ADC power requirements: 2.4V to 3.6V
- ADC input range:  $V_{REF-} \leq V_{IN} \leq V_{REF+}$
- There are DMA requests generated during the rule channel transitions. The following figure shows the block diagram of the ADC module.

*Notes: If there is a VREF-pin, it must be connected to VSSA*

## 11.3 ADC Functional Description

The following figure shows the block diagram of an ADC module. Table 60 shows the description of the ADC pins.

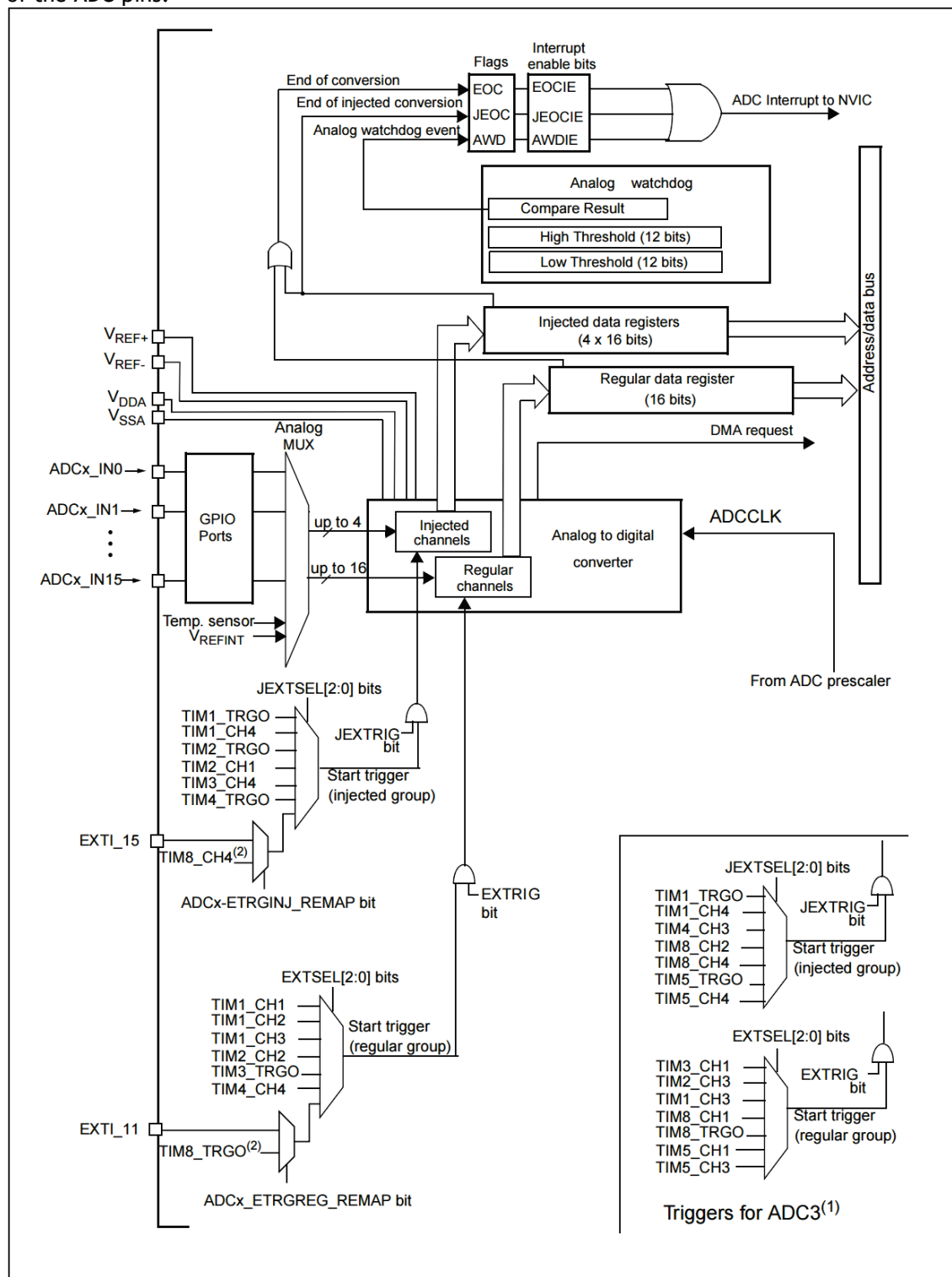


Figure 22 Single ADC Block Diagram

1. The regular conversion and injection conversion triggers of ADC3 are different from those of ADC1 and ADC2.

Table60 ADC Pins

name (of a thing)	Signal Type	clarification
VREF+	Input, Analog Reference Positive	High end/positive reference voltage used by ADC, $2.4V \leq VREF+ \leq VDDA$
VDDA <sup>(1)</sup>	Input, Analog Power	VDD-equivalent analog supply and: $2.4V \leq VDDA \leq VDD(3.6V)$
VREF-	Input, Analog Reference Negative	Low/negative reference voltage used by the ADC, $VREF- = VSSA$
VSSA <sup>(1)</sup>	Input, Analog Power Ground	Analog power ground equivalent to VSS
ADCx_IN[15:0]	Analog Input Signal	16 analog input channels

(1). VDDA and VSSA should be connected to VDD and VSS respectively.

### 11.3.1 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.

Conversion starts when the ADC power-up is delayed for some time ( $t_{STAB}$ ) and the ADON bit is set again.

Conversion can be stopped by clearing the ADON bit and placing the ADC in power-down mode.

In this mode, the ADC draws almost no power (only a few  $\mu A$ ).

### 11.3.2 ADC Clock

The ADCCLK clock provided by the clock controller is synchronized with PCLK2 (APB2 clock). The RCC controller provides a dedicated programmable prescaler for the ADC clock, as described in Section6 -Reset and Clock Control (RCC).

### 11.3.3 Channel Selection

There are 16 multiplexed channels. Conversions can be organized into two groups: rule groups and injection groups. A series of conversions performed in any order on any number of channels constitutes a grouped conversion. For example, conversions 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 rule set consists of up to 16 conversions. 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 four conversions. The injection channels and their conversion order are selected in the ADC\_JSQR register. The total number of conversions in the injection group should be written to the L[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.

#### Temperature Sensor/VREFINT Internal Channel

The temperature sensor is connected to channel ADC1\_IN16 and the internal reference voltage VREFINT is connected to ADC1\_IN17. These two internal channels can be converted by injection or rule channel.

*Notes: The temperature sensor and VREFINT can only appear in the main ADC1.*

### 11.3.4 Single Conversion Mode

In single conversion mode, the ADC performs only one conversion. This mode can be initiated either by setting the ADON bit in the ADC\_CR2 register (for rule channels only) or by external triggering (for rule or injection channels), at which point the CONT bit is 0. Once the conversion for the selected channel is complete:

- If a rule channel is converted:
  - Conversion data is stored in the 16-bit ADC\_DR register
  - 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\_DRJ1 register
  - JEOC (end of injection conversion) flag is set

- If the JEOCIE bit is set, an interrupt is generated.
- Then the ADC stops.

### 11.3.5 Continuous Conversion Mode

In continuous conversion mode, another conversion is initiated as soon as the previous ADC conversion is completed. This mode can be initiated by an external trigger or by setting the ADON bit on the ADC\_CR2 register, when the CONT bit is 1.

After each conversion:

- If a rule channel is converted:
  - Conversion data is stored in the 16-bit ADC\_DR register
  - 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\_DRJ1 register
  - JEOP (end of injection conversion) flag is set
  - If the JEOCIE bit is set, an interrupt is generated.

### 11.3.6 Timing Diagram

As shown in the figure below, the ADC requires a stabilization time  $t_{STAB}$  before starting the exact conversion. After starting the ADC conversion and 14 clock cycles, the EOC flag is set and the 16-bit ADC data register contains the result of the conversion.

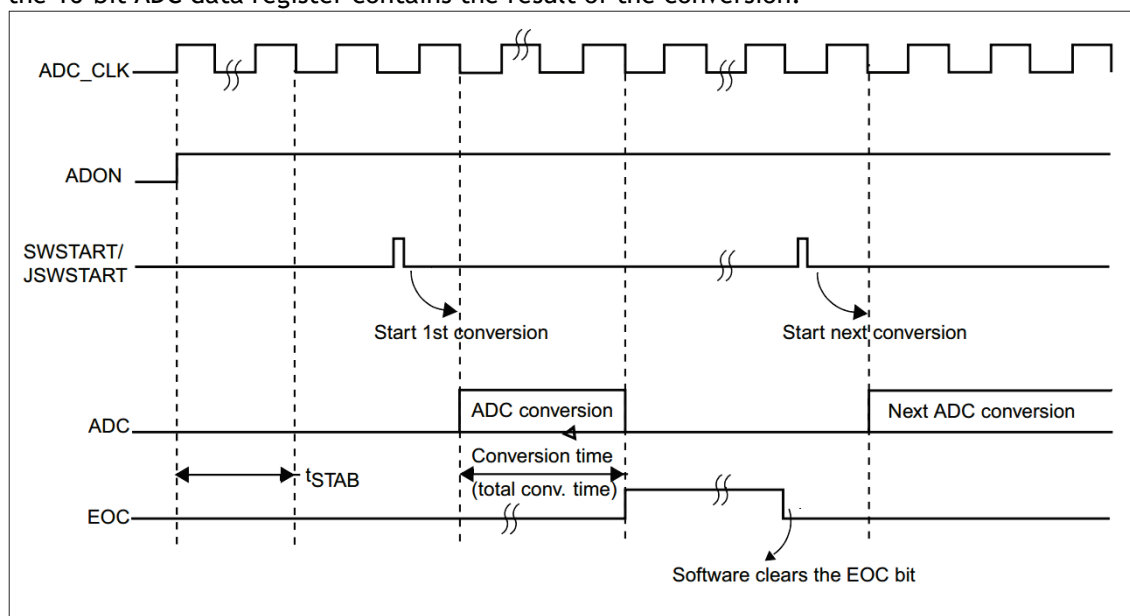


Figure23 Timing diagram

### 11.3.7 Analog Watchdog

The AWD analog watchdog status bit is set if the analog voltage being converted by the ADC is below the low threshold or above the high threshold. The threshold value is located in the lowest 12 valid bits of the ADC\_HTR and ADC\_LTR registers. The AWDIE bit of the ADC\_CR1 register is set to allow the corresponding interrupt to be generated.

The threshold is independent of the data alignment mode selected by the ALIGN bit on the ADC\_CR2 register. Comparison is done prior to alignment see section 11.5 ).

The analog watchdog can act on 1 or more channels by configuring the ADC\_CR1 register as shown in Table 61 .

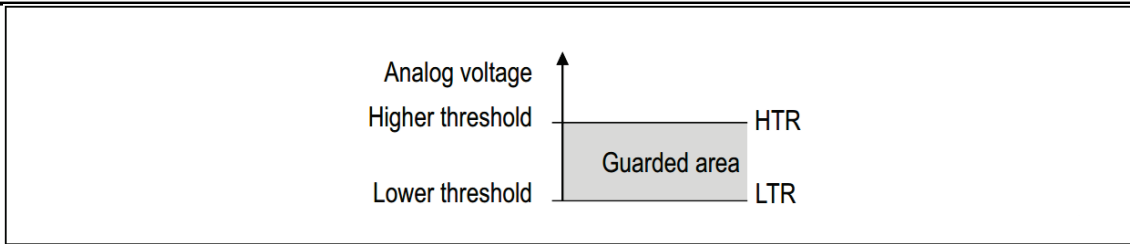


Figure24 Analog Watchdog Alert Area

Table61 Analog Watchdog Channel Selection

Simulating the passage of a watchdog alert	ADC_CR1 register control bits		
	AWDSGL bit	AWDEN bit	JAWDEN Bit
not have	arbitrary value	0	0
All injection channels	0	0	1
All rules channel	0	1	0
All injection and rule channels	0	1	1
Single <sup>(1)</sup> injection channel	1	0	1
Single <sup>(1)</sup> rule channel	1	1	0
Single <sup>(1)</sup> injection or rule channel	1	1	1

(1) Selected by AWDCH[4:0] bits

### 11.3.8 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). A single conversion is performed on each channel in each group. At the end of each conversion, 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 selected group, but continues again from the first channel of the selected group.

If the DMA bit is set, after each EOC, the DMA controller transfers the conversion data of the rule group channels into SRAM. And the injected channel conversion data is always stored in the ADC\_JDRx register.

### 11.3.9 Injection Channel Management

#### Trigger Injection

Clearing the JAUTO bit of the ADC\_CR1 register and setting the SCAN bit allows the trigger injection function to be used.

1. The conversion of a set of rule channels is initiated using an external trigger or by setting the ADON bit in the ADC\_CR2 register.
2. 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.
3. 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 be executed at the end of the injection sequence. Figure25 shows its timing diagram.

**Injection:** When using triggered injection conversions, it must be ensured that the interval between trigger events is longer than the injection sequence. For example, if the length of the sequence is 28 ADC clock cycles (i.e., 2 conversions with 1.5 clock intervals of sample time), the minimum interval between triggers must be 29 ADC clock cycles.

#### Auto-Injection

If the JAUTO bit is set, the injection group channels are automatically converted after the rule group channels. This can be used to convert up to 20 conversion sequences 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.

For ADC clock prescaler coefficients of 4 to 8, 1 ADC clock interval is automatically inserted when switching from a regular transition to an injected sequence or from an injected transition to a regular sequence; when the ADC clock prescaler coefficient is 2, there is a delay of 2 ADC clock intervals.

**Notes:** *It is not possible to use automatic injection and intermittent mode at the same time.*

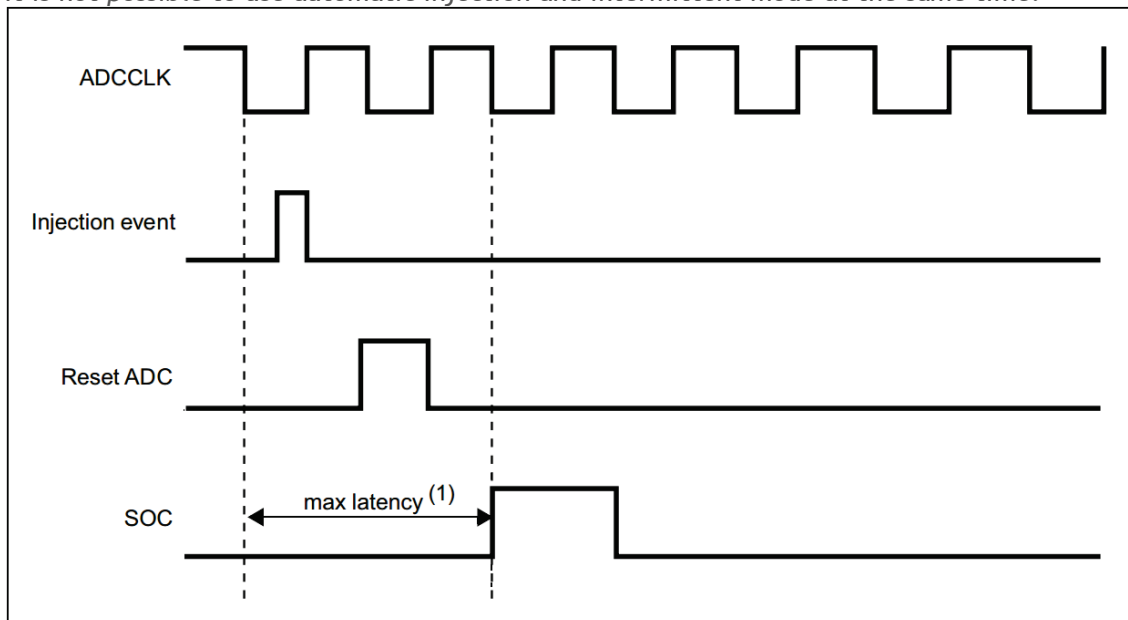


Figure25 Injection Conversion Delay

## 11.3.10 Intermittent Mode

### 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 the conversions in this sequence

The changeover is not completed until the changeover is complete. The total sequence length is defined by L[3:0] in the ADC\_SQR1 register.

Examples:

$n=3$ , channel being converted = 0, 1, 2, 3, 6, 7, 9, 10

First trigger: the sequence of conversion is 0, 1, 2

Second Trigger: Converted Sequence of 3, 6, 7

Third trigger: the converted sequences are 9 and 10 and an EOC event is generated

Fourth trigger: converted sequence 0, 1, 2

**Notes:** *When converting a rule group in interrupted mode, the conversion sequence does not automatically start from the beginning when it ends.*

*When all subgroups have been converted, the next trigger initiates the conversion of the first subgroup. In the example above, the fourth trigger reconverts channels 0, 1, and 2 of the first subgroup.*

### 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 in the sequence selected by the ADC\_JSQR register until all of the channels in the sequence are of the conversion until it is complete. The total sequence length is defined by the JL[1:0] bits of the ADC\_JSQR register.



**Examples:**

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

*Notes: 1 When all injection channel conversions are completed, the next trigger initiates the conversion of the 1st injection channel. In the above example, the fourth trigger reconverts the 1st injection channel 1.*

*2 You cannot use automatic injection and intermittent mode at the same time.*

*3 It must be avoided to set interrupted mode for both rules and injection groups. Intermittent mode can only work on one set of transformations.*

## 11.4 Calibrations

The ADC has a built-in self-calibration mode. Calibration significantly reduces quasi-accuracy errors due to variations in the internal capacitor bank. During calibration, an error correction code (digital value) is calculated on each capacitor, and this code is used to eliminate errors generated on each capacitor in subsequent conversions.

Calibration is initiated by setting the CAL bit in the ADC\_CR2 register. Once calibration is complete, the CAL bit is reset by hardware and normal conversion can begin. It is recommended that an ADC calibration be performed at power-up. At the end of the calibration phase, the calibration code is stored in ADC\_DR.

*Notes: 1 It is recommended to perform a calibration after each power-up.*

*2 The ADC must be in the power-down state (ADON='0') for more than at least two ADC clock cycles before calibration is initiated.*

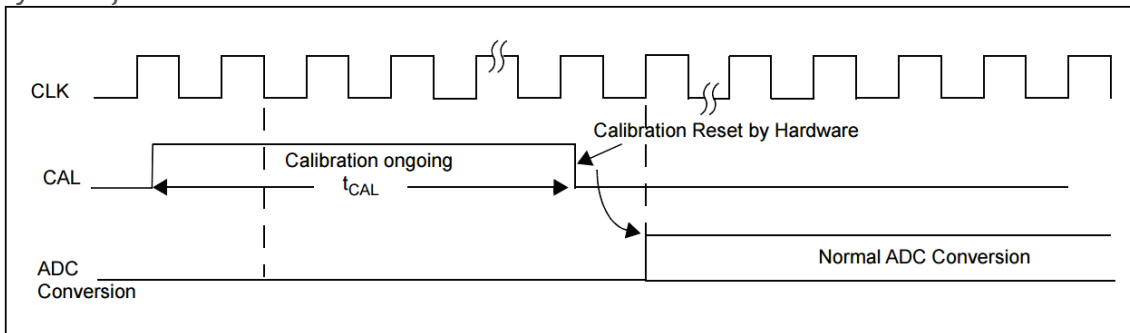


Figure26 Calibration Timing Diagram

## 11.5 Data Alignment

The ALIGN bit in the ADC\_CR2 register selects the alignment in which the converted data is stored. The data can be left- or right-aligned, such as

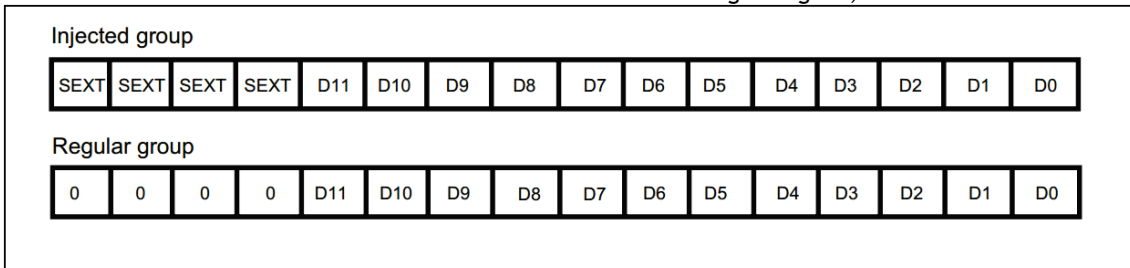


Figure27 and

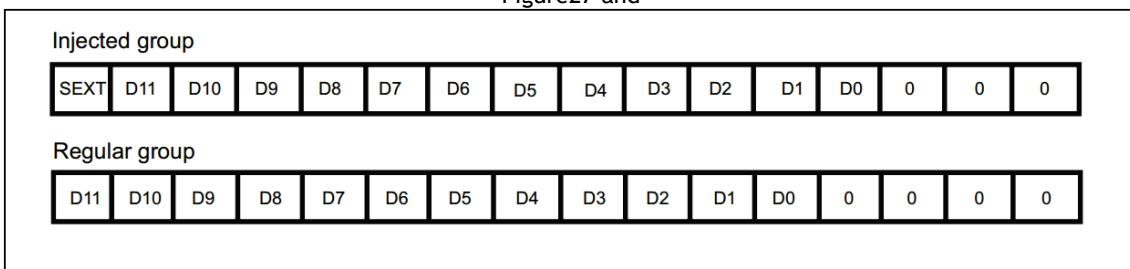


Figure28 shows. The data value injected into the group channel conversion has been subtracted from the offset defined in the ADC\_JOFRx register, so the result can be a negative value. the SEXT bit is the extended sign value.

For rule group channels, there is no need to subtract the offset value, so only 12 bits are valid.

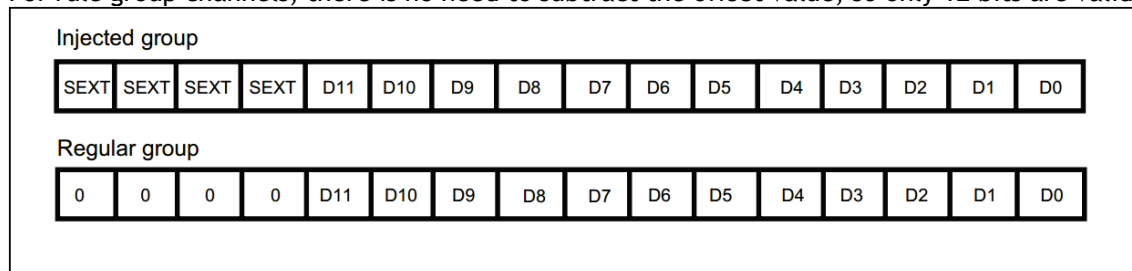


Figure27 Data right-aligned

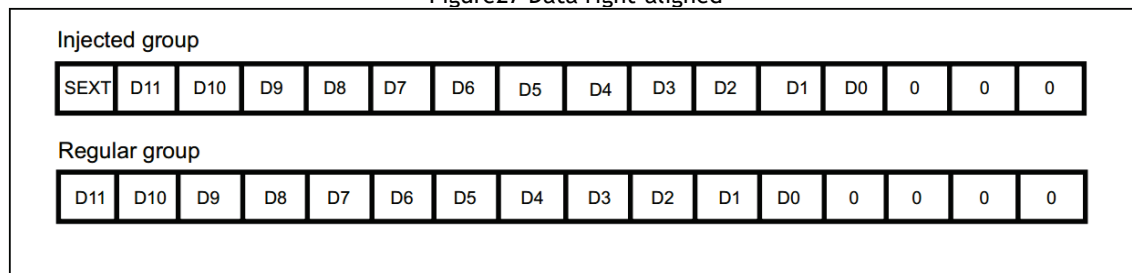


Figure28 Data Left Alignment

## 11.6 Programmable Channel 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 and ADC\_SMPR2 registers. Each channel can be sampled separately at different times.

The total conversion time is calculated as follows:

$$TCONV = \text{sampling time} + 12.5 \text{ cycles}$$

Example:

When ADCCLK = 14MHz, the sampling time is 1.5 cycles

$$TCONV = 1.5 + 12.5 = 14 \text{ cycles} = 1 \mu\text{s}$$

## 11.7 Externally Triggered Conversion

Conversions can be triggered by external events (e.g. timer capture, EXTI line). External events are able to trigger conversions if the EXTTRIG control bits are set. The EXTSEL[2:0] and JEXTSEL[2:0] control bits allow the application program to select one of eight possible events that can trigger sampling of rules and injection groups.

**Notes:** When an external trigger signal is selected for ADC rule or injection conversion, only its rising edge can initiate the conversion.

Table62 External triggering of ADC1 and ADC2 for rule channels

trigger source	typology	EXTSEL [2:0]
TIM1_CC1 event	Internal signal from on-chip timer	000
TIM1_CC2 event		001
TIM1_CC3 event		010
TIM2_CC2 event		011
TIM3_TRGO event		100
TIM4_CC4 event		101
EXTI line 11/TIM8_TRGO event <sup>(1)</sup>	External pin/internal signal from on-chip timer	110
SWSTART	software control bits	111

(1). For the rule channel, selecting EXTI line 11 or TIM8\_TRGO as the external trigger event can be realized by setting the ADC1\_ETRGREG\_REMAP bit and ADC2\_ETRGREG\_REMAP bit of ADC1 and ADC2, respectively.

Table63 External triggering of ADC1 and ADC2 for injection channels

trigger source	Connection type	JEXTSE [2:0]
TIM1_TRGO event	Internal signal from on-chip timer	000
TIM1_CC4 event		001
TIM2_TRGO event		010
TIM2_CC1 event		011
TIM3_CC4 event		100
TIM4_TRGO event		101
EXTI line 15/TIM8_CC4 event <sup>(1)</sup>	External pins/internal signals from on-chip timer	110
JSWSTART	software control bits	111

(1). Selection of the external trigger EXTIline15 or TIM8-CC4 event for the injection channel is accomplished by configuring bits ADC1ETRGINJREMAP and ADC2 ETRGINJ-REMAP for ADC1 and ADC2, respectively.

Table64 ADC3 External Trigger for Rule Channel

trigger source	Connection type	EXTSEL [2:0]
TIM3_CC1 event	Internal signal from on-chip timer	000
TIM2_CC3 event		001
TIM1_CC3 event		010
TIM8_CC1 event		011
TIM8_TRGO event		100
TIM5_CC1 event		101
TIM5_CC3 event		110
SWSTART	software control bits	111

Table65 External triggering of ADC3 for injection channels

trigger source	Connection type	JEXTSE [2:0]
TIM1_TRGO event	Internal signal from on-chip timer a	000
TIM1_CC4 event		001
TIM4_CC3 event		010
TIM8_CC2 event		011
TIM8_CC4 event		100
TIM5_TRGO event		101
TIM5_CC4 event		110
JSWSTART	software control bits	111

Software trigger events can be generated by a SWSTART or JSWSTART position '1' to register ADC\_CR2. Rule group transitions can be interrupted by injection triggers.

## 11.8 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.

*Notes: Only ADC1 and ADC3 have the DMA function. Data converted by ADC2 can be transferred through the Dual ADC mode using the DMA function of ADC1.*

## 11.9 Dual ADC Mode

In products with 2 or more ADC modules, the Dual ADC mode can be used (seeFigure29 Dual ADC Block Diagram).

In Dual ADC mode, the conversion can be initiated by either alternating or synchronous triggering of the ADC1 master and ADC2 slave, depending on the mode selected by the DUALMOD[2:0] bits in the ADC1\_CR1 register.

---

*Notes: In Dual ADC mode, when the conversion is configured to be triggered by an external event, the user must set it to trigger the master ADC only and the slave ADC to software trigger, which prevents accidental triggering of the slave conversion. However, external triggering must be activated for both master and slave ADCs.*

There are six possible models:

- synchronous injection model
- synchronous rule model
- Rapid crossover mode
- Slow crossover mode
- Alternate Trigger Mode
- stand-alone mode

It is also possible to use the above patterns in combination in the following ways:

- Synchronized Injection Patterns+ Synchronized Rule Patterns
- Synchronized Rule Mode+ Alternate Trigger Mode
- Synchronized Injection Mode+ Cross Mode

*Notes: In dual ADC mode, the DMA bit must be enabled in order to read slave conversion data on the master data register, even if DMA is not used to transfer rule channel data.*

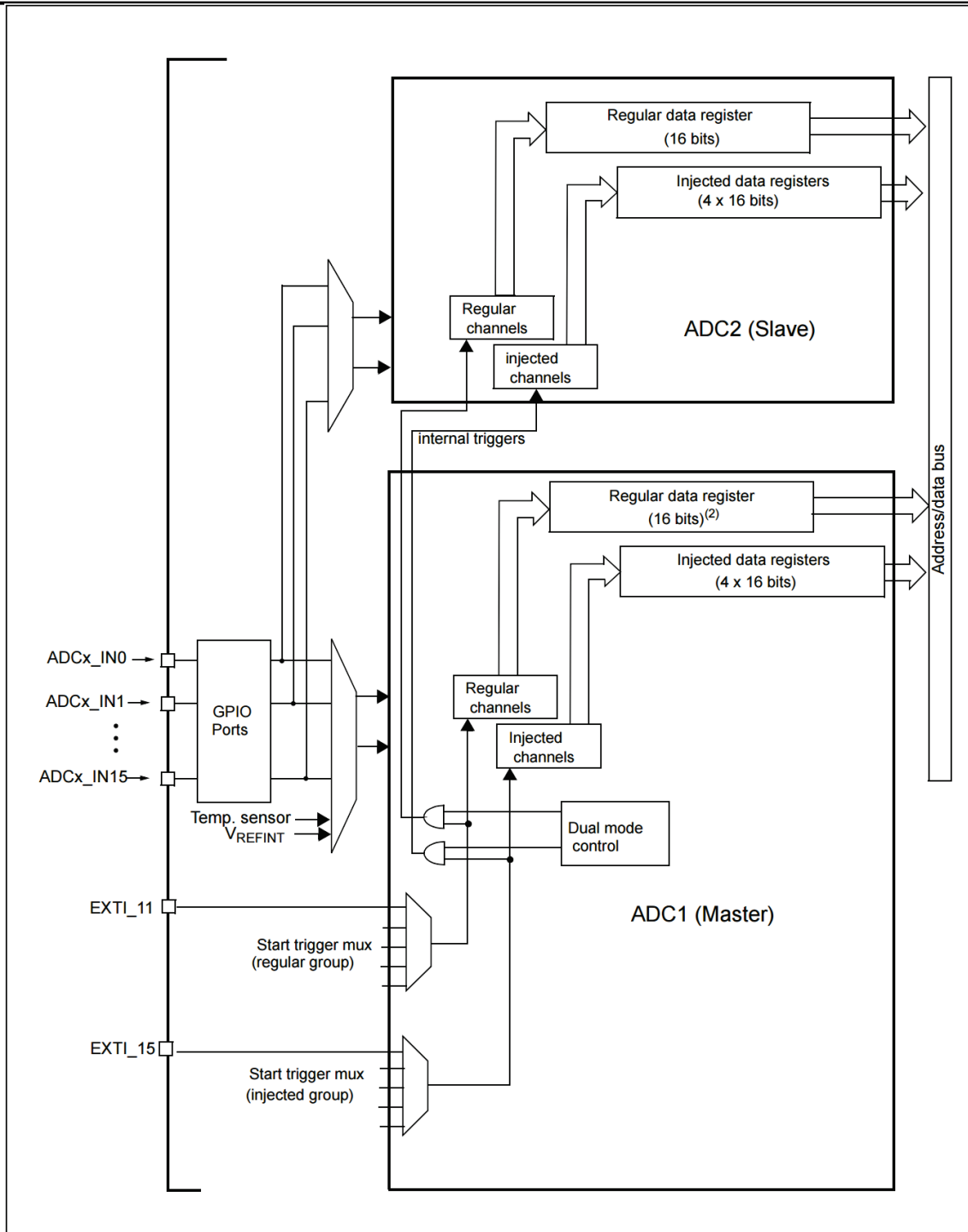


Figure 29 Dual ADC Block Diagram <sup>(1)</sup>

(1). An external trigger signal acts on ADC2, but is not shown in this figure.

(2). In some dual ADC modes, the rule-converted data for ADC1 and ADC2 are contained in the full ADC1 data register (ADC1\_DR).

### 11.9.1 Synchronous Injection Model

This mode converts an injection channel group. The external trigger comes from the injection group multiswitch of ADC1 (selected by JEXTSEL[2:0] in the ADC1\_CR2 register), which also provides a synchronization trigger to ADC2.

*Notes:* Do not convert the same channel on 2 ADCs (the sampling times of two ADCs on the same channel cannot overlap).

At the end of the conversion of ADC1 or ADC2:

- The converted data is stored in the ADC\_JDRx register of each ADC interface.

- The JEOP interrupt is generated when all ADC1/ADC2 injection channels are converted (if either ADC interface is open for interrupts).

**Notes:** In synchronized mode, sequences with the same time length must be converted or the trigger interval must be guaranteed to be longer than the longer of the 2 sequences, otherwise the ADC conversion with the shorter sequence may be restarted while the conversion of the longer sequence is still pending.

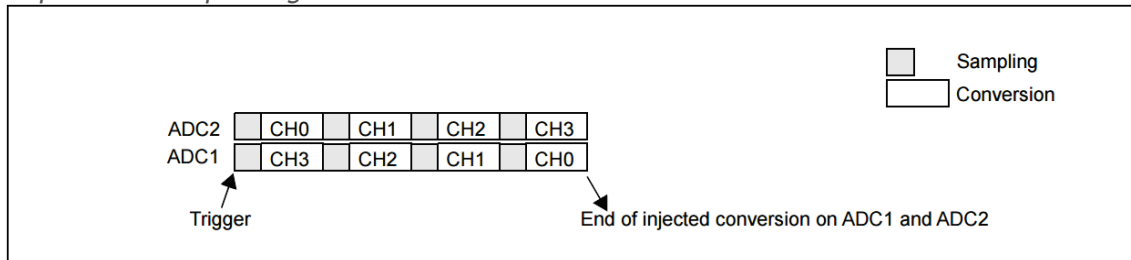


Figure30 Synchronized injection pattern on 4 channels

## 11.9.2 Synchronous Rule Model

This mode is executed on the rule channel group. The external trigger comes from the ADC1's rule group multiswitch (selected by EXTSEL[2:0] in the ADC1\_CR2 register), which also provides synchronization triggering to the ADC2.

**Notes:** Do not convert the same channel on 2 ADCs ((the sampling times of two ADCs on the same channel cannot overlap).

At the end of the conversion of ADC1 or ADC2:

- A 32-bit DMA transfer request is generated (if the DMA bit is set), and the contents of the 32-bit ADC1\_DR register are transferred into SRAM, which contains the ADC2 conversion data in the upper half word and the ADC1 conversion data in the lower half word.
- When all ADC1/ADC2 rule channels have been converted, an EOC interrupt is generated (if either ADC interface is open for interrupts).

**Notes:** In Synchronization Rule mode, sequences with the same time length must be converted or the trigger interval must be guaranteed to be longer than the longer of the 2 sequences, otherwise ADC conversions with shorter sequences may be restarted while the conversion of the longer sequence is still pending.

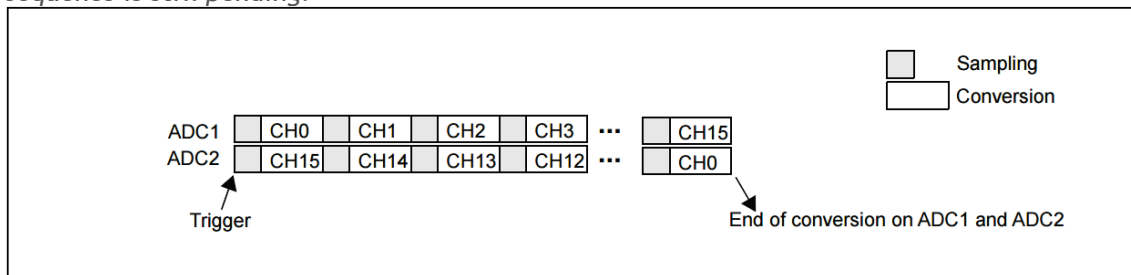


Figure31 Synchronization rule pattern on 16 channels

## 11.9.3 Rapid Crossover Mode

This mode applies only to groups of regular channels (usually one channel). The external trigger comes from the ADC1's regular channel multiswitch. After the external trigger is generated:

- ADC2 starts immediately and
- ADC1 starts after a delay of 7 ADC clock cycles

If the CONT bits of ADC1 and ADC2 are set at the same time, the two selected ADC rule channels will be converted consecutively.

After ADC1 generates an EOC interrupt (enabled by EOCIE), a 32-bit DMA transfer request is generated (if the DMA bit is set), and the 32-bit data in the ADC1\_DR register is transferred to the SRAM, with the upper half-word of the ADC1\_DR containing the converted data from ADC2 and the lower half-word containing the converted data from ADC1.

**Notes:** The maximum allowable sampling time is < 7 ADCCLK cycles to avoid overlapping of two sampling cycles when ADC1 and ADC2 convert the same channel.

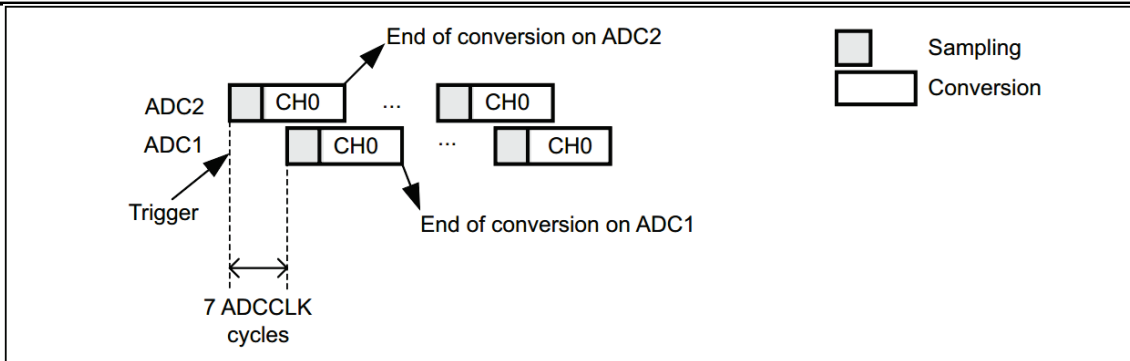


Figure.32 Fast crossover mode in continuous conversion mode on 1 channel

## 11.9.4 Slow Crossover Mode

This mode is only available for rule channel groups (one channel only). The external trigger comes from the ADC1's regular channel multiswitch. After the external trigger is generated:

- ADC2 starts immediately and
- ADC1 starts after a delay of 14 ADC clock cycles
- After delaying the second 14 ADC cycles ADC2 starts again and so on.

**Notes:** The maximum allowable sampling time is <14 ADCCLK cycles to avoid overlap with the next conversion.

After ADC1 generates an EOC interrupt (enabled by EOCIE), a 32-bit DMA transfer request is generated (if the DMA bit is set), and the 32-bit data in the ADC1\_DR register is transferred to the SRAM, with the upper half-word of the ADC1\_DR containing the converted data from ADC2 and the lower half-word containing the converted data from ADC1.

A new ADC2 conversion is automatically initiated after 28 ADC clock cycles.

The CONT bit cannot be set in this mode because it will continuously convert the selected rule channel.

**Notes:** The application must ensure that there can be no external triggers generated by the injection channel when using crossover mode.

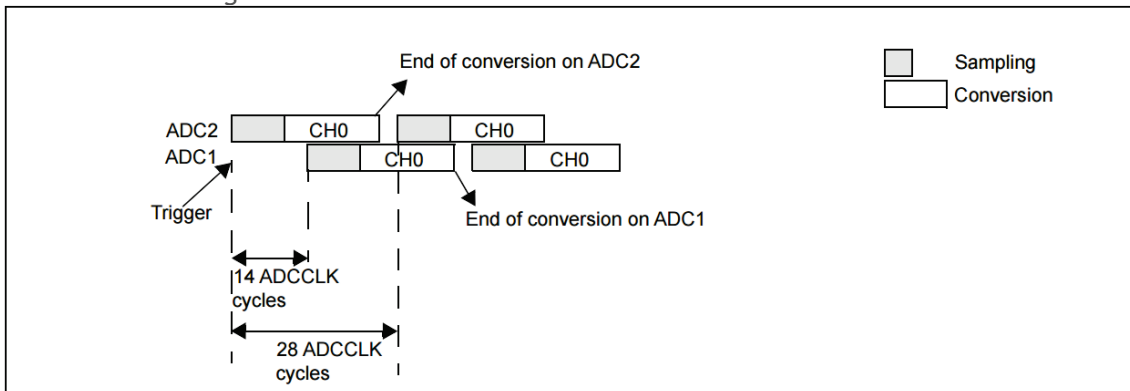


Figure33 Slow crossover mode on 1 channel

## 11.9.5 Alternate Trigger Mode

This mode is only available for injection channel groups. The external trigger source comes from the injection channel multiswitch of the ADC1.

- When the first trigger is generated, all injection group channels on ADC1 are converted.
- When the second trigger arrives, all injection group channels on ADC2 are converted.
- So much for the cycle .....

If JEOC interrupts are allowed to be generated, a JEOC interrupt is generated after all ADC1 injection group channel conversions. If JEOC interrupt generation is allowed, a JEOC interrupt is generated after all ADC2 injection group channel conversions.

When all injection group channels have been converted, if there is another external trigger, alternate trigger processing restarts from converting the ADC1 injection group channel.

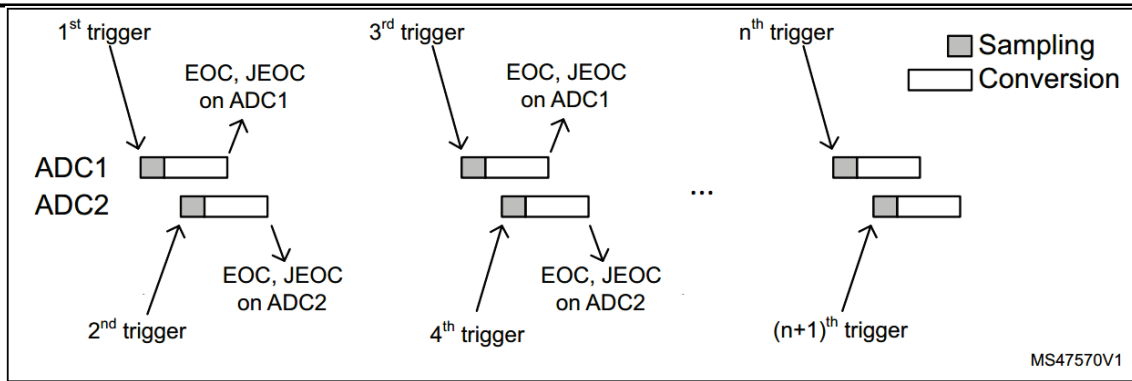


Figure34 Alternate triggering: injected channel groups per ADC1

If the injection intermittent mode is used on both ADC1 and ADC2:

- When the first trigger is generated, the first injected channel on ADC1 is converted.
- When the second trigger arrives, the first injection channel on the ADC2 is converted.
- So much for the cycle .....

If JEOC interrupts are allowed to be generated, a JEOC interrupt is generated after all ADC1 injection group channel conversions. If JEOC interrupt generation is allowed, a JEOC interrupt is generated after all ADC2 injection group channel conversions.

When all injection group channels have been converted, if there is another external trigger, the alternate trigger process starts again.

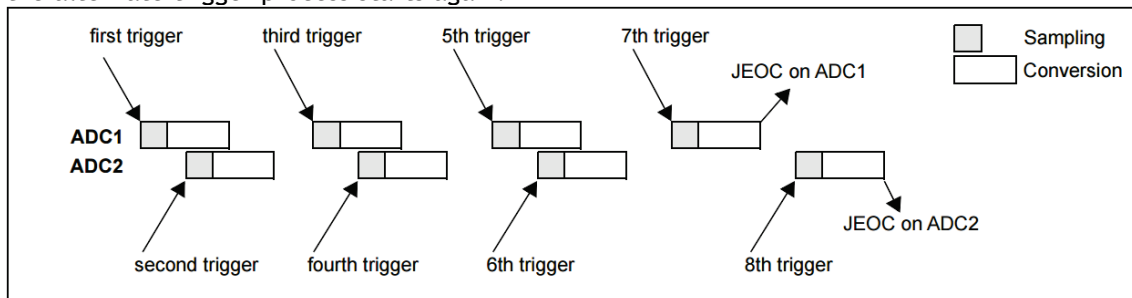


Figure35 Alternate triggering: 4 injection channels on each ADC in intermittent mode

## 11.9.6 Stand-Alone Mode

In this mode, the dual ADCs are not synchronized and each ADC interface works independently.

## 11.9.7 Mixed Rule/Injection Synchronization Patterns

Rule group synchronization transitions can be interrupted to initiate synchronization transitions for injected groups.

*Injection:* In mixed rule/injection synchronization mode, sequences with the same time length must be converted or the trigger interval must be guaranteed to be longer than the longer of the 2 sequences, otherwise ADC conversions with shorter sequences may be restarted while the conversion of the longer sequence is still pending.

## 11.9.8 Hybrid Synchronization Rules+ Alternate Trigger Modes

A Rule Group Synchronization Transition can be interrupted to initiate an Injection Group Alternate Trigger Transition. Figure36 shows a rule synchronization transition interrupted by an alternate trigger.

Injection alternate transitions are initiated as soon as the injection event arrives. If rule transitions are already running, to ensure synchronization after the injection transition, rule transitions for all ADCs (master and slave) are stopped and synchronization resumes at the end of the injection transition.

*Notes:* In mixed synchronization rules+ alternate trigger mode, sequences with the same time length must be converted or the trigger interval must be guaranteed to be longer than the longer of the 2 sequences, otherwise ADC conversions with shorter sequences may be restarted while the conversion of the longer sequence is still pending.



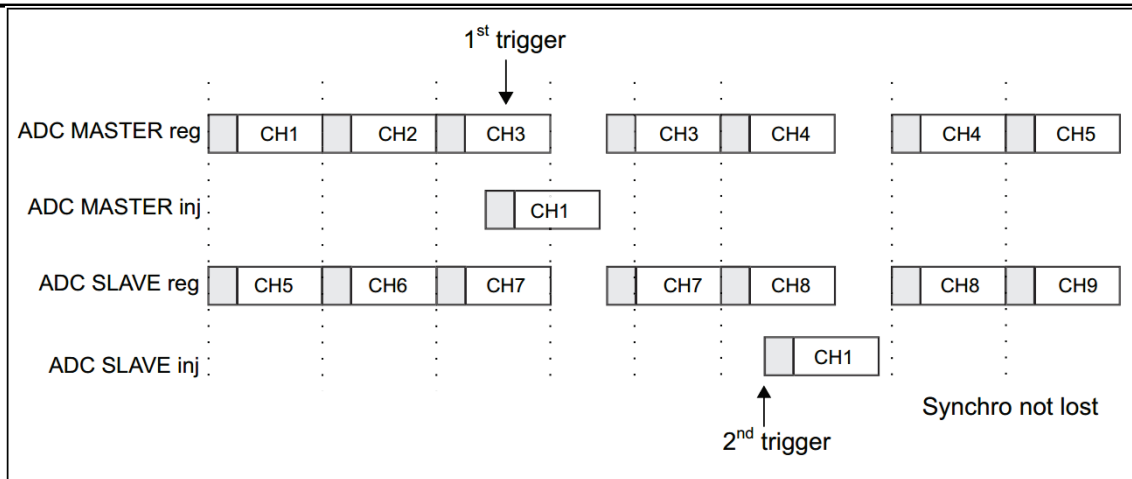


Figure36 Alternate+ Rule Synchronization

If the trigger event occurs during an injection transformation that interrupts a rule transformation, this trigger event is ignored. The following figure illustrates the operation in this case (the 2nd trigger is ignored).

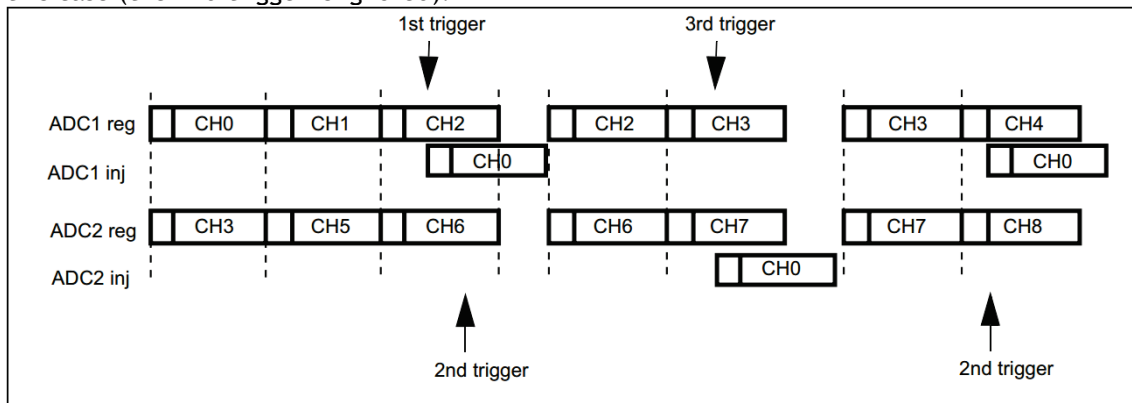


Figure37 Trigger event occurs during injection transition

## 11.9.9 Hybrid Synchronized Injection+ Cross Mode

An injection event can interrupt a cross-conversion. In this case, the cross-conversion is interrupted, the injection conversion is initiated, and the cross-conversion is resumed at the end of the injection sequence conversion. The following figure shows an example of this situation.

**Notes:** When the ADC clock prescaler factor is set to 4, the crossover mode recovery does not evenly distribute the sampling time, and the sampling interval is 8 ADC clock cycles alternating with 6 ADC clock cycles instead of an even 7 ADC clock cycles.

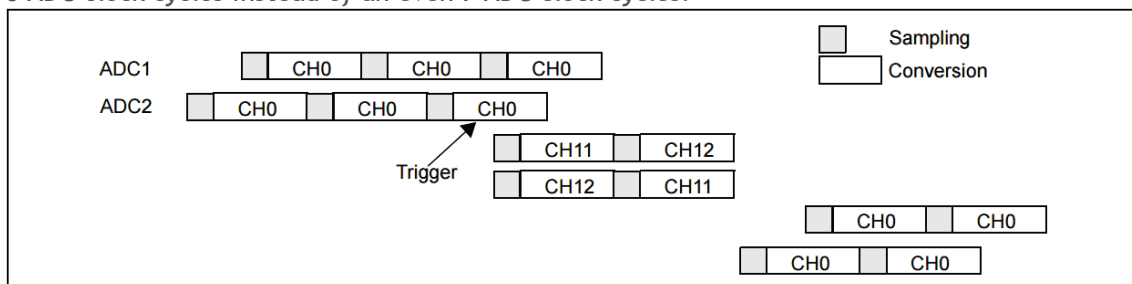


Figure38 Crossed single-channel conversions are injected into the sequence CH11 and CH12 interrupts

## 11.10 Temperature Sensor

A temperature sensor can be used to measure the temperature (TA) around the device. The temperature sensor is internally connected to the ADC1\_IN16 input channel, which converts the sensor output voltage into a digital value. The recommended sampling time for the analog input of the temperature sensor is 17.1  $\mu$ s.

Figure39 is a block diagram of the temperature sensor.

When not being used, the sensor can be placed in power down mode.

**Notes:** The TSVREFE bit must be set to activate the conversion of the internal channels: the ADC1\_IN16 (temperature sensor) and ADC1\_IN17 (VREFINT).

The output voltage of the temperature sensor varies linearly with temperature. Due to variations in the production process, the offset of the temperature change curve can vary from chip to chip (up to 45° C difference).

Internal temperature sensors are better suited for detecting changes in temperature rather than measuring absolute temperatures. If an accurate temperature measurement is required, an external temperature sensor should be used.

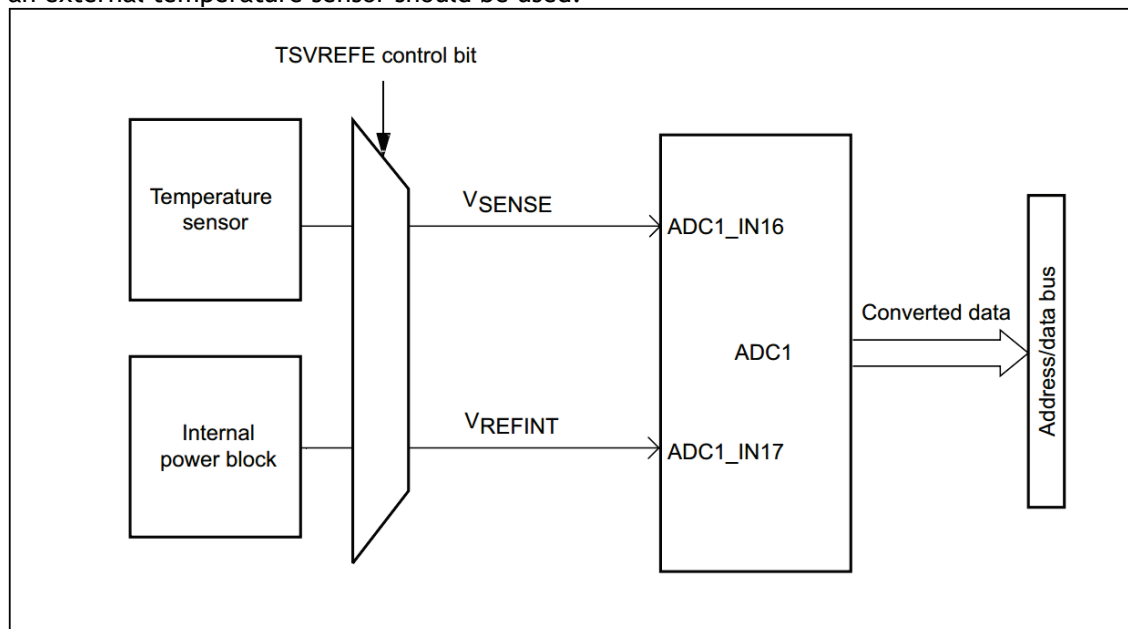


Figure39 Temperature Sensor and VREFINT Channel Block Diagram

### Temperature Reading

For the use of sensors:

1. Select ADC1\_IN16 input channel
2. Select sampling time of 17.1μs
3. Set the TSVREFE bit of ADC Control Register 2 (ADC\_CR2) to wake up the temperature sensor in power-down mode
4. ADC conversion is initiated by setting the ADON bit (or with an external trigger)
5. Result of reading VSENSE data on ADC data registers
6. Use the following equation to derive the temperature

$$\text{Temperature (}^{\circ}\text{C)} = \{(V25 - V\text{SENSE}) / \text{Avg\_Slope}\} + 25$$

Here:

$$V25 = V\text{SENSE at } 25^{\circ}\text{C}$$

Avg\_Slope = average slope of the temperature vs. VSENSE curve (in mV/° C or μV/° C)

Refer to the Electrical Characteristics section of the datasheet for actual values of V25 and Avg\_Slope.

**Notes:** There is a build-up time after the sensor wakes up from power-down mode until it can output the correct level of VSENSE. The ADC also has a build-up time after power-up, so to shorten the delay, the ADON and TSVREFE bits should be set at the same time.

## 11.11 ADC Interrupt

Interrupts can be generated at the end of rule and injection group transitions, and also when the analog watchdog status bit is set. They all have separate interrupt enable bits.

**Notes:** ADC1 and ADC2 interrupts are mapped on the same interrupt vector, while ADC3 interrupts have their own interrupt vector.

There are 2 other flags in the ADC\_SR register, but they have no interrupts associated with them:

- JSTRT (injection group channel transformation initiation)
- STRT (initiation of rule group channel transition)

Table66 ADC Interrupts

disruption event	event marker	Enable Control Bit
End of rule group conversion	EOC	EOCIE
End of injection group conversion	JEOC	JEOCIE
Analog watchdog status bits set	AWD	AWDIE

## 11.12 ADC Registers

Refer to Section1 for some of the abbreviations used in register descriptions.

These peripheral registers must be operated in word (32-bit) format.

### 11.12.1 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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											STRT	JSTRT	JEOC	EOC	AWD
											rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

Bit	notation	clarification
31:15	Reserved	Reserved, always reads 0.
4	STRT	<b>STRT</b> : Regular channel Start flag This bit is set by hardware at the start of a rule channel transition and cleared by software. 0: Rule channel conversion not started; 1: Rule channel conversion has begun.
3	JSTRT	<b>JSTRT</b> : Injected channel Start flag This bit is set by hardware at the start of the injection channel group conversion and cleared by software. 0: Injection channel group conversion has not started; 1: Injection channel group conversion has begun.
2	JEOC	<b>JEOC</b> : inject channel end of conversion bit (Injected channel end of conversion) This bit is set by hardware at the end of all injected channel group transitions and cleared by software 0: Conversion not completed; 1: Conversion complete.
1	EOC	<b>EOC</b> : End of conversion (End of conversion) 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 complete.
0	AWD	<b>AWD</b> : Analog watchdog flag bit (Analog watchdog flag) This bit is set by hardware when the converted voltage value is outside the range defined by the ADC_LTR and ADC_HTR registers, and is cleared by software 0: No analog watchdog event occurred; 1: An analog watchdog event occurs.

### 11.12.2 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
Reserved								AWDEN	JAWDEN	Reserved		DUALMOD[3:0]			
								rw	rw			rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DISCNUM [2:0]			JDISCEN	DISCEN	JAUTO	AWDSGL	SCAN	JEOCIE	AWDIE	EOCIE	AWDCH[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	notation	clarification
31:24	Reserved	Reserved, always reads 0.
23	AWDEN	<b>AWDEN</b> : Analog watchdog enable on regular channels This bit is set and cleared by software.

		0: Disable analog watchdog on the rule channel; 1: Use an analog watchdog on the rule channel.
22	JAWDEN	<b>JAWDEN:</b> Enable analog watchdog on injected channels (Analog watchdog enable on injected channels) This bit is set and cleared by software. 0: Disables the analog watchdog on the injection channel; 1: Use an analog watchdog on the injection channel.
21:20	Reserved	Reserved. Must be kept at 0.
19:16	DUALMOD [3:0]	<b>DUALMOD[3:0]:</b> Dual mode selection The software uses these bits to select the mode of operation. 0000: Stand-alone mode 0001: Mixed Synchronization Rules+ Injection Synchronization Mode 0010: Mixed Synchronization Rules+ Alternate Trigger Modes 0011: Hybrid Synchronized Injection+ Fast Cross Mode 0100: Mixed Synchronized Injection+ Slow Cross Mode 0101: Injection Synchronized Mode 0110: Rule Synchronization Mode 0111: Rapid crossover mode 1000: Slow cross mode 1001: Alternate trigger mode Note: These bits are reserved in ADC2 and ADC3. In Dual Mode, changing the configuration of a channel creates a restart condition, which will result in a loss of synchronization. It is recommended that dual mode be turned off before making any configuration changes.
15:13	DISCNUM [2:0]	<b>DISCNUM[2:0]:</b> Discontinuous mode channel count (Discontinuousmodechannelcount) The software defines the number of channels that will be converted to regular channels after receiving external triggers in the discontinuous mode through these bits. 000: 1 channel 001: 2 channels ..... 111: 8 channels
12	JDISCEN	<b>JDISCEN:</b> Discontinuous mode on injected channels This bit is set and cleared by software to enable or disable intermittent mode on the injector channel set 0: Intermittent mode is disabled on the injection channel group; 1: Use intermittent mode on the injection channel group.
11	DISCEN	<b>DISCEN:</b> Discontinuous mode on regular channels (Discontinuous mode on regular channels) This bit is set and cleared by software to enable or disable intermittent mode on a rule channel group 0: Intermittent mode is disabled on the rule channel group; 1: Use intermittent mode on rule channel groups.
10	JAUTO	<b>JAUTO:</b> Automatic Injected Group conversion This bit is set and cleared by software to turn on or off the automatic injection of channel group transitions at the end of a rule channel group transition 0: Turns off automatic injection channel group switching; 1: Enable automatic injection channel group switching.
9	AWDSGL	<b>AWDSGL:</b> scan mode in a single channel on the use of watchdog (Enable the watchdog on a single channel in scan mode) This bit is set and cleared by software to enable or disable the analog watchdog function on the channel specified by the AWDCH[4:0] bits 0: Use analog watchdog on all channels; 1: Use an analog watchdog on a single channel.
8	SCAN	<b>SCAN:</b> Scan mode This bit is set and cleared by software to turn scan mode on or off. In scan mode, converts the channel selected by the ADC_SQRx or ADC_JSQRx registers. 0: Turns off scanning mode; 1: Use scan mode. Note: If the EOCIE or JEOCIE bits are set separately, an EOC or JEOC interrupt is generated only after the last channel conversion is complete.
7	JEOCIE	<b>JEOCIE:</b> Allow generation of end-of-conversion interrupts for injected channels (Interrupt enable for injected channels) This bit is set and cleared by software to prohibit or allow generation of interrupts at the end of conversion for all injected channels. 0: JEOC interrupt is disabled; 1: Allow JEOC interrupt. Interrupts are generated when the JEOC bit is set by hardware.
6	AWDIE	<b>AWDIE:</b> allow generation of analog watchdog interrupt (Analog watchdog interrupt enable) This bit is set and cleared by software to disable or allow the analog watchdog to generate interrupts. In scan mode, if the watchdog detects an out-of-range value, the scan will only abort if this bit is set. 0: Disable analog watchdog interrupt; 1: Allow analog watchdog interrupt.

5	EOCIE	EOCIE: Allow to generate EOC interrupt (Interrupt enable for EOC) This bit is set and cleared by software to disable or allow interrupts to be generated at the end of a conversion. 0: Disable EOC interrupt; 1: Allow EOC interrupt. Interrupts are generated when the EOC bit is set by hardware.
4:0	AWDCH[4:0]	AWDCH[4:0]: Analog watchdog channel select bits (Analog watchdog channel select bits) These bits are set and cleared by software to select the input channel for analog watchdog protection. 00000: ADC analog input channel 000001: ADC analog input channel 1 ..... 01111: ADC analog input channel 15 10000: ADC analog input channel 16 10001: ADC analog input channel 17 retains all other values. Note: The analog input channels 16 and 17 of the ADC1 are connected to the temperature sensor and VREFINT, respectively, inside the chip. The analog input channels 16 and 17 of ADC2 are connected to VSS inside the chip. the ADC3 analog input channels 9, 14, 15, 16, and 17 are connected to Vss.

### 11.12.3 ADC Control Register 2 (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
Reserved								TSVREFE	SWSTART	JSWSTART	EXTTRIG	EXTSEL [2:0]		Reserved	
								rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JEXTTRIG	JEXTSEL [2:0]			ALIGN	Reserved		DMA	Reserved				RSTCAL	CAL	CONT	ADON
rw	rw	rw	rw	rw			rw					rw	rw	rw	rw

Bit	notation	clarification
31:24	Reserved	Reserved, always reads 0.
23	TSVREFE	TSVREFE: Temperature sensor and VREFINT enable (Temperature sensor and VREFINT enable) This bit is set and cleared by software to enable or disable the temperature sensor and VREFINT channels. In devices with more than 1 ADC, this bit appears only in ADC1. 0: Disable the temperature sensor and VREFINT; 1: Enable the temperature sensor and VREFINT.
22	SWSTART	SWSTART: Start conversion of regular channels (Start conversion of regular channels) This bit is set by software to initiate the conversion and cleared by hardware immediately after the conversion has started. This bit is used to initiate a conversion for a set of rule channels if SWSTART is selected as the trigger event in the EXTSEL[2:0] bits, the 0: Reset state; 1: Start converting the rule channel.
21	JSWSTART	JSWSTART: Start conversion of injected channels (Start conversion of injected channels) Set by software to initiate a conversion, this bit can be cleared by software or cleared by hardware as soon as the conversion is initiated. This bit is used to initiate a conversion for a group of injection channels if JSWSTART is selected as the trigger event in the JEXTSEL[2:0] bits. 0: Reset state; 1: Start converting the injection channel.
20	EXTTRIG	EXTTRIG: External trigger conversion mode for regular channels This bit is set and cleared by software to enable or disable externally triggered events that can initiate rule channel group transitions. 0: No external event is used to initiate the conversion; 1: Use an external event to initiate the conversion.
19:17	EXTSEL [2:0]	EXTSEL[2:0]: select the external eventselectforregulargroup conversion. These bits select the external event used to initiate the rule channel group conversion The trigger configuration for ADC1 and ADC2 is as follows 000: CC1 event for timer 1 001: CC2 event of timer 1 010: CC3 event of timer 1 011: CC2 event of timer 2 The trigger configuration for ADC3 is as follows 000: CC1 event of timer 3 001: CC3 event of timer 2 100: TRGO event of timer 3 101: CC4 event of timer 4 110: EXTI line 11/TIM8_TRGO event 111: SWSTART 100: TRGO event of timer 8 101: CC1 event of timer 5 110: CC3 event of timer 5

		010: CC3 event of timer 1 011: CC1 event of timer 8 111: SWSTART
16	Reserved	Reserved, always reads 0.
15	JEXTTRIG	<b>JEXTTRIG:</b> External trigger conversion mode for injected channels This bit is set and cleared by software to enable or disable externally triggered events that can initiate injected channel group transitions. 0: No external event is used to initiate the conversion; 1: Use an external event to initiate the conversion.
14:12	JEXTSEL [2:0]	<b>JEXTSEL[2:0]:</b> select external event select for injected channel group transformation These bits select the external events used to initiate the injection of channel group transitions. The trigger configuration for ADC1 and ADC2 is as follows 000: TRGO event of timer 1 001: CC4 event of timer 1 010: TRGO event of timer 2 011: CC1 event of timer 2 100: CC4 event of timer 3 101: TRGO event for timer 4 110: EXTI line 15/TIM8_CC4 event 111: JSWSTART The trigger configuration for ADC3 is as follows 000: TRGO event of timer 1 001: CC4 event of timer 1 010: CC3 event for timer 4 011: CC2 event of timer 8 100: CC4 event of timer 8 101: TRGO event of timer 5 110: CC4 event of timer 5 111: JSWSTART
11	ALIGN	<b>ALIGN:</b> Data alignment This bit is set and cleared by software, refer to Figure 27 and Figure 28. 0: Right-aligned; 1: Left Alignment.
10:9	Reserved	Reserved. Must be kept at 0.
8	DMA	<b>DMA:</b> Direct memory access mode (Direct memory access mode) This bit is set and cleared by software. See the DMA Controller chapter for details. 0: DMA mode is not used; 1: Use DMA mode. Note: Only ADC1 and ADC3 can generate DMA requests.
7:4	Reserved	Reserved. Must be kept at 0.
3	RSTCAL	<b>RSTCAL:</b> Reset calibration This bit is set by software and cleared by hardware. This bit is cleared after the calibration register is initialized. 0: The calibration register is initialized; 1: Initialize the calibration registers. Note: If RSTCAL is set while a conversion is in progress, clearing the calibration register requires an additional cycle.
2	CAL	<b>CAL:</b> A/D Calibration This bit is set by software to start calibration and cleared by hardware at the end of calibration. 0: Calibration complete; 1: Start calibration.
1	CONT	<b>CONT:</b> Continuous conversion This bit is set and cleared by software. If this bit is set, conversion will continue until the bit is cleared. 0: Single conversion mode; 1: Continuous conversion mode.
0	ADON	<b>ADON:</b> turn on/off A/D converter (A/D converter ON/OFF) This bit is set and cleared by software. When this bit is '0', writing a '1' will wake up the ADC from power-down mode. When this bit is '1', writing '1' will initiate the conversion. The application program should note that there is a delay tSTAB between the power up of the converter and the start of the conversion, see Figure 23. 0: Turns off ADC conversion/calibration and enters power-down mode; 1: Turn on the ADC and start the conversion. Note: If there are other bits in this register that are changed along with ADON, the conversion is not triggered. This is to prevent triggering an incorrect conversion.

## 11.12.4 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
Reserved								SMP17[2:0]			SMP16[2:0]			SMP15[2:1]	
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
SMP15_0	SMP14[2:0]			SMP13[2:0]			SMP12[2:0]			SMP11[2:0]			SMP10[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	notation	clarification								
31:24	Reserved	Reserved, always reads 0.								
23:0	SMPx[2:0]	<p><b>SMPx[2:0]:</b> channel x sample time selection (Channel x Sample time selection) These bits are used to independently select the sample time for each channel. The channel selection bits must be held constant during the sampling cycle.</p> <table><tr><td>000: 1.5 cycles</td><td>100: 41.5 cycles</td></tr><tr><td>001: 7.5 cycles</td><td>101: 55.5 cycles</td></tr><tr><td>010: 13.5 cycles</td><td>110: 71.5 cycles</td></tr><tr><td>011: 28.5 cycles</td><td>111: 239.5 cycles</td></tr></table> <p>Note: The analog input channels 16 and 17 of the ADC1 are connected to the temperature sensor and VREFINT, respectively, inside the chip. The analog input channels 16 and 17 of ADC2 are connected to Vss inside the chip. ADC3 analog input channels 14, 15, 16, and 17 are connected to Vss</p>	000: 1.5 cycles	100: 41.5 cycles	001: 7.5 cycles	101: 55.5 cycles	010: 13.5 cycles	110: 71.5 cycles	011: 28.5 cycles	111: 239.5 cycles
000: 1.5 cycles	100: 41.5 cycles									
001: 7.5 cycles	101: 55.5 cycles									
010: 13.5 cycles	110: 71.5 cycles									
011: 28.5 cycles	111: 239.5 cycles									

## 11.12.5 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
Reserved				SMP9[2:0]			SMP8[2:0]			SMP7[2:0]			SMP6[2:0]		
rw				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMP5_0	SMP4[2:0]			SMP3[2:0]			SMP2[2:0]			SMP1[2:0]			SMP0[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	notation	clarification	
31:30	Reserved	Reserved, always reads 0.	
29:0	SMPx[2:0]	SMPx[2:0]: channel x sample time selection (Channel x Sample time selection) These bits are used to independently select the sample time for each channel. The channel selection bits must be held constant during the sampling cycle.	
		000: 1.5 cycles	100: 41.5 cycles
		001: 7.5 cycles	101: 55.5 cycles
		010: 13.5 cycles	110: 71.5 cycles
		011: 28.5 cycles	111: 239.5 cycles
		Note: ADC3 analog input channel 9 is connected to Vss	

## 11.12.6 ADC Injection Channel Data Offset Register x (ADC\_JOFRx) (x=1..4)

Address offset: 0x14-0x20

Reset value: 0x0000 0000

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

Bit	notation	clarification
31:12	Reserved	Reserved, always reads 0.

11:0	JOFFSETx [11:0]	JOFFSETx[11:0]: data offset for injected channel x (Data offset for injected channel x) These bits define the value to be used to subtract from the original converted data when converting the injected channel. The result of the conversion can be read out in the ADC_JDRx register.
------	--------------------	---

## 11.12.7 ADC Watchdog High Threshold Register (ADC\_HTR)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				HT[11:0]											
				rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW
Bit	notation	clarification													
31:12	Reserved	Reserved, always reads 0.													
11:0	HT[11:0]	HT[11:0]: Analog watchdog high threshold These bits define the analog watchdog's threshold high limit.													

**Notes:** The software can write to these registers while an ADC conversion is in progress. The programmed values will take effect when the next conversion is completed. Writes to this register have a write delay, which may create uncertainty as to when the new value is programmed to be effective.

## 11.12.8 ADC Watchdog Low Threshold Register (ADC\_LRT)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				LT[11:0]											
				rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW
Bit	notation	clarification													
31:12	Reserved	Reserved, always reads 0.													
11:0	LT[11:0]	LT[11:0]: analog watchdog low threshold These bits define the low limit of the analog watchdog threshold.													

**Notes:** The software can write to these registers while an ADC conversion is in progress. The programmed values will take effect when the next conversion is completed. Writing to this register has a write delay, which may create uncertainty about the effective time for programming the new value.

## 11.12.9 ADC Rule Sequence Register 1 (ADC\_SQR1)

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								L[3:0]				SQ16[4:1]			
								rW	rW	rW	rW	rW	rW	rW	rW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ16_0	SQ15 [4:0]					SQ14 [4:0]					SQ13 [4:0]				
rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW
Bit	notation	clarification													
31:24	Reserved	Reserved, always reads 0.													
23:20	L[3:0]	L[3:0]: Regular channel sequence length (Regular channel sequence length) These bits define by software the number of channels in a regular channel conversion sequence. 0000: 1 conversion 0001: 2 conversions													



		..... 1111: 16 conversions
19:15	SQ16[4:0]	SQ16[4:0]: 16th conversion in regular sequence (16th conversion in regular sequence) These bits are used by the software to define the numbering of the 16th conversion channel (0-17) in the conversion sequence.
14:10	SQ15 [4:0]	SQ15[4:0]: 15th conversion in regular sequence
9:5	SQ14 [4:0]	SQ14[4:0]: 14th conversion in regular sequence
4:0	SQ13 [4:0]	SQ13[4:0]: 13th conversion in regular sequence

## 11.12.10ADC Rule Sequence Register 2 (ADC\_SQR2)

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ10_0															

Bit	notation	clarification
31:24	Reserved	Reserved, always reads 0.
29:25	SQ12[4:0]	SQ12[4:0]: 12th conversion in regular sequence (12th conversion in regular sequence) These bits are used by the software to define the numbering of the 12th conversion channel (0-17) in the conversion sequence.
24:20	SQ11[4:0]	SQ11[4:0]: 11th conversion in regular sequence
19:15	SQ10[4:0]	SQ10[4:0]: 10th conversion in regular sequence
14:10	SQ9 [4:0]	SQ9[4:0]: 9th conversion in regular sequence
9:5	SQ8[4:0]	SQ8[4:0]: 8th conversion in regular sequence (82nd conversion in regular sequence)
4:0	SQ7 [4:0]	SQ7[4:0]: 7th conversion in regular sequence

## 11.12.11ADC Rule Sequence Register 3 (ADC\_SQR3)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ4_0															

Bit	notation	clarification
31:30	Reserved	Reserved, always reads 0.
29:25	SQ6 [4:0]	SQ6[4:0]: 6th conversion in regular sequence (6th conversion in regular sequence) These bits are used by the software to define the number of the 6th conversion channel (0-17) in the conversion sequence.
24:20	SQ5 [4:0]	SQ5[4:0]: 5th conversion in regular sequence
19:15	SQ4[4:0]	SQ4[4:0]: 4th conversion in regular sequence
14:10	SQ3[4:0]	SQ3[4:0]: 3rd conversion in regular sequence
9:5	SQ2[4:0]	SQ2[4:0]: the 2nd conversion in regular sequence (2nd conversion in regular sequence)
4:0	SQ1[4:0]	SQ1[4:0]: 1st conversion in regular sequence

## 11.12.12 ADC Injection Sequence Register (ADC\_JSQR)

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved										JL[3:0]		JSQ4[4:1]			
										rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JSQ4_0	JSQ3[4:0]					JSQ2[4:0]					JSQ1[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	notation	clarification
31:22	Reserved	Reserved, always reads 0.
21:20	JL[1:0]	JL[1:0]: Injected sequence length These bits define by software the number of channels in a regular channel conversion sequence. 00: 1 conversion 01: 2 conversions 10: 3 conversions 11: 4 conversions
19:15	JSQ4 [4:0]	JSQ4[4:0]: 4th conversion in injected sequence (4th conversion in injected sequence) These bits are defined by the software as the number of the 4th conversion channel (0-17) in the conversion sequence. Note: Unlike the regular conversion sequence, if the length of JL[1:0] is less than 4, the sequence order of conversion starts from (4-JL). For example, ADC_JSQR[21:0]=1000011000110011100010 means that the scan conversions will be converted in the following channel order: 7, 3, 3, not 2, 7, 3.
14:10	JSQ3[4:0]	JSQ3[4:0]: 3rd conversion in injected sequence
9:5	JSQ2[4:0]	JSQ2[4:0]: 2nd conversion in injected sequence
4:0	JSQ1[4:0]	JSQ1[4:0]: 1st conversion in injected sequence

## 11.12.13 ADC Injection Data Register x (ADC\_JDRx) (x=1..4)

Address offset: 0x3C-0x48

Reset value: 0x0000 0000

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

Bit	notation	clarification
31:16	Reserved	Reserved, always reads 0.
15:0	JDATA[15:0]	JDATA[15:0]: injected data (Injected data) These bits are read-only and contain the conversion results of the injected channel. The data is left- or right-aligned, as shown in Figure 27 and Figure 28

## 11.12.14 ADC Rule Data Register (ADC\_DR)

Address offset: 0x4C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADC2DATA[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
DATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit	notation	clarification
31:16	ADC2 DATA[15:0]	ADC2DATA[15:0]: ADC2 converted data (ADC2 data) -In ADC1: Dual mode, these bits contain the rule channel data converted by ADC2. See 11.9: Dual ADC mode

		-In ADC2 and ADC3: These bits are not used.
15:0	DATA[15:0]	DATA[15:0]: Regular converted data (Regular data) These bits are read-only and contain the conversion results for the rule channel. The data is left- or right-aligned, as shown in Figure 27 and Figure 28.

## 11.12.15 ADC Register Address Map

The following table lists all ADC registers.

Table 67 ADC Register Image and Reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
00h	ADC_SR	Reserved																												STRT	JSTRT	JEOC	EOC	AWD			
	Reset value																													0	0	0	0	0			
04h	ADC_CR1	Reserved								AWDEN	JAWDEN	Reserved	DUALMOD [3:0]			DISCNUM [2:0]			JDISEN	DISCEN	JAUTO	AWDSGL	SCAN	JEOCIE	AWDIE	EOCIE	AWDCH[4:0]										
	Reset value									0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
08h	ADC_CR2	Reserved								TSVREFE	SWSTART	JSWSTAR	EXTTRIG	EXTSEL [2:0]			Reserved	JEXTTRIG	JEXTSEL [2:0]			ALIGN	Reserved	DMA	Reserved			RSTCAL	CAL	CONT	ADON						
	Reset value									0	0	0	0	0	0	0		0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0			
0Ch	ADC_SMPR1	Sampling time bit SMPx_x																																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
10h	ADC_SMPR2	Sampling time bit SMPx_x																																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
14h	ADC_JOFR1	Reserved																JOFFSET1[11:0]																			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
18h	ADC_JOFR2	Reserved																JOFFSET2[11:0]																			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1Ch	ADC_JOFR3	Reserved																JOFFSET3[11:0]																			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
20h	ADC_JOFR4	Reserved																JOFFSET4[11:0]																			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1Ch	ADC_HTR	Reserved																HT[11:0]																			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
20h	ADC_LTR	Reserved																LT[11:0]																			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2Ch	ADC_SQR1	Reserved								L[3:0]		Rule channel sequence SQx_x bits																									
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
30h	ADC_SQR2	Reserved	Rule channel sequence SQx_x bits																																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
34h	ADC_SQR3	Reserved	Rule channel sequence SQx_x bits																																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
38h	ADC_JSQR	Reserved								JL [1:0]		Inject channel sequence JSQx_x bit																									
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
3Ch	ADC_JDR1	Reserved																JDATA[15:0]																			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
40h	ADC_JDR2	Reserved																JDATA[15:0]																			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
44h	ADC_JDR3	Reserved																JDATA[15:0]																			

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
48h	ADC_JDR4	Reserved																JDATA[15:0]																	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4Ch	ADC_DR	ADC2DATA[15:0]																Rule DATA[15:0]																	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				

See Table 1 for register start addresses.

## 12 Digital-to-Analog Conversion (DAC)

### 12.1 DAC Introduction

The Digital/Analog Converter Module (DAC) is a 12-bit digital input, voltage output digital/analog converter. The DAC can be configured in 8-bit or 12-bit mode, or used with a DMA controller. When the DAC operates in 12-bit mode, the data can be set to be left- 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. The DAC can be pin-inputted to a reference voltage, VREF+, for more accurate conversion results.

### 12.2 DAC Key Features

- 2 DAC converters: 1 output channel for each converter
- 8-bit or 12-bit monotonic output
- 12-bit mode with data left- or right-aligned
- Synchronized Update Function
- Noise waveform generation
- Triangle waveform generation
- Dual DAC channels converted simultaneously or separately
- DMA function for each channel
- Externally Triggered Conversion
- Input reference voltage VREF+
- The block diagram of a single DAC channel is shown below, and the pin descriptions are given Table 68.

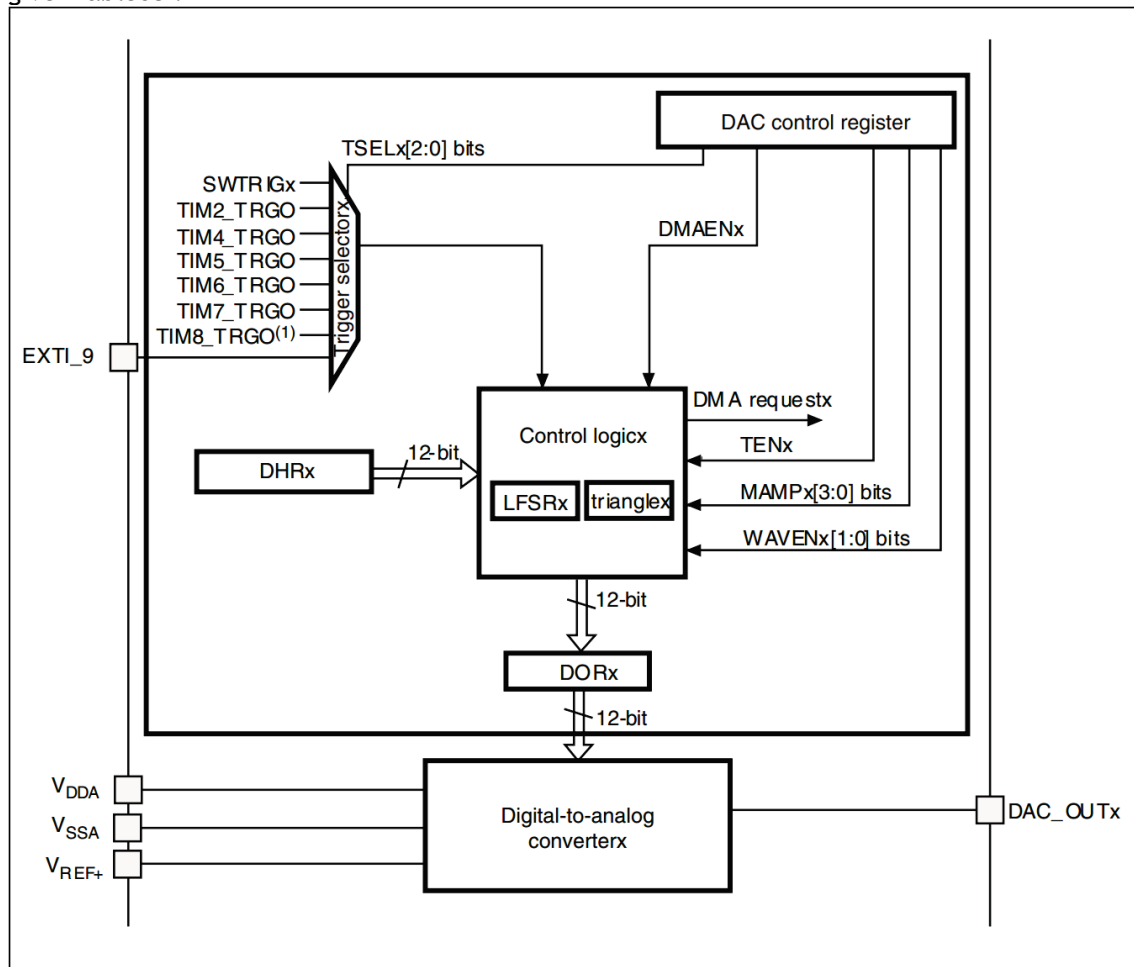


Figure 40 DAC Channel Module Block Diagram

Table68 DAC Pins

name (of a thing)	Model Type	marginal notes
VREF+	Input, Positive Analog Reference Voltage	High end/positive reference voltage used by DAC, $2.4V \leq VREF+ \leq VDDA(3.3V)$
VDDA	Input, Analog Power	analog power
VSSA	Input, Analog Power Ground	Ground for analog power supply
DAC_OUTx	Analog Output Signal	Analog output of DAC channel x

Notes: Once the DACx channel is enabled, the corresponding GPIO pin (PA4 or PA5) is automatically connected to the analog output of the DAC (DAC\_OUTx). To avoid parasitic interference and additional power consumption, pins PA4 or PA5 should be set to analog input (AIN) before.

## 12.3 DAC Functional Description

### 12.3.1 Enable DAC Channel

Power to DAC channel x is turned on by setting the ENx position '1' in the DAC\_CR register. After a startup time tWAKEUP, DAC channel x is enabled.

Notes: 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'.

### 12.3.2 Enable DAC output buffering

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.

### 12.3.3 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)

Depending on the operation of the DAC\_DHRyyx register, the written data is dumped into the DHRx register after a corresponding shift (DHRx is the internal data saving register x). Subsequently, the contents of the DHRx register are either automatically transferred to the DORx register or are transferred to the DORx register by a software trigger or an external event trigger.

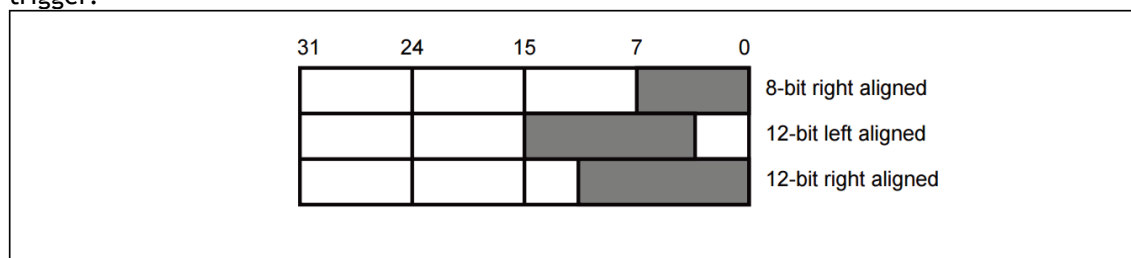


Figure41 Data Registers for Single DAC Channel Mode

- 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), 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), 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 into register DHR1[11:0] bits), write DAC channel 2 data into register DAC\_DHR12RD[27:16] bits (actually into register DHR2[11:0] bits)

Depending on the operation of the DAC\_DHRyyyD register, the written data is dumped into the DHR1 and DHR2 registers after corresponding shifts (DHR1 and DHR2 are internal data saving registers x). Subsequently, the contents of DHR1 and DHR2 are either automatically transferred to the DORx registers, or are transferred to the DORx registers via a software trigger or external event trigger.

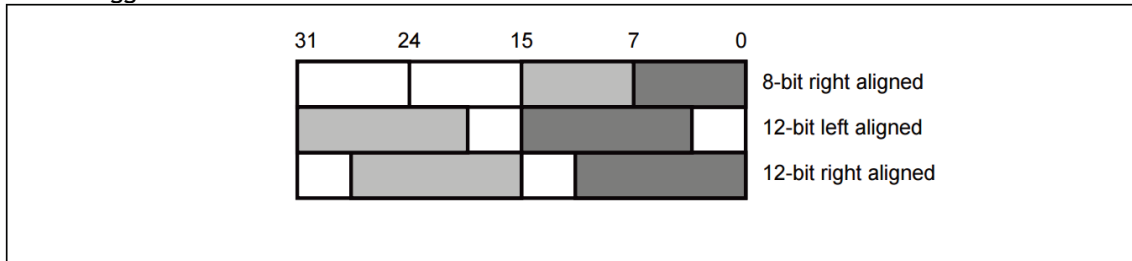


Figure 42 Data Registers for Dual DAC Channel Mode

### 12.3.4 DAC Conversion

Data cannot be written 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\_CR1), the data deposited into register DAC\_DHRx is automatically transferred to register DAC\_DORx after one APB1 clock cycle. If the hardware trigger is checked (TENx position '1' of register DAC\_CR1), the data transfer is completed after 3 APB1 clock cycles from the time when the trigger occurs. If hardware trigger is selected (TENx position '1' of register DAC\_CR1), data transfer is completed 3 APB1 clock cycles after the trigger occurs.

Once the data has been loaded from the DAC\_DHRx register into the DAC\_DORx register, the output is valid after the elapsed time  $t_{SETTLING}$ , which varies depending on the supply voltage and analog output load.

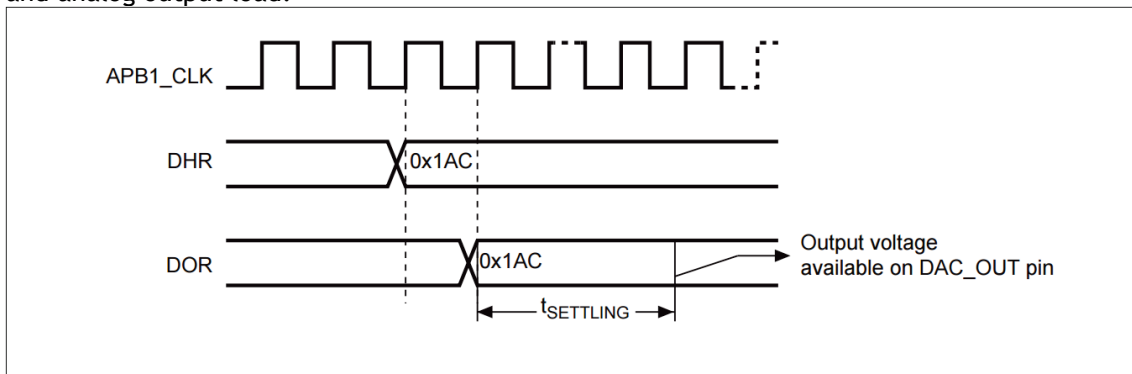


Figure 43 Time block diagram of conversion when TEN=0 triggers deactivation

### 12.3.5 DAC Output Voltage

The digital input is linearly converted by the DAC to an analog voltage output ranging from 0 to  $V_{REF+}$ .

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

$$\text{DAC output} = V_{REFx}(\text{DOR}/4095).$$

### 12.3.6 Select DAC Trigger

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

Table69 External Trigger

trigger source	typology	TSELx[2:0]
Timer 6TRGO event	Internal signal from on-chip timer	000
Timer 8TRGO event		001
Timer 7TRGO event		010
Timer 5TRGO event		011
Timer 2TRGO event		100
Timer 4TRGO event		101
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 output, or external interrupt line 9, the data most recently stored in register DAC\_DHRx is transferred to register DAC\_DORx. After 3 APB1 clock cycles, register DAC\_DORx is updated to the new value. If software triggering is selected, the conversion starts as soon as the SWTRIG bit is '1'. The SWTRIG bit is automatically cleared '0' by hardware after the data is transferred from the DAC\_DHRx register to the DAC\_DORx register.

- Notes:
1. the TSELx[2:0] bits cannot be changed while ENx is '1'.
  2. If software trigger is selected, data transfer from register DAC\_DHRx to register DAC\_DORx takes only one APB1 clock cycle.

### 12.3.7 DMA Request

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

If DMAENx position '1', once an external trigger (not a software trigger) occurs, a DMA request is generated and then the data in the DAC\_DHRx register is transferred to the DAC\_DORx register.

In Dual DAC mode, if the DMAENx bits of both channels are '1', 2 DMA requests are generated. If only one DMA transfer is actually required, only one of the DMAENx bits '1' should be selected. In this way, the program can handle 2 DAC channels operating in Dual DAC mode while using only one DMA request for one DMA channel.

DMA requests from the DAC are not cumulative, so if the 2nd external trigger occurs before the response to the 1st external trigger, the 2nd DMA request cannot be processed and no error is reported.

### 12.3.8 Noise Generation

Pseudo-noise with varying amplitude can be generated using the Linear Feedback Shift Register LFSR. Setting the WAVE[1:0] bits to '01' selects the DAC noise generation function. The pre-loaded value of register LFSR is 0xAAA. the value of this register is updated after 3 APB1 clock cycles after each trigger event according to a specific algorithm.

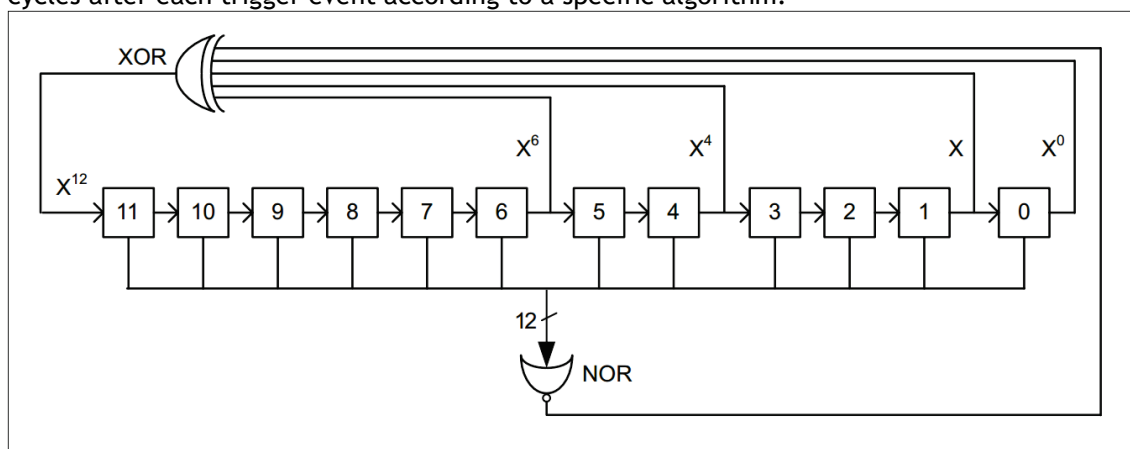


Figure44 DACLFSR Register Algorithm

Setting the MAMPx[3:0] bits of the DAC\_CR register can mask some or all of the LFSR data, so that the LSFR value obtained is added with the value of DAC\_DHRx, and after removing the overflow bits, it is written to the DAC\_DORx register.



If the register LFSR value is 0x000, a '1' is injected (anti-locking mechanism). Setting WAVEx[1:0] position '0' resets the LFSR waveform generation algorithm.

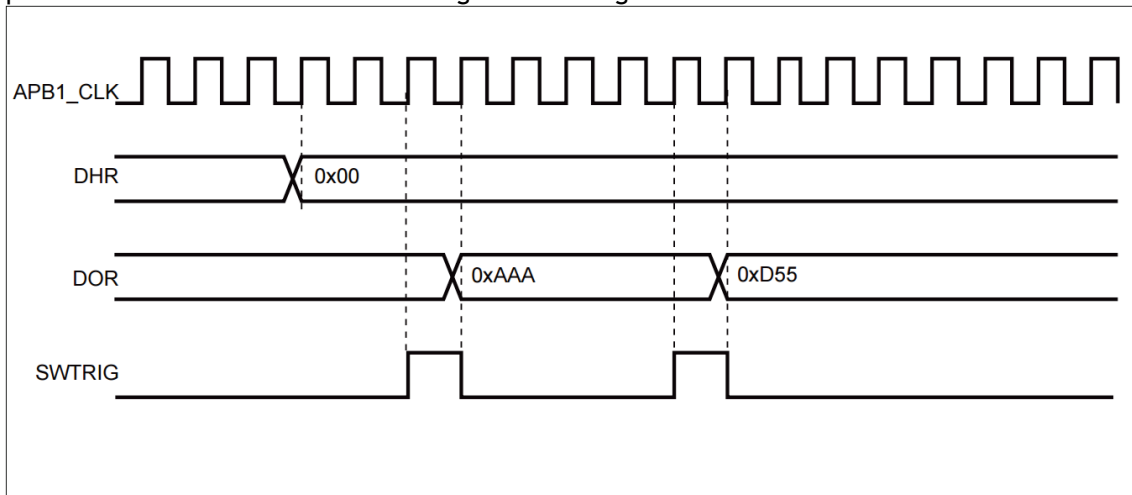


Figure45 DAC Conversion with LFSR Waveform Generation (Enable Software Trigger)

Notes: In order to generate noise, DAC triggering must be enabled, i.e., set the TENx bit of the DAC\_CR register to '1'.

### 12.3.9 Triangle Wave Generation

A small amplitude triangle wave can be added to a DC or slowly changing signal. Setting the WAVEx[1:0] bits to '10' selects the triangle wave generation function of the DAC. Setting the MAMPx[3:0] bits of the DAC\_CR register selects the amplitude of the triangle wave. The internal triangle wave counter accumulates 1 after 3 APB1 clock cycles after each trigger event. the counter value is added to the value in the DAC\_DHRx register and the overflow bit is discarded and written to the DAC\_DORx register. The delta wave counter accumulates gradually when the value passed into the DAC\_DORx register is less than the maximum amplitude defined by the MAMP[3:0] bits. Once the set maximum amplitude is reached, the counter starts to decrement, reaches 0 and starts to accumulate again, and so on.

Positioning WAVEx[1:0] at '0' resets the triangle wave generation.

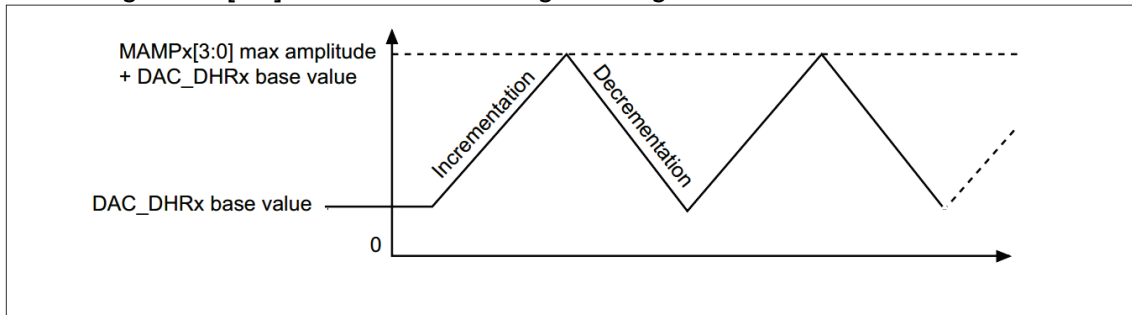


Figure46 DAC Triangle Wave Generation

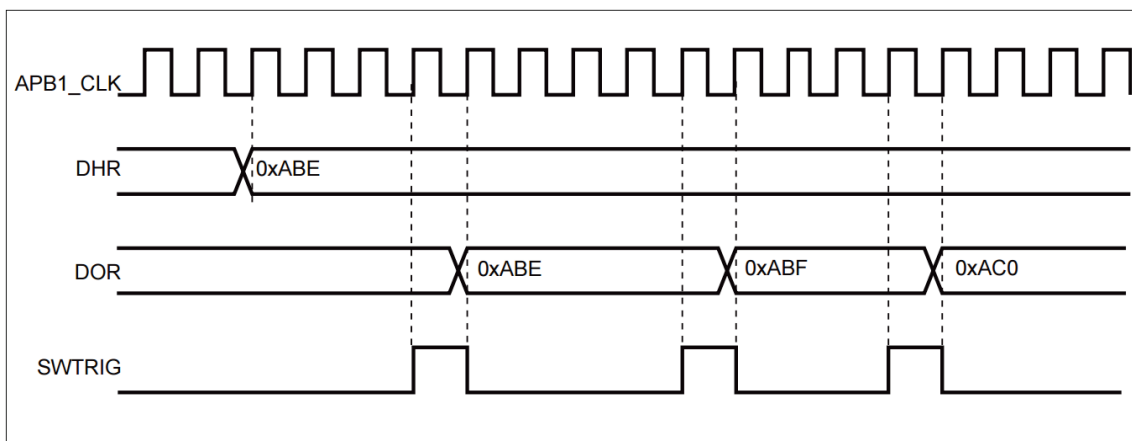


Figure47 DAC conversion with delta generation (enable software trigger)

Notes: 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 MAMP[3:0] bits must be set before enabling the DAC, otherwise their value cannot be modified.

## 12.4 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.

### 12.4.1 Stand-Alone Triggering Without Waveform Generator

Set the DAC to operate in this conversion mode in the following order:

- Set the trigger enable bits TEN1 and TEN2 of the 2 DAC channels to '1' respectively;
- Configure different trigger sources for each of the 2 DAC channels by setting the TSEL1[2:0] and 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 APB1 clock cycles) the value of register DHR1 is passed into register DAC\_DOR1.

When a DAC channel 2 trigger event occurs, (after a delay of 3 APB1 clock cycles) the value of register DHR2 is passed into register DAC\_DOR2.

### 12.4.2 Independent Triggering Using the Same LFSR

Set the DAC to operate in this conversion mode in the following order:

- Set the trigger enable bits TEN1 and TEN2 of the 2 DAC channels to '1' respectively;
- Configure different trigger sources for each of the 2 DAC channels by setting the TSEL1[2:0] and TSEL2[2:0] bits to different values;
- Set the WAVEx[1:0] bits of the 2 DAC channels to "01" and set 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 DHR1 register value, and (after a delay of 3 APB1 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 DHR2 register value, and (after a delay of 3 APB1 clock cycles) the result is passed into register DAC\_DOR2, which then updates the LFSR2 counter.

### 12.4.3 Independent Triggering Using Different LFSRs

Set the DAC to operate in this conversion mode in the following order:

- Set the trigger enable bits TEN1 and TEN2 of the 2 DAC channels to '1' respectively;
- Configure different trigger sources for each of the 2 DAC channels by setting the TSEL1[2:0] and TSEL2[2:0] bits to different values;
- Set the WAVEx[1:0] bits of the 2 DAC channels to "01" and set 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 MAMP1[3:0] is added to the DHR1 register value, and (after a delay of 3 APB1 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 MAMP2[3:0] is added to the DHR2 register value, and (after a delay of 3 APB1 clock cycles) the result is passed into register DAC\_DOR2, which then updates the LFSR2 counter.

---

#### 12.4.4 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 TEN1 and TEN2 of the 2 DAC channels to '1' respectively;
- Configure different trigger sources for each of the 2 DAC channels by setting the TSEL1[2:0] and TSEL2[2:0] bits to different values;
- Set the WAVEx[1:0] bits of the 2 DAC channels to "1x" and set 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 DHR1 register, and (after a delay of 3 APB1 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 DHR2 register, and (after a delay of 3 APB1 clock cycles) the result is passed into register DAC\_DOR2, which then updates the DAC channel 2 triangular wave counter.

#### 12.4.5 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 TEN1 and TEN2 of the 2 DAC channels to '1' respectively;
- Configure different trigger sources for each of the 2 DAC channels by setting the TSEL1[2:0] and TSEL2[2:0] bits to different values;
- Set the WAVEx[1:0] bits of the 2 DAC channels to '1x' and set 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 triangular wave amplitude value set by MAMP1[3:0] is added to the DHR1 register value, and (after a delay of 3 APB1 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 delta wave amplitude value set by MAMP2[3:0] is added to the DHR2 register value, and (after a delay of 3 APB1 clock cycles) the result is passed into register DAC\_DOR2, which then updates the DAC channel 2 delta wave counter.

#### 12.4.6 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 APB1 clock cycle, the values of the DHR1 and DHR2 registers are passed into the DAC\_DOR1 and DAC\_DOR2 registers, respectively.

#### 12.4.7 Simultaneous Triggering Without Waveform Generator

Set the DAC to operate in this conversion mode in the following order:

- Set the trigger enable bits TEN1 and TEN2 of the 2 DAC channels to '1' respectively;
- Configure the 2 DAC channels to use the same trigger source by setting the TSEL1[2:0] and 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 APB1 clock cycles) the values of the DHR1 and DHR2 registers are passed into the DAC\_DOR1 and DAC\_DOR2 registers, respectively.

#### 12.4.8 Simultaneous Triggering Using the Same LFSR

Set the DAC to operate in this conversion mode in the following order:

- Set the trigger enable bits TEN1 and TEN2 of the 2 DAC channels to '1' respectively;
- Configure the 2 DAC channels to use the same trigger source by setting the TSEL1[2:0] and TSEL2[2:0] bits to the same value, respectively;
- Set the WAVEx[1:0] bits of the 2 DAC channels to "01" and set 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 MAMP1[3:0] is added to the value of the DHR1 register, and (after a delay of 3 APB1 clock cycles) the result is passed to the DAC\_DOR1 register, which then updates the LFSR1 counter.
- Similarly, the LFSR2 counter value of the mask set by MAMP1[3:0] is added to the value of the DHR2 register, and (after a delay of 3 APB1 clock cycles) the result is passed into register DAC\_DOR2, which then updates the LFSR2 counter.

## 12.4.9 Simultaneous Triggering Using Different LFSRs

Set the DAC to operate in this conversion mode in the following order:

- Set the trigger enable bits TEN1 and TEN2 of the 2 DAC channels to '1' respectively;
- Configure the 2 DAC channels to use the same trigger source by setting the TSEL1[2:0] and TSEL2[2:0] bits to the same value, respectively;
- Set the WAVEx[1:0] bits of the 2 DAC channels to '01' and set 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 DHR1 register value, and (after a delay of 3 APB1 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 DHR2 register value, and (after a delay of 3 APB1 clock cycles) the result is passed into register DAC\_DOR2, which then updates the LFSR2 counter.

## 12.4.10 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 TEN1 and TEN2 of the 2 DAC channels to '1' respectively;
- Configure the 2 DAC channels to use the same trigger source by setting the TSEL1[2:0] and TSEL2[2:0] bits to the same value, respectively.
- Set the WAVEx[1:0] bits of the 2 DAC channels to '1x' and set 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 triangular wave amplitude is added to the DHR1 register value and (after a delay of 3 APB1 clock cycles) the result is passed into register DAC\_DOR1, which then updates the LFSR1 counter.

At the same time, the same triangular wave amplitude is added to the DHR2 register value and (after a delay of 3 APB1 clock cycles) the result is passed into register DAC\_DOR2, which then updates the LFSR2 counter.

## 12.4.11 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 TEN1 and TEN2 of the 2 DAC channels to '1' respectively;
- Configure the 2 DAC channels to use the same trigger source by setting the TSEL1[2:0] and TSEL2[2:0] bits to the same value, respectively.
- Set the WAVEx[1:0] bits of the 2 DAC channels to '1x' and set 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 triangle wave amplitude set by MAMP1[3:0] is added to the DHR1 register value, and (after a delay of 3 APB1 clock cycles) the result is passed to register DAC\_DOR1, which then updates the LFSR1 counter.

At the same time, the triangular wave amplitude set by MAMP2[3:0] is added to the DHR2 register value, and (after a delay of 3 APB1 clock cycles) the result is passed to register DAC\_DOR2, which then updates the LFSR2 counter.

## 12.5 DAC Register

These peripheral registers must be operated in word (32-bit) format.

### 12.5.1 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
Reserved			DMA EN2	MAMP2[3:0]				WAVE2[2:0]		TSEL2[2:0]			TEN2	BOFF2	EN2
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			DMA EN1	MAMP1[3:0]				WAVE1[2:0]		TSEL1[2:0]			TEN1	BOFF1	EN1
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	notation	clarification
31:29	Reserved	Reserved, always reads 0.
28	DMAEN2	<b>DMAEN2</b> : DAC channel2 DMA enable This bit is set and cleared by software. 0: Disables the DAC channel 2DMA mode; 1: Enable DAC channel 2DMA mode.
27:24	MAMP2[3:0]	<b>MAMP2[3:0]</b> : DAC channel2 mask/amplitude selector 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. 0000: Unmasked LSFR bit 0/triangular wave amplitude equal to 1; 0001: Unmasked LSFR bits [1:0]/Triangle wave amplitude equal to 3; 0010: Unmasked LSFR bits [2:0]/Triangle wave amplitude equal to 7; 0011: Unmasked LSFR bits [3:0]/Triangle wave amplitude equal to 15; 0100: Unmasked LSFR bits [4:0]/Triangle wave amplitude equal to 31; 0101: Unmasked LSFR bits [5:0]/Triangle wave amplitude equal to 63; 0110: Unmasked LSFR bits [6:0]/Triangle wave amplitude equal to 127; 0111: Not masked LSFR bits [7:0]/Triangle wave amplitude equal to 255; 1000: Unmasked LSFR bits [8:0]/Triangle wave amplitude equal to 511; 1001: Unmasked LSFR bits [9:0]/Triangle wave amplitude equal to 1023; 1010: not masked LSFR bits [10:0]/triangle wave amplitude equal to 2047; ≥1011: unmasked LSFR bits [11:0]/triangular wave amplitude equal to 4095.
23:22	WAVE2[1:0]	<b>WAVE2[1:0]</b> : DAC channel2 noise/triangle wave generation enable (DAC channel2 noise/triangle wave generation enable) This 2-bit is set and cleared by software. 00: Turns off the waveform generator; 10: Enable the noise waveform generator; 1x: Enable the triangle wave generator.
21:19	TSEL2[2:0]	<b>TSEL2[2:0]</b> : DAC channel2 trigger selection (DAC channel2 trigger selection) This 3-bit is used to select the external trigger event for DAC channel 2. 000: TIM6TRGO Event; 001: TIM8TRGO Event; 010: TIM7TRGO Event; 011: TIM5TRGO event; 100: TIM2TRGO event; 101: TIM4TRGO event; 110: External interrupt line 9; 111: Software trigger. Note: This 3-bit can only be set when TEN2=1 (DAC channel 2 trigger enable).
18	TEN2	<b>TEN2</b> : DAC channel2 trigger enable This bit is set and cleared by software to enable/disable triggering of DAC channel 2. 0: Turn off DAC channel 2 triggering, data written to the DAC_DHRx register is passed into the DAC_DOR2 register after 1 APB1 clock cycle; 1: Enable DAC channel 2 triggering, data written to the DAC_DHRx register is passed into the DAC_DOR2 register after 3 APB1 clock cycles. Note: If software triggering is selected, the data written to register DAC_DHRx requires only 1 APB1 clock cycle to be passed to register DAC_DOR2.
17	BOFF2	<b>BOFF2</b> : disable DAC channel2 output buffer (DAC channel2 output buffer disable) This bit is set and cleared by software to enable/disable the DAC channel2 output buffer. 0: Enable DAC channel 2 output buffering; 1: Disables the DAC channel 2 output buffer.
16	EN2	<b>EN2</b> : DAC channel2 enable This bit is set and cleared by software to enable/disable DAC channel 2. 0: Close DAC channel 2; 1: Enable DAC channel 2.

15:13	Reserved	Reserved.
12	DMAEN1	<b>DMAEN1:</b> DAC channel1 DMA enable This bit is set and cleared by software. 0: Disables DAC channel 1 DMA mode; 1: Enable DAC channel 1 DMA mode.
11:8	MAMP1[3:0]	<b>MAMP1[3:0]:</b> DAC channel1 mask/amplitude selector 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. 0000: Unmasked LSFR bit 0/triangular wave amplitude equal to 1; 0001: Unmasked LSFR bits [1:0]/Triangle wave amplitude equal to 3; 0010: Unmasked LSFR bits [2:0]/Triangle wave amplitude equal to 7; 0011: Unmasked LSFR bits [3:0]/Triangle wave amplitude equal to 15; 0100: Unmasked LSFR bits [4:0]/Triangle wave amplitude equal to 31; 0101: Unmasked LSFR bits [5:0]/Triangle wave amplitude equal to 63; 0110: Unmasked LSFR bits [6:0]/Triangle wave amplitude equal to 127; 0111: Not masked LSFR bits [7:0]/Triangle wave amplitude equal to 255; 1000: Unmasked LSFR bits [8:0]/Triangle wave amplitude equal to 511; 1001: Unmasked LSFR bits [9:0]/Triangle wave amplitude equal to 1023; 1010: not masked LSFR bits [10:0]/triangle wave amplitude equal to 2047; ≥1011: unmasked LSFR bits [11:0]/triangular wave amplitude equal to 4095.
7:6	WAVE1[1:0]	<b>WAVE1[1:0]:</b> DAC channel1 noise/triangle wave generation enable (DAC channel1 noise/triangle wave generation enable) This 2-bit is set and cleared by software. 00: Turn off waveform generation; 10: Enable the noise waveform generator; 1x: Enable the triangle wave generator.
5:3	TSEL1[2:0]	<b>TSEL1[2:0]:</b> DAC channel1 trigger selection (DAC channel1 trigger selection) This bit is used to select the external trigger event for DAC channel 1. 000: TIM6TRGO Event; 001: TIM8TRGO Event; 010: TIM7TRGO Event; 011: TIM5TRGO event; 100: TIM2TRGO event; 101: TIM4TRGO event; 110: External interrupt line 9; 111: Software trigger. Note: This bit can only be set when TEN1=1 (DAC channel 1 trigger enable).
2	TEN1	<b>TEN1:</b> DAC channel1 trigger enable This bit is set and cleared by software to enable/disable triggering of DAC channel 1. 0: Turn off DAC channel 1 trigger, data written to register DAC_DHRx is passed to register DAC_DOR1 after 1 APB1 clock cycle; 1: Enable DAC channel 1 trigger, data written to register DAC_DHRx is passed to register DAC_DOR1 after 3 APB1 clock cycles. Note: If software triggering is selected, the data written to register DAC_DHRx requires only 1 APB1 clock cycle to be passed into register DAC_DOR1.
1	BOFF1	<b>BOFF1:</b> disable DAC channel1 output buffer (DAC channel1 output buffer disable) This bit is set and cleared by software and is used to enable/disable the output buffer of DAC channel1. 0: Enable DAC channel 1 output buffering; 1: Disables the DAC channel 1 output buffer.
0	EN1	<b>EN1:</b> DAC channel1 enable This bit is set and cleared by software to enable/disable DAC channel 1. 0: Turns off DAC channel 1; 1: Enable DAC channel 1.

## 12.5.2 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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													SWTRIG2		SWTRIG1
													w		w

Bit	notation	clarification
31:29	Reserved	Reserved, always reads 0.
1	SWTRIG2	<b>SWTRIG2:</b> DAC channel2 software trigger (DAC channel2 software trigger) This bit is set and cleared by software to enable/disable the software trigger. 0: Disables DAC channel 2 software triggering; 1: Enable DAC channel 2 software trigger.



		Note: Once the data from register DAC_DHR2 is passed into register DAC_DOR2, (after 1 APB1 clock cycle) this bit is set '0' by hardware.
0	SWTRIG1	SWTRIG1: DAC channel1 software trigger (DAC channel1 software trigger) This bit is set and cleared by software to enable/disable the software trigger. 0: Disables DAC channel 1 software triggering; 1: Enable DAC channel 1 software trigger. Note: Once the data from register DAC_DHR1 is passed into register DAC_DOR1, (after 1 APB1 clock cycle) this bit is set '0' by hardware.

### 12.5.3 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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				DACC1DHR [11:0]											
				rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW

Bit	notation	clarification
31:16	Reserved	Reserved, always reads 0.
11:0	DACC1DHR [11:0]	DACC1DHR[11:0]: 12-bit right-aligned data for DAC channel 1 (DAC channel1 12-bit right-aligned data) This bit is written by software and indicates the 12-bit data for DAC channel 1.

### 12.5.4 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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC1DHR [11:0]												Reserved			
rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW

Bit	notation	instructions
31:16	Reserved	Reserved, always reads 0.
15:4	DACC1DHR [11:0]	DACC1DHR[11:0]: 12-bit right-aligned data for DAC channel 1 (DAC channel1 12-bit right-aligned data) This bit is written by software and indicates the 12-bit data for DAC channel 1.
3:0	Reserved	Reserved, always reads 0.

### 12.5.5 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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								DACC1DHR[7:0]							
								rW	rW	rW	rW	rW	rW	rW	rW

Bit	notation	clarification
31:18	Reserved	Reserved, always reads 0.
7:0	DACC1DHR[7:0]	DACC1DHR[7:0]: 8-bit right-aligned data for DAC channel 1 (DAC channel1 8-bit right-aligned data) This bit is written by software and indicates the 8-bit data for DAC channel 1.

## 12.5.6 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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				DACC2DHR [11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit	notation			clarification											
31:12	Reserved			Reserved, always reads 0.											
11:0	DACC2DHR [11:0]			DACC2DHR[11:0]: 12-bit right-aligned data for DAC channel 2 (DAC channel2 12-bit right-aligned data) This bit is written by software and indicates the 12-bit data for DAC channel 2.											

## 12.5.7 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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHR [11:0]												Reserved			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bit	notation	clarification
31:16	Reserved	Reserved, always reads 0.
15:4	DACC2DHR [11:0]	DACC2DHR[11:0]: 12-bit left-aligned data for DAC channel 2 (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	Reserved	Reserved, always reads 0.

## 12.5.8 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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								DACC2DHR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
Bit	notation			clarification											
31:18	Reserved			Reserved, always reads 0.											
7:0	DACC2DHR [7:0]			DACC2DHR[7:0]: 8-bit right-aligned data for DAC channel 2 (DAC channel2 8-bit right-aligned data) This bit is written by software and indicates the 8-bit data for DAC channel 2.											

## 12.5.9 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
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----



Reserved				DACC2DHR [11:0]											
				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
Reserved				DACC1DHR [11:0]											
				rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW

Bit	notation	clarification
31:28	Reserved	Reserved, always reads 0.
27:16	DACC2DHR [11:0]	DACC2DHR [11:0]:12-bit right-aligned data for DAC channel 2 This bit is written by software and indicates the 12-bit data for DAC channel 2.
7:0	DACC2DHR [11:0]	DACC2DHR[11:0]: 12-bit right-aligned data for DAC channel 1 This bit is written by software and indicates the 12-bit data for DAC channel 1.

## 12.5.10 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]												Reserved			
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
DACC1DHR [11:0]												Reserved			
rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW				
Bit	notation	clarification													
31:20	DACC2DHR [11:0]	DACC2DHR[11:0]: 12-bit left-aligned data for DAC channel 2 (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	Reserved	Reserved, always reads 0.													
15:4	DACC1DHR [11:0]	DACC1DHR[11:0]: 12-bit left-aligned data for DAC channel 1 (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	Reserved	Reserved, always reads 0.													

## 12.5.11 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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHR[7:0]								DACC1DHR[7:0]							
rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW
Bit	notation	clarification													
31:16	Reserved	Reserved, always reads 0.													
15:8	DACC2DHR [7:0]	DACC2DHR[7:0]: 8-bit right-aligned data for DAC channel 2 (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]	DACC1DHR[7:0]: 8-bit right-aligned data for DAC channel 1 (DAC channel1 8-bit right-aligned data) This bit is written by software and indicates the 8-bit data for DAC channel 1.													

## 12.5.12 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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				DACC1DOR [11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit	notation			clarification											
31:12	Reserved			Reserved, always reads 0.											
11:0	DACC1DOR [11:0]			DACC1DOR[11:0]: DAC channel1 output data (DAC channel1 data output) This bit is written by the software and indicates the output data of DAC channel1.											

## 12.5.13 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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				DACC2DOR [11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit	notation			clarification											
31:12	Reserved			Reserved, always reads 0.											
11:0	DACC2DOR [11:0]			DACC2DOR[11:0]: DAC channel2 output data (DAC channel2 data output) This bit is written by the software and indicates the output data of DAC channel2.											

## 12.5.14 DAC Register Image

The following table lists all DAC registers.

Table 70 DAC Register Image

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0						
0x00	DAC_CR	Reserved			DMAEN2	MAMP2[3:0]				WAVE2 [2:0]		TSEL2 [2:0]			TEN2	BOFF2	EN2	Reserved			DMAEN1	MAMP13:0]				WAVE1 [2:0]		TSEL1 [2:0]			TEN1	BOFF1	EN1						
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x04	DAC_SWTRIGR	Reserved																										SWTRIG2	SWTRIG1										
	Reset value																											0	0										
0x08	DAC_DHR12R1	Reserved										DACC1DHR [11:0]											0	0	0	0	0	0	0	0	0	0	0	0					
	Reset value																						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x0C	DAC_DHR12L1	Reserved										DACC1DHR [11:0]											Reserved																
	Reset value											0 0 0 0 0 0 0 0 0 0 0 0 0 0																											
0x10	DAC_DHR8R1	Reserved										DACC1DHR[7:0]											0	0	0	0	0	0	0	0	0	0	0	0					
	Reset value																						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x14	DAC_DHR12R2	Reserved										DACC2DHR [11:0]											0	0	0	0	0	0	0	0	0	0	0	0	0				
	Reset value																						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x18	DAC_DHR12L2	Reserved										DACC2DHR [11:0]											Reserved																
	Reset value											0 0 0 0 0 0 0 0 0 0 0 0 0 0																											
0x1C	DAC_DHR8R2	Reserved										DACC2DHR[7:0]											0	0	0	0	0	0	0	0	0	0	0	0	0				
	Reset value																						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x20	DAC_DHR12RD	Reserved	DACC2DHR [11:0]										Reserved	DACC1DHR [11:0]																									
	Reset value		0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						
0x24	DAC_DHR12LD	DACC2DHR [11:0]										Reserved	DACC1DHR [11:0]											Reserved															
	Reset value	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						
0x28	DAC_DHR8RD	Reserved										DACC2DHR[7:0]											DACC1DHR[7:0]																
	Reset value											0 0 0 0 0 0 0 0 0 0 0 0 0 0											0 0 0 0 0 0 0 0 0 0 0 0 0 0																
0x2C	DAC_DOR1	Reserved										DACC1DOR [11:0]											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	Reset value																						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	DAC_DOR2	Reserved										DACC2DOR [11:0]											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Reset value																						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

See Table 1 for register start addresses.

## 13 Advanced Control Timer (TIM1 and TIM8)

### 13.1 Introduction to TIM1 and TIM8

The advanced control timers (TIM1 and TIM8) consist of a 16-bit auto-load counter driven by a programmable prescaler.

It is suitable for a variety of uses, including measuring the pulse width of an input signal (input capture), or generating an output waveform (output comparison, PWM, complementary PWM with embedded dead time, etc.).

Adjustment of pulse width and waveform period from a few microseconds to a few milliseconds is possible using the timer prescaler and the RCC clock control prescaler.

The advanced control timers (TIM1 and TIM8) and the general-purpose timer (TIMx) are completely independent; they do not share any resources. They can be operated synchronously, as described in the 13.3.20 section.

### 13.2 TIM1 and TIM8 Main Features

The functions of the TIM1 and TIM8 timers include:

- 16-bit up, down, up/down auto-load counter
- 16-bit programmable (can be modified in real time) prescaler, counter clock frequency division factor of any value between 1 ~ 65535
- Up to 4 independent channels:
  - Input Capture
  - Output Comparison
  - PWM generation (edge or center alignment mode)
  - Single pulse mode output
- Dead-time programmable complementary outputs
- Synchronization circuits using external signals to control timers and timer interconnects
- Repeat counter that allows the timer register to be updated after a specified number of counter cycles
- The brake input signal can place the timer output signal in a reset state or a known state.
- An interrupt/DMA is generated when the following events occur:
  - Updates: Counter overflow up/down, counter initialization (via software or internal/external trigger)
  - Trigger event (counter starts, stops, initializes, or counts triggered internally/externally)
  - Input Capture
  - Output Comparison
  - Brake signal input
- Supports incremental (quadrature) encoder and Hall sensor circuits for positioning
- Trigger input as external clock or per-cycle current management

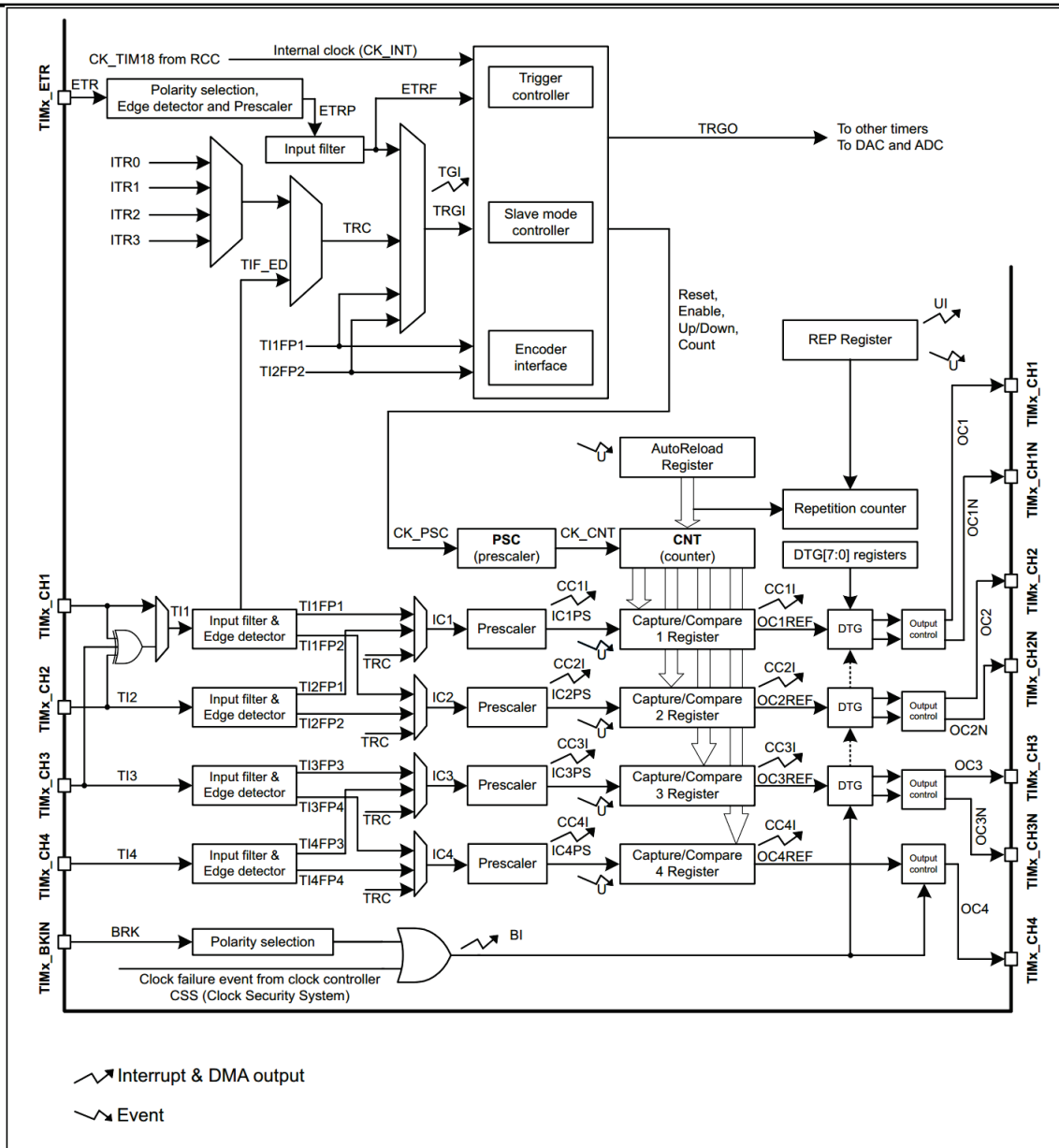



Figure48 Level Control Timer Block Diagram

Notes:  Depending on the setting of the control bit, the contents of the preloaded register are transmitted to the working register at the U (update) event

 event

 Interrupt and DMA output

## 13.3 TIM1 and TIM8 Functional Description

### 13.3.1 Time Base Unit (in computing)

The main part of the programmable advanced control timer is a 16-bit counter and its associated auto-load register. This counter can count up, count down, or count up and down in both directions. The counter clock is divided by a prescaler.

The counter, auto-load 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 contains:

- Counter Register (TIMx\_CNT)
- Prescaler Register (TIMx\_PSC)
- Auto-Reload Register (TIMx\_ARR)
- Repetition Count Register (TIMx\_RCR)

The autoloader registers are preloaded, and writing or reading the auto-reload registers will access the preload 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. An update event is generated when the counter reaches an overflow condition (underflow condition when counting down) and when the UDIS bit in the TIMx\_CR1 register were equal to zero. Update events can also be generated by software. The generation of update events for each configuration is described in detail subsequently.

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. (See the Slave Mode description of the controller for more details on enabling the counter). 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 frequency by any value between 1 and 65536. It is based on a 16-bit counter controlled by a 16-bit register (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 used when the next update event arrives.

Figure.49 and Figure 50 give examples of changing counter parameters while the prescaler is running.

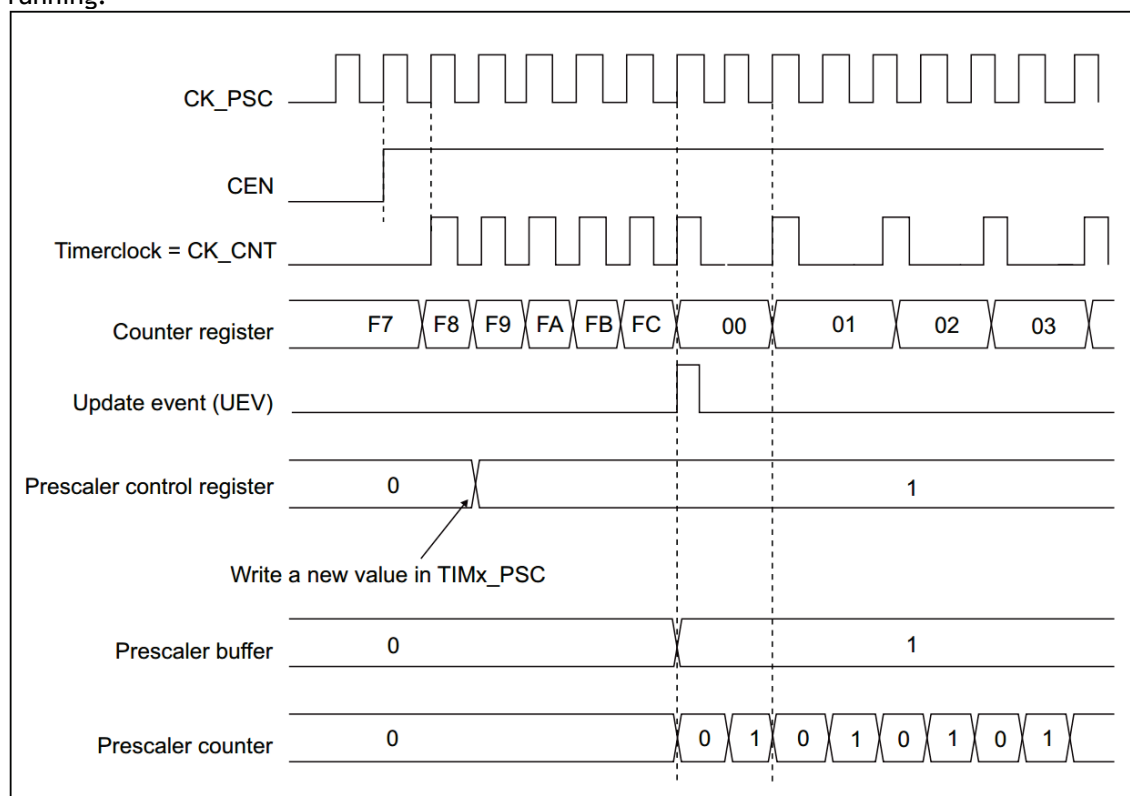


Figure.49 Timing diagram of the counter when the parameter of the prescaler is changed from 1 to 2

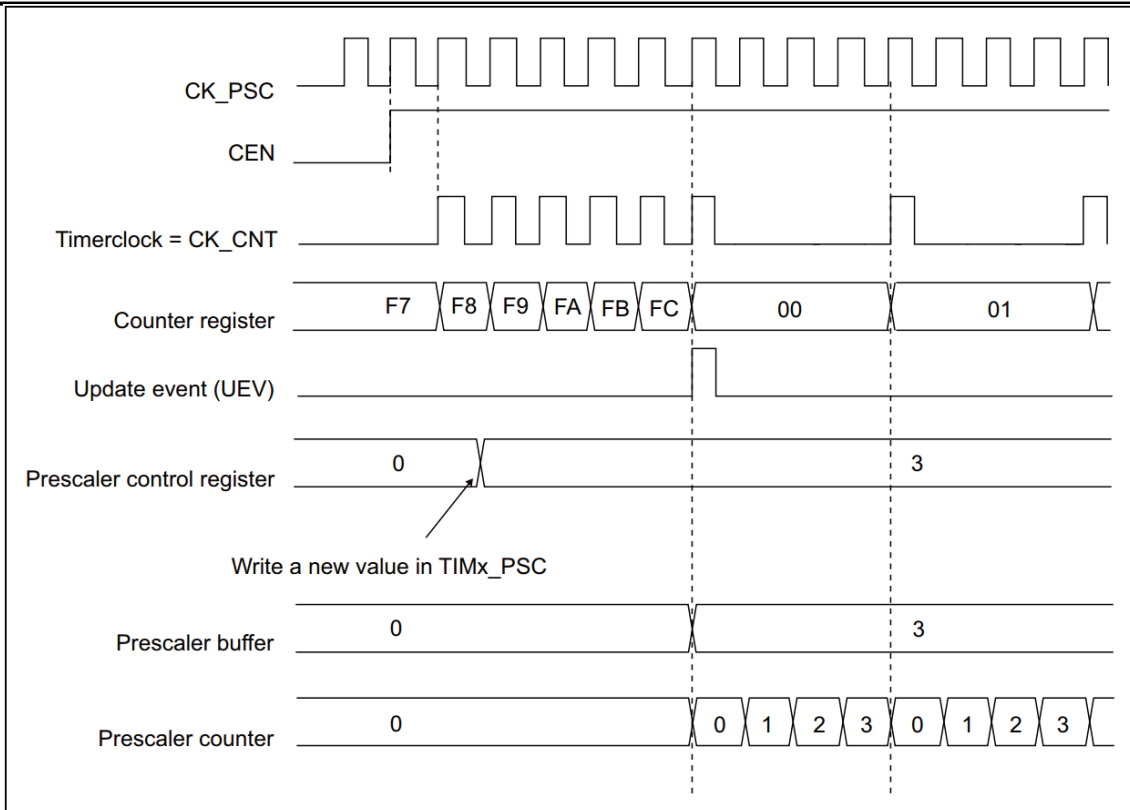


Figure50 Timing diagram of the counter when the parameter of the prescaler is changed from 1 to 4

## 13.3.2 Counter Mode

### Up Count Mode

In count-up mode, the counter counts from 0 to the auto-load value (the contents of the TIMx\_ARR counter), then starts counting from 0 again and generates a counter overflow event. If the repeat counter function is used, the update event (UEV) is generated when the up count reaches the set number of repeat counts (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) can similarly generate an update event.

Setting the UDIS bit in the TIMx\_CR1 register disables the update event; this prevents the shadow register from being updated when a new value is written to the preload register. An update event will not be generated until the UDIS bit is cleared '0'. However, when an update event should be generated, the counter will still be cleared '0' and the prescaler count will be invited to 0 (but the prescaler value will remain unchanged). In addition, if the URS bit in the TIMx\_CR1 register is set (select update request), 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 the counter is cleared 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 autoloading 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 the action of the counter at different clock frequencies when TIMx\_ARR=0x36.

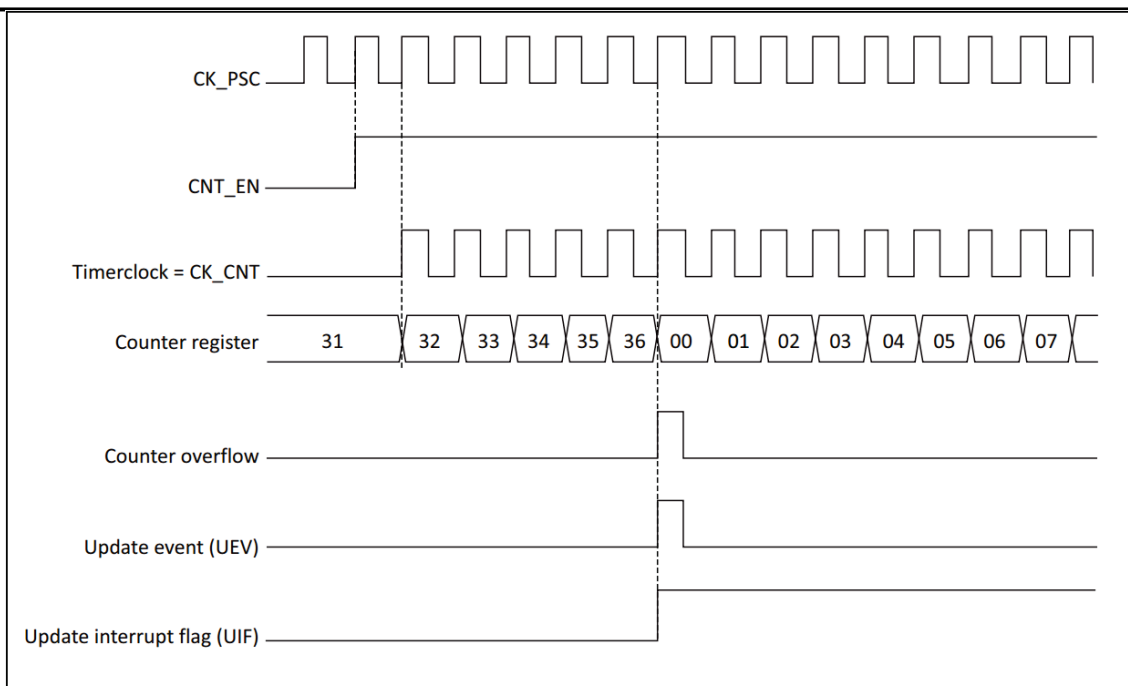


Figure51 Timing diagram of the counter with internal clock division factor of 1

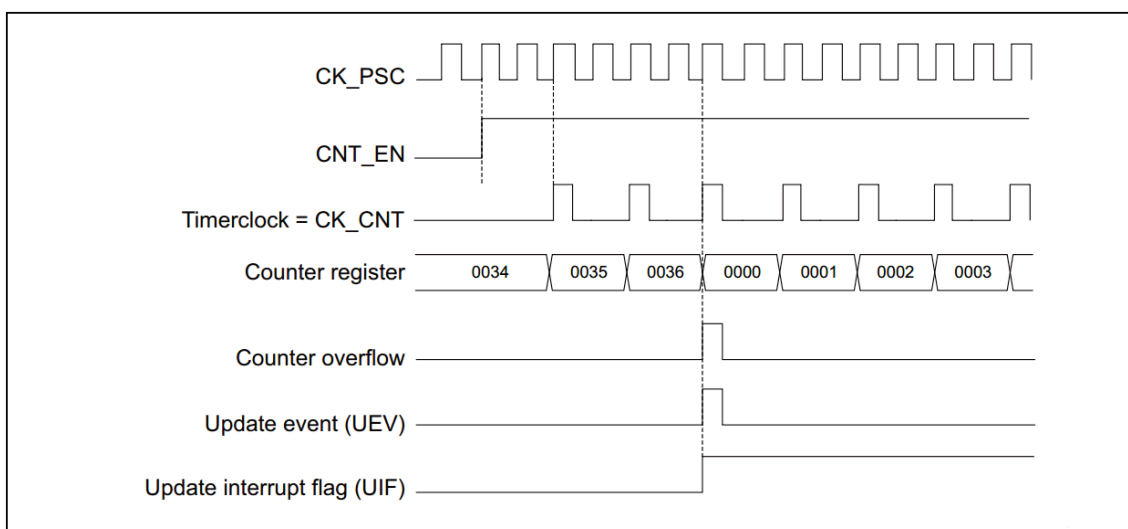


Figure52 Timing diagram of the counter with internal clock division factor of 2

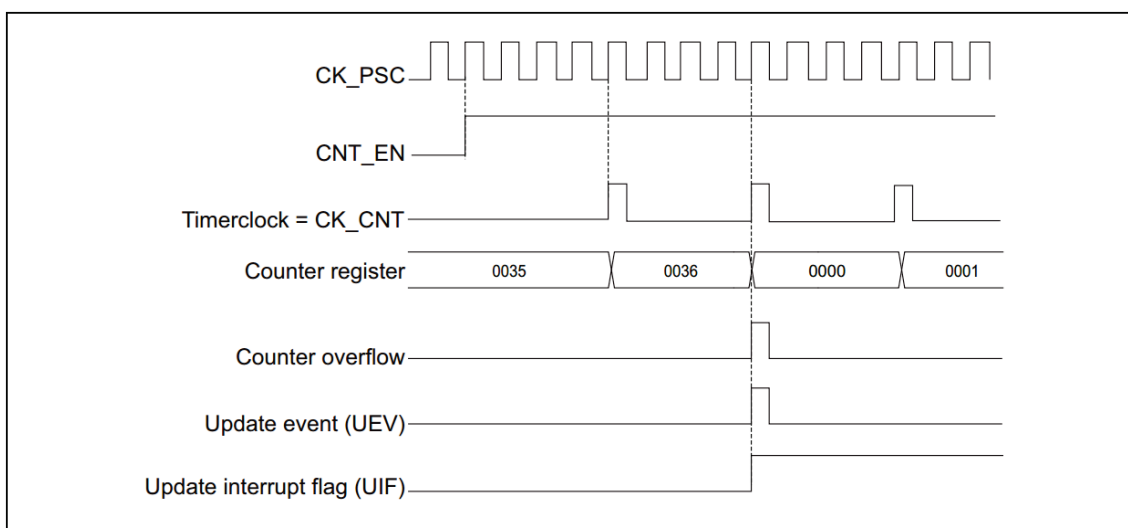


Figure53 Timing diagram of the counter with internal clock division factor of 4



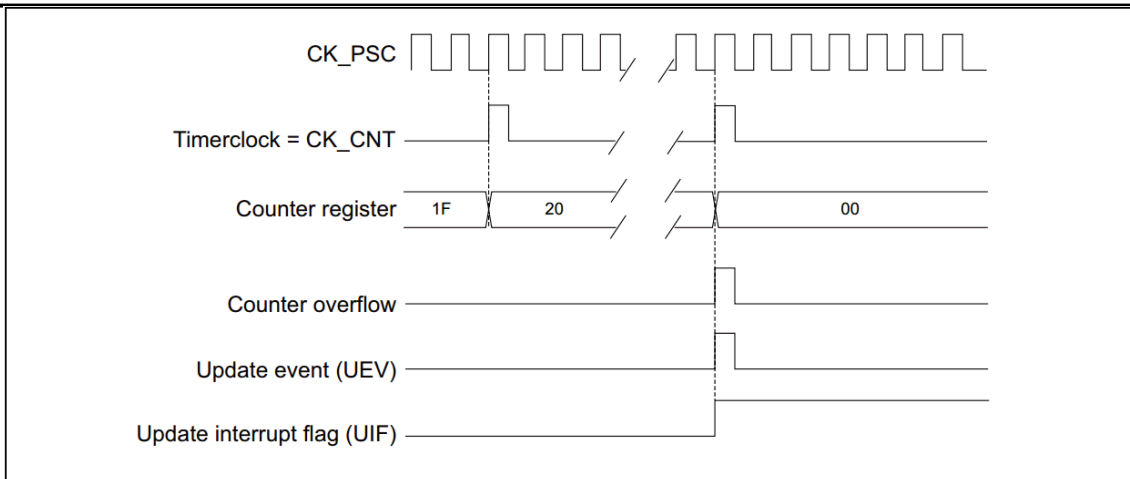


Figure54 Timing diagram of the counter with internal clock division factor N

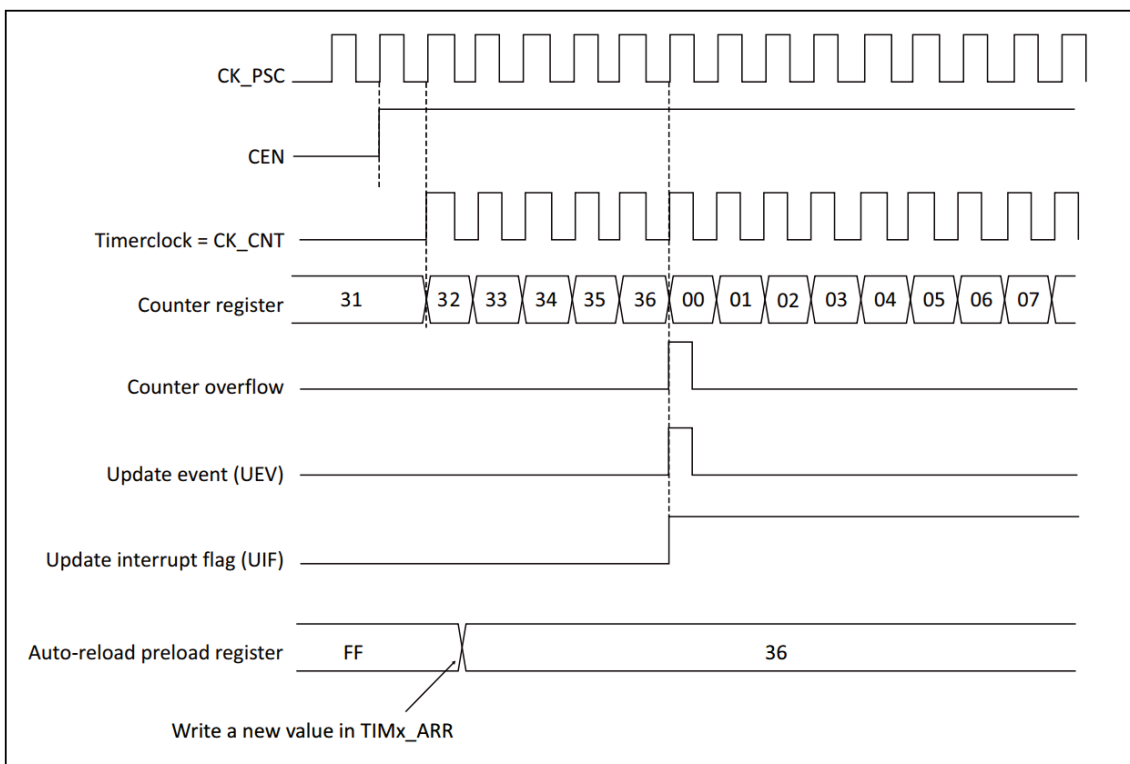


Figure55 Counter timing diagram, update event when ARPE=0 (TIMx\_ARR is not preloaded)

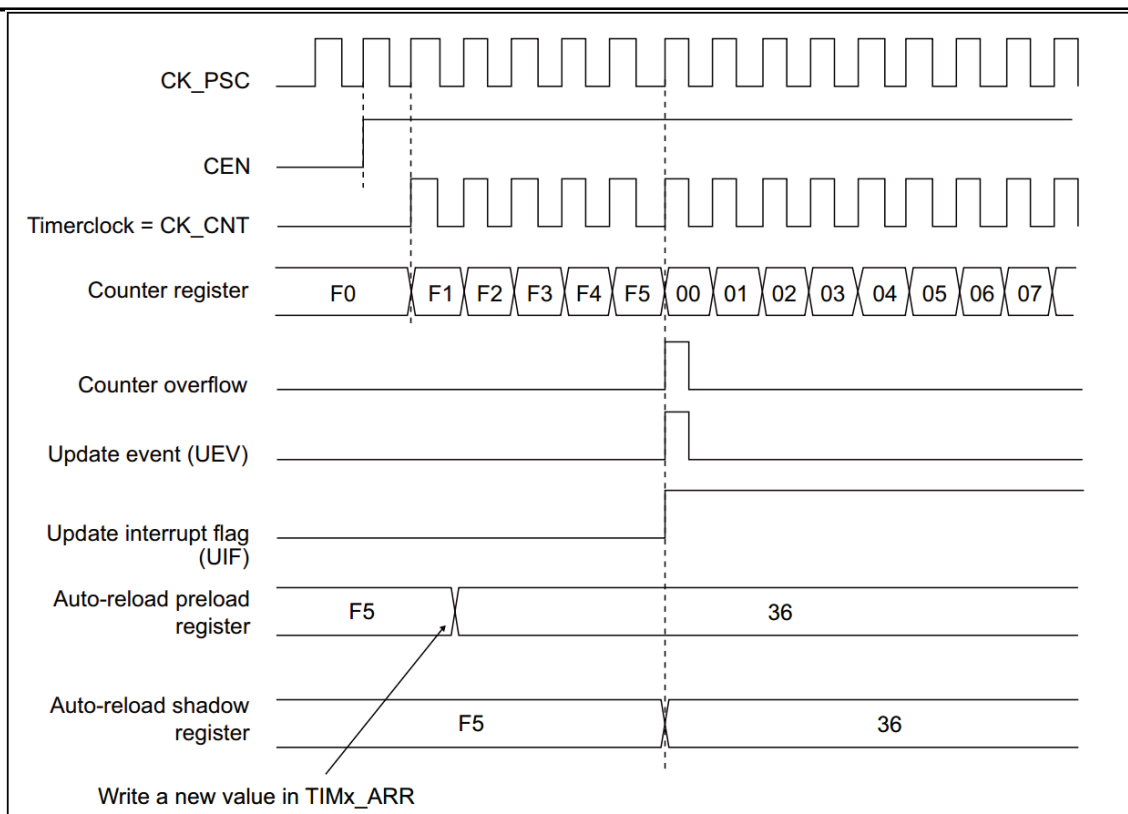


Figure 56 Counter timing diagram, update event when ARPE=1 (preloaded with TIMx\_ARR)

### Down Count Mode

In down mode, the counter counts down to 0 from the auto-loaded value (the value of the TIMx\_ARR counter), then restarts from the auto-loaded value and generates a counter down 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 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 of the TIMx\_CR1 register disables UEV events. This prevents the shadow register from being updated when a new value is written to the preload register. Therefore no update event is generated until the UDIS bit is cleared to 0. However, the counter will still restart counting from the current autoloader value and the prescaler counter restarts from 0 (but the prescaler coefficient remains unchanged).

In addition, if the URS bit in the TIMx\_CR1 register is set (select 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 DMA request), this is to avoid generating both an update and capture interrupt 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 preloaded value (the value of the TIMx\_PSC register).
- The current autoloader register is updated to the preloaded value (the contents of the TIMx\_ARR register). Note: The autoloader is updated before the counter is reloaded, so the next cycle will be the expected value.

Here are some examples of counter operation at different clock frequencies when TIMx\_ARR=0x36.

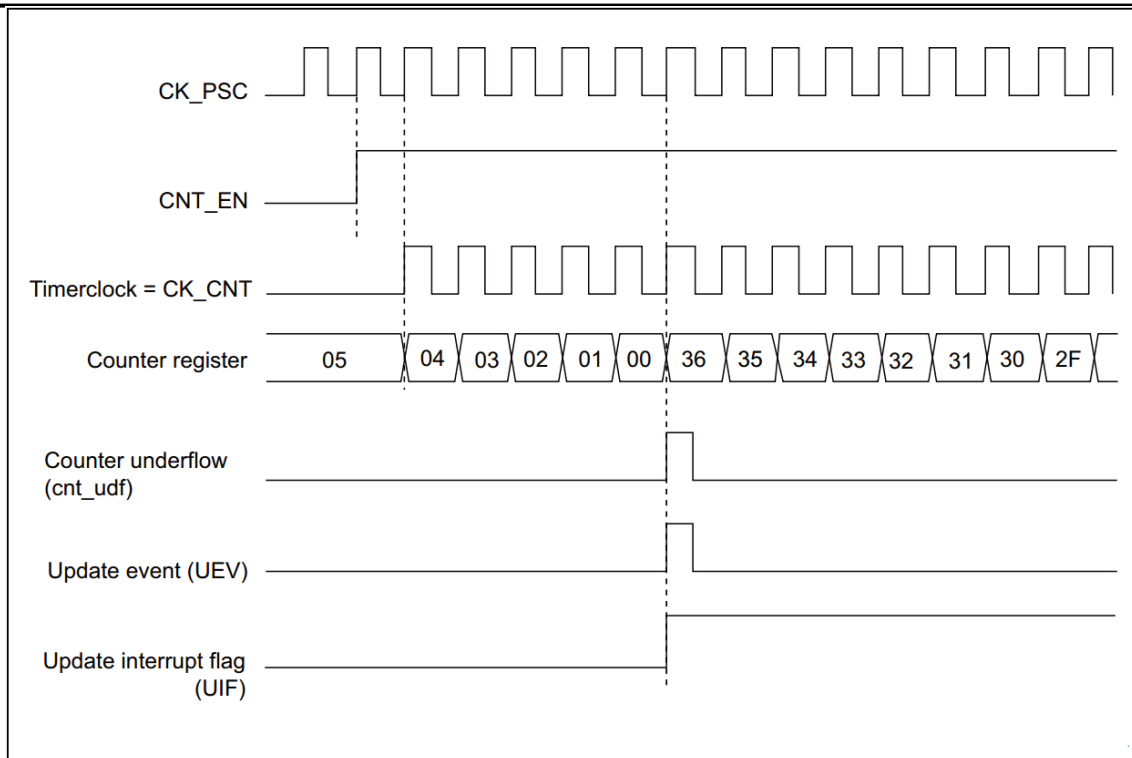


Figure57 Timing diagram of the counter with internal clock division factor of 1

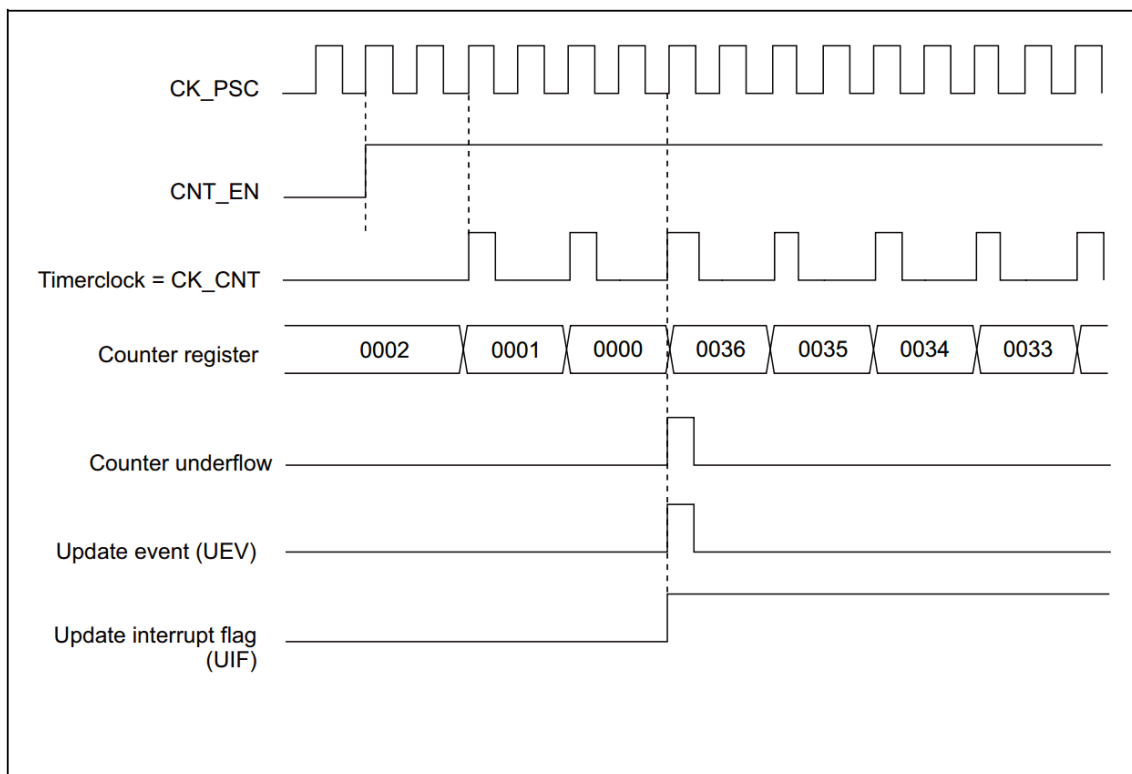


Figure58 Timing diagram of the counter with internal clock division factor of 2

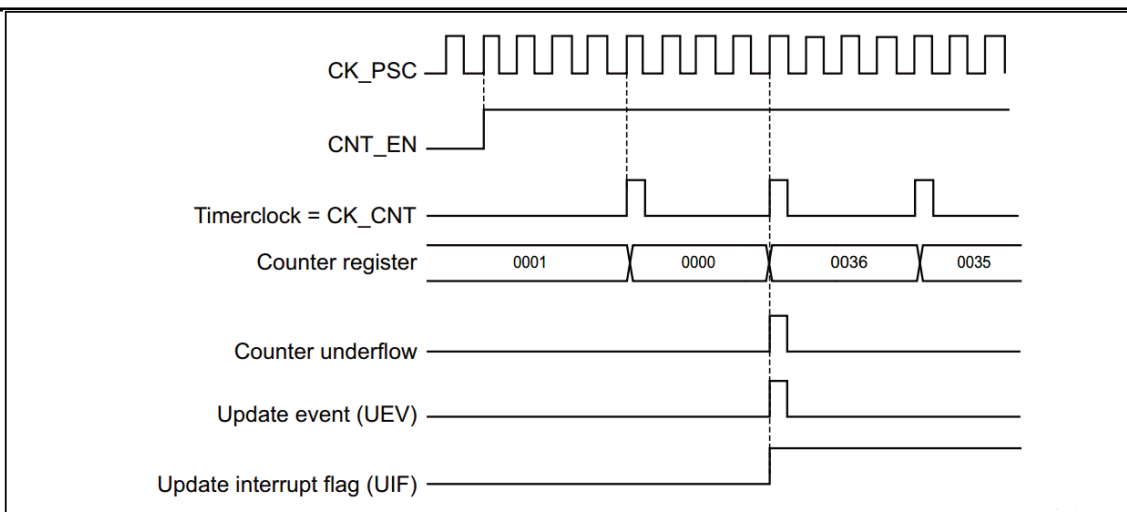


Figure59 Timing diagram of the counter with internal clock division factor of 4

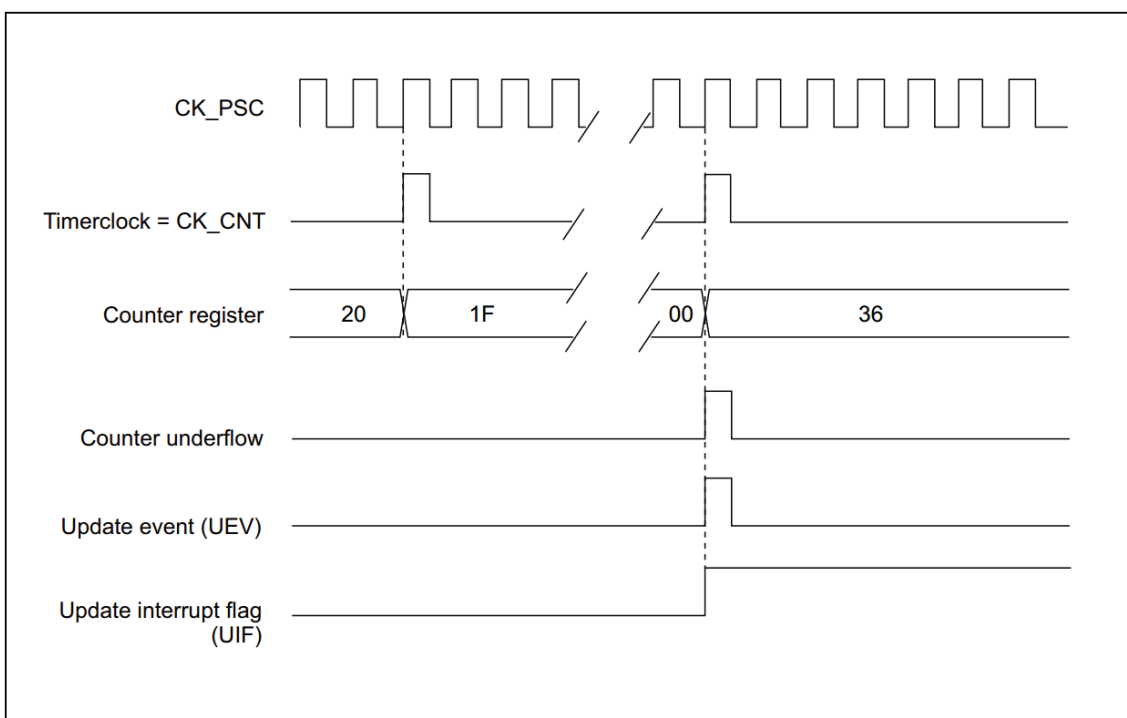


Figure60 Timing diagram of the counter with internal clock division factor N

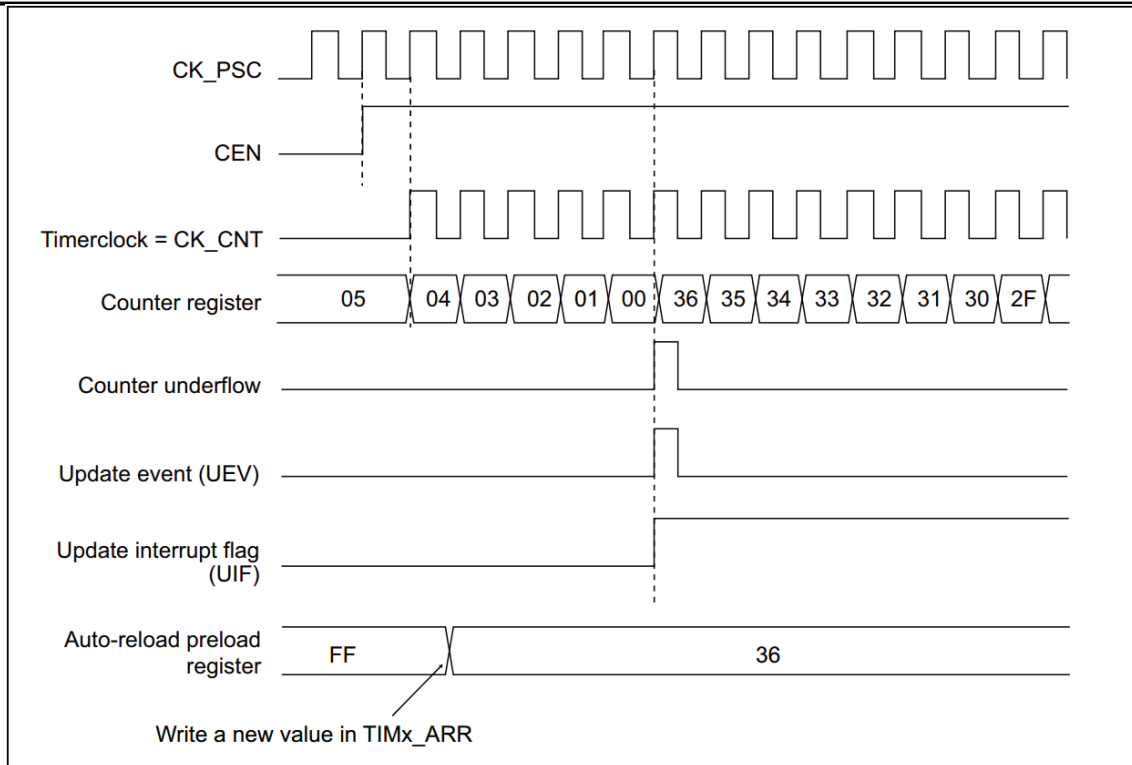


Figure61 Counter Timing Diagram, Update Events When Repeat Counter is Not Used

### 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; and then counts from 0 again.

In this mode, the DIR direction bit in TIMx\_CR1 cannot be written. It is updated by hardware and indicates the current count direction.

An update event can be generated on every count overflow and every count underflow; it 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 prevents the shadow register from being updated when a new value is written to the preload register. Therefore no update events are generated until the UDIS bit is cleared to 0. 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 (select 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 DMA request), this is to avoid generating both an update and capture interrupt 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 (the contents of 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).

The following are some examples of counter operation at different clock frequencies:

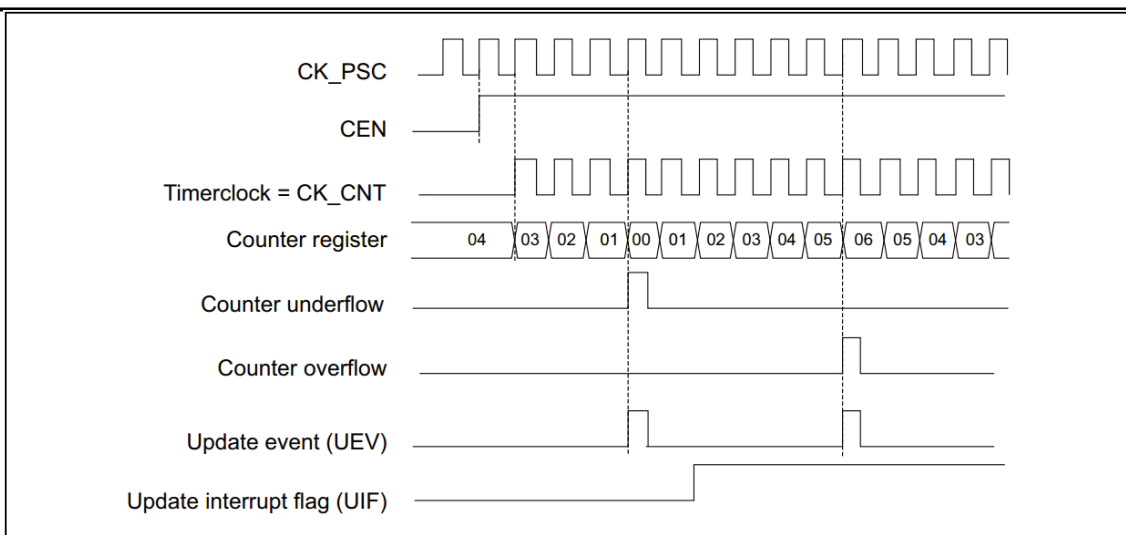


Figure62 Counter Timing Diagram with Internal Clock Division Factor of 1 and TIMx\_ARR=0x6  
1. Center alignment mode 1 is used here (see13.4.1 for details).

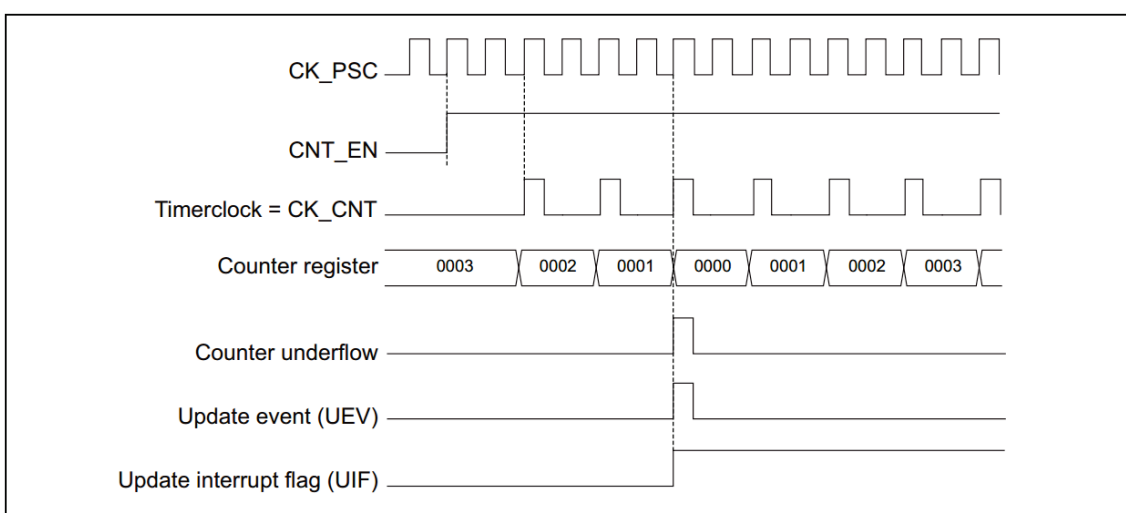


Figure63 Timing diagram of the counter with internal clock division factor of 2

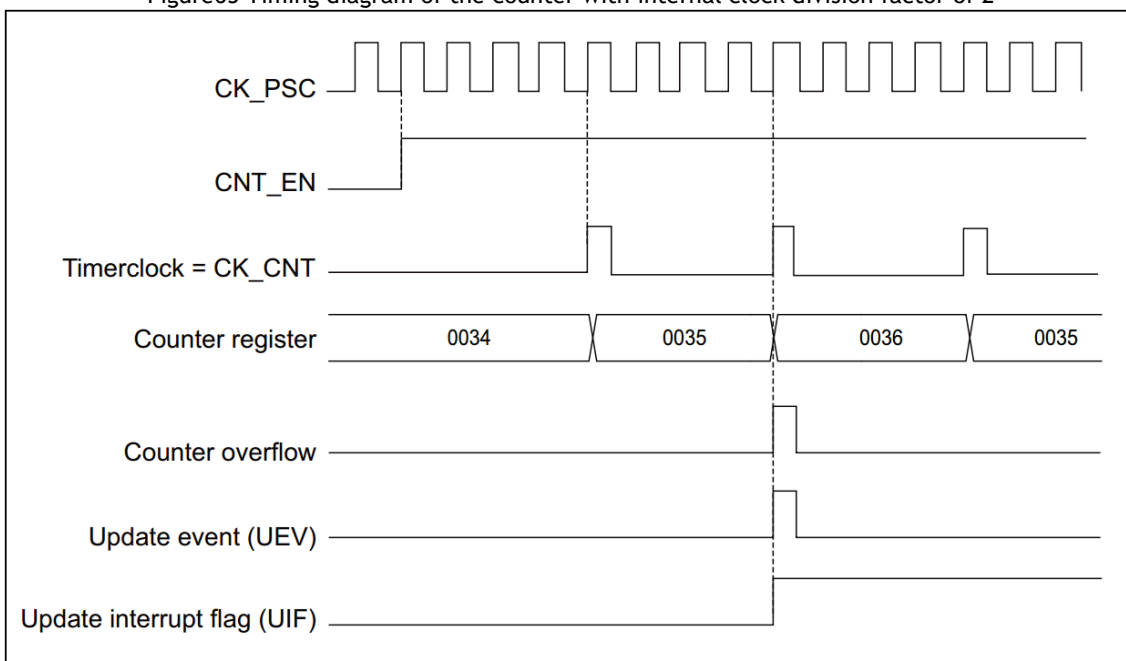


Figure64 Counter Timing Diagram with Internal Clock Division Factor of 4, TIMx\_ARR=0x36  
1. Center-aligned mode 2 or 3 is used with an UIF on overflow.

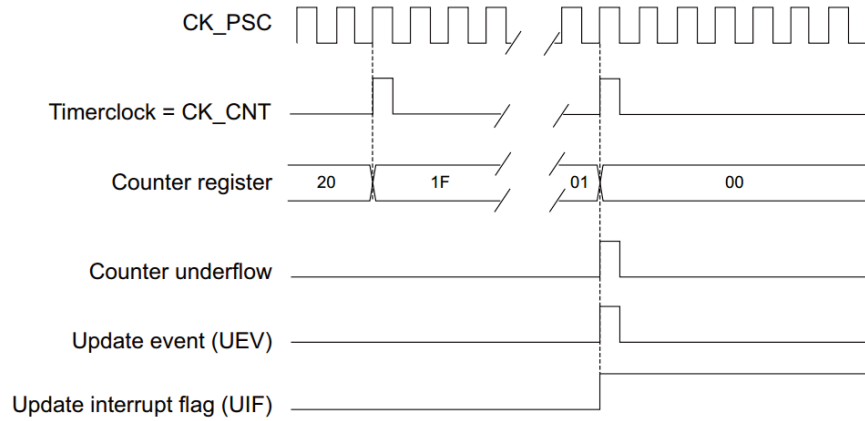


Figure65 Timing diagram of the counter with internal clock division factor N

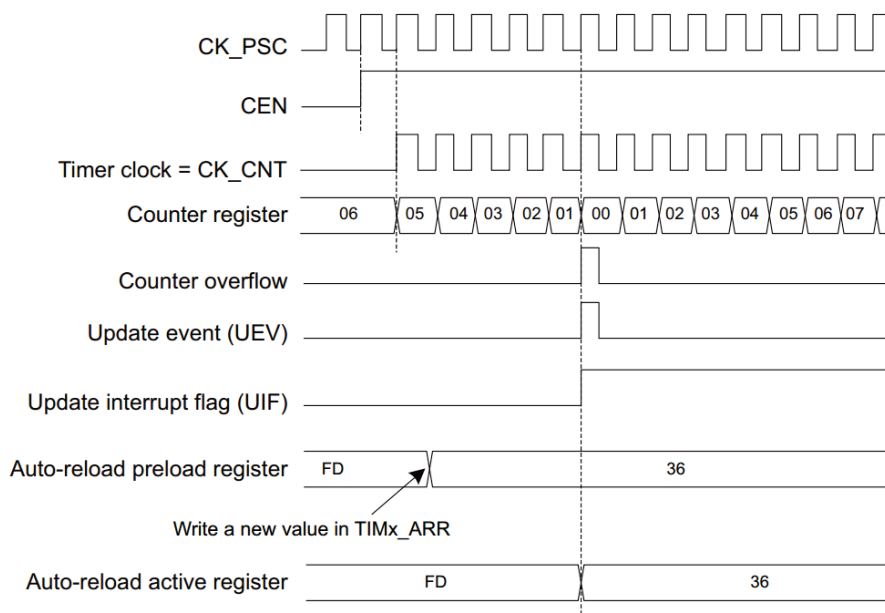


Figure66 Counter Timing Diagram, Update Event at ARPE=1 (Counter Underflow)

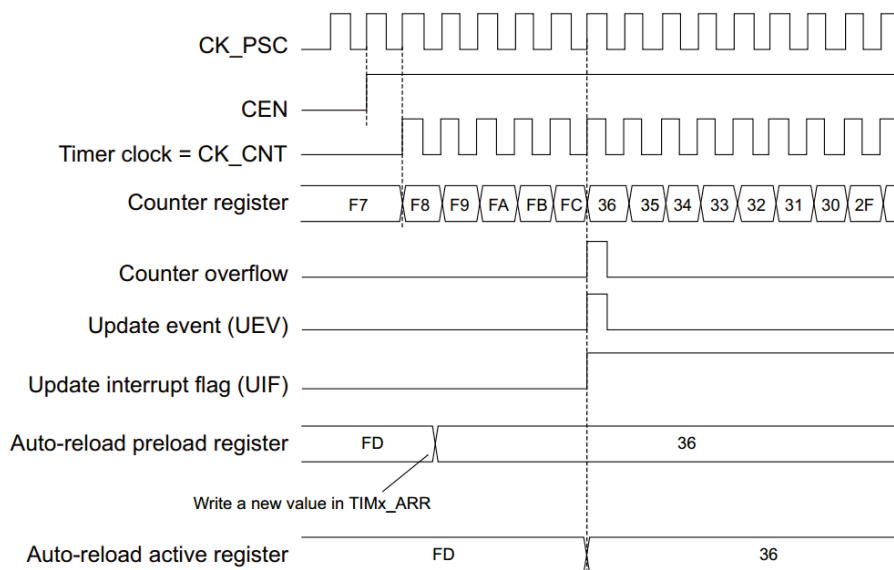


Figure67 Counter Timing Diagram, Update Event at ARPE=1 (Counter Overflow)

### 13.3.3 Repetition Counter

13.3.1 Section "Time Base Unit" explains how the Update Event on Counter Overflow/Underflow (UEV) is generated, but in fact it can only be generated when the repetition count reaches zero. This feature is very 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 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 upward count mode, the
- Each time the counter overflows in down count mode, the
- center-aligned mode at each overflow and at each underflow. Although this limits the maximum cycle period of the PWM to 128, it is able to update the duty cycle 2 times per PWM cycle. In centrally aligned mode, because the waveform is symmetrical, the maximum resolution is  $2xT_{ck}$  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 (refer to Figure 68). When an update event is generated by software (by setting the UG bit in TIMx\_EGR) or by a hardware slave mode controller, the update event occurs immediately and the contents of the TIMx\_RCR register are reloaded into the repeat counter, regardless of the value of the repeat counter.

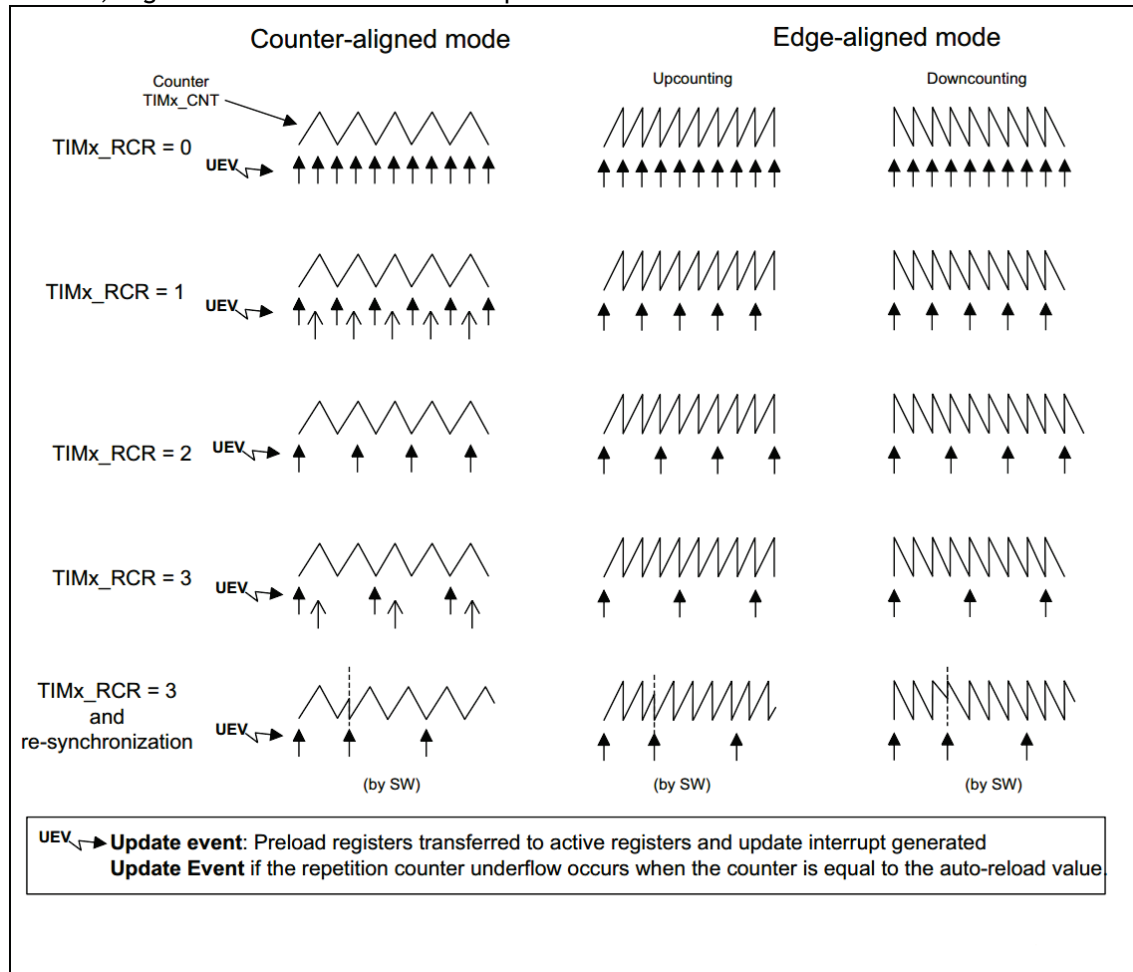


Figure 68 Examples of update rates in different modes, and register settings for TIMx\_RCR



### 13.3.4 Clock Selection

The counter clock can be provided 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, it is possible to configure one timer Timer1 to act as a prescaler for another timer Timer2. See the next chapter for details.

#### Internal Clock Source (CK\_INT)

If the slave mode controller is disabled (SMS=000), the CEN, DIR (TIMx\_CR1 register) and UG bits (TIMx\_EGR register) are de facto control bits and can only be modified by software (the UG bit is still cleared automatically). As long as the CEN bit is written to '1', the prescaler clock is provided by the internal clock CK\_INT.

The following figure shows the operation of the control circuit and up counter in the general mode without prescaler.

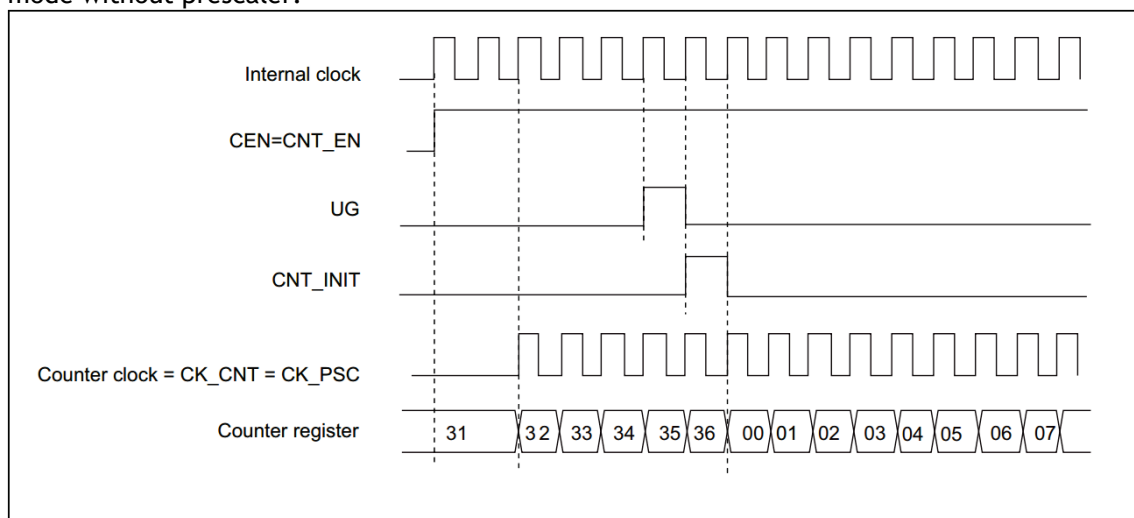


Figure69 Control circuit in general mode with internal clock division factor of 1

#### 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.

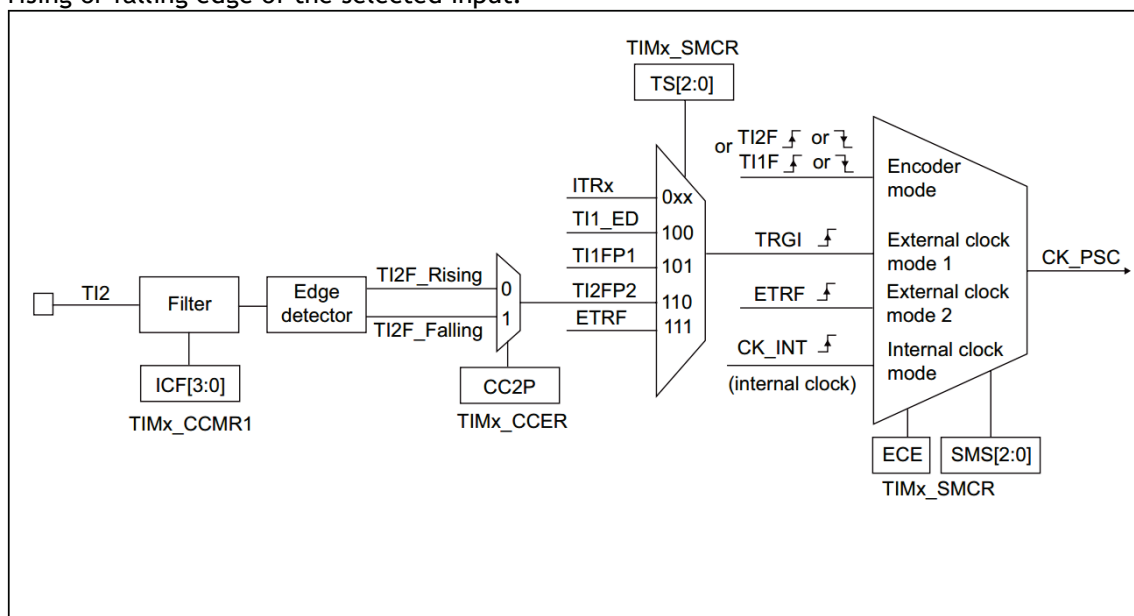


Figure70 TI2 External Clock Connection Example

For example, to configure the up counter to count on the rising edge of the T12 input, use the following steps: configure the TIMx\_CMR1 register CC2S=01 and configure channel 2 to detect the rising edge of the T12 input

Configure IC2F[3:0] of the TIMx\_CMR1 register to select the input filter bandwidth (if no filter is required, keep IC2F=0000)

Configure CC2P=0 in the TIMx\_CCER register to select the rising edge polarity

Configure SMS=111 in the TIMx\_SMCR register to select the timer external clock mode 1

Configure TS=110 in the TIMx\_SMCR register to select T12 as the trigger input source

Set CEN=1 in the TIMx\_CR1 register to start the counter

**Notes:** *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.

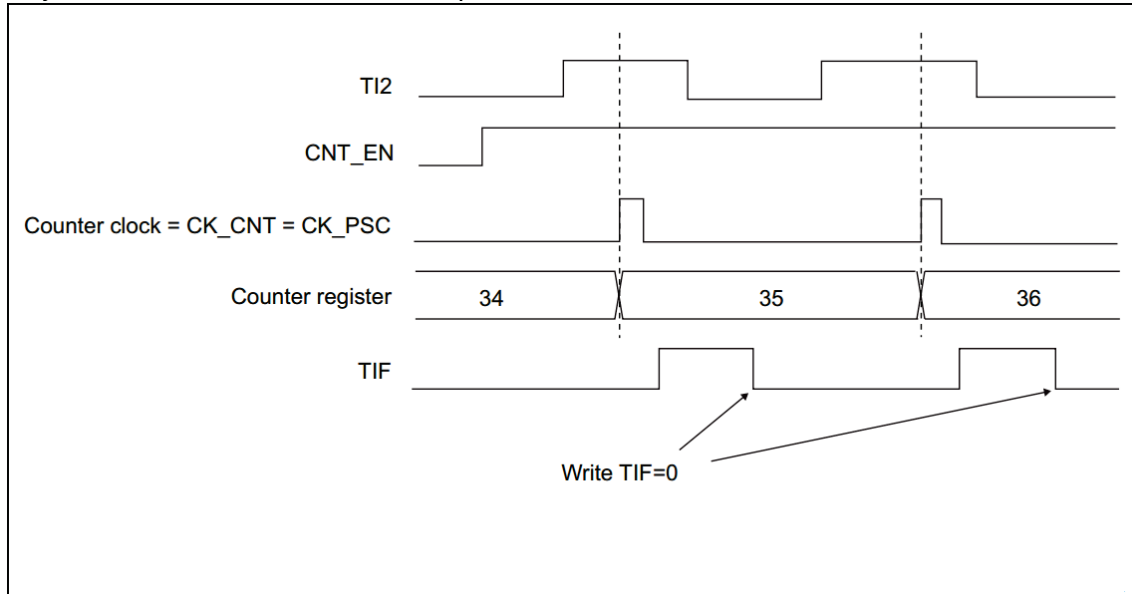


Figure71 Control circuit in external clock mode 1

## External Clock Source Mode 2

This mode is selected by making the ECE=1 counter in the TIMx\_SMCR register capable of counting on every rising or falling edge of an externally triggered ETR.

The following figure shows the block diagram of the external trigger input

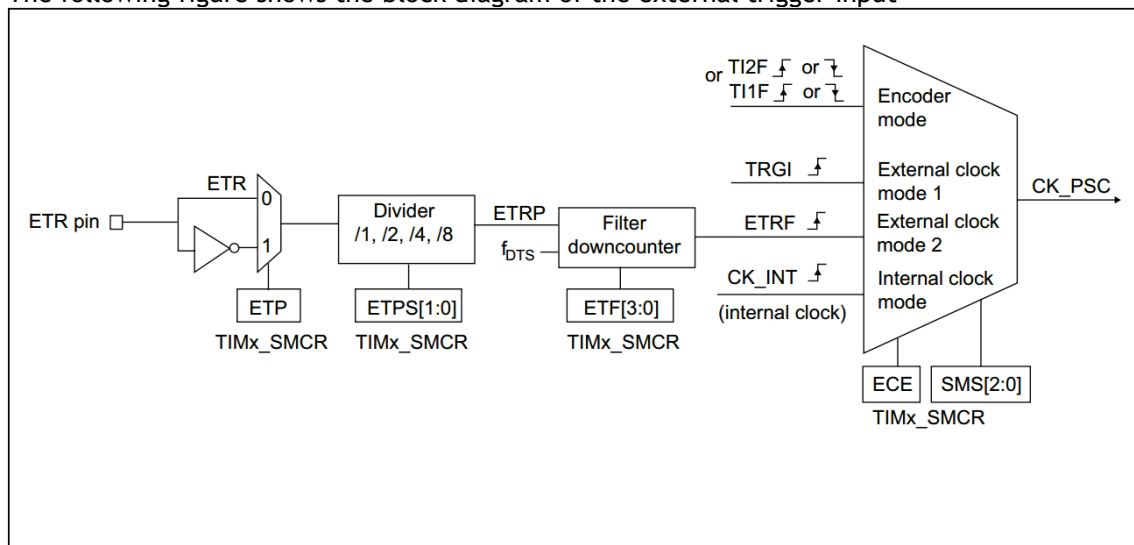


Figure72 External Trigger Input Block Diagram

For example, to configure an up counter that counts every 2 rising edges under ETR, use the following procedure:

4. No filter is needed in this example, set ETF[3:0]=0000 in TIMx\_SMCR register.

Set the prescaler, set ETPS[1:0]=01 in the TIMx\_SMCR registers

Select rising edge detection of ETR, set ETP=0 in TIMx\_SMCR register to turn on external clock mode 2, write ECE=1 in TIMx\_SMCR register

To start the counter, write CEN=1 counter in the TIMx\_CR1 register to count every 2 ETR rising edges.

The delay between the rising edge of ETR and the actual clock of the counter depends on the resynchronization circuitry at the ETRP signal end.

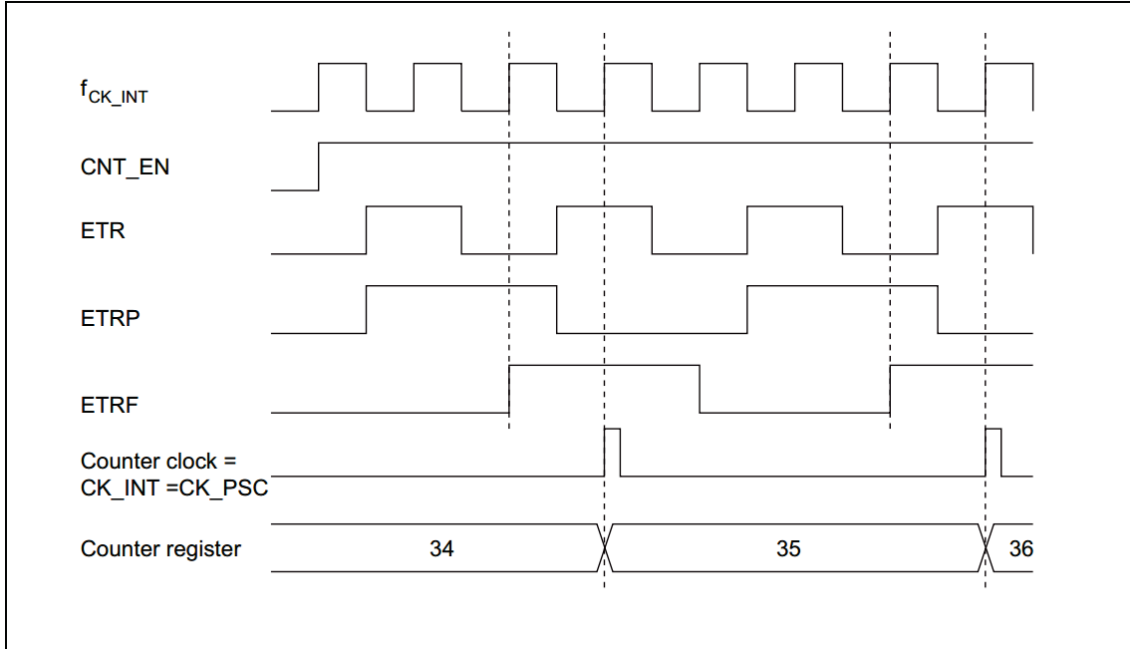


Figure73 Control Circuit in External Clock Mode 2

### 13.3.5 Capture/Compare Channel

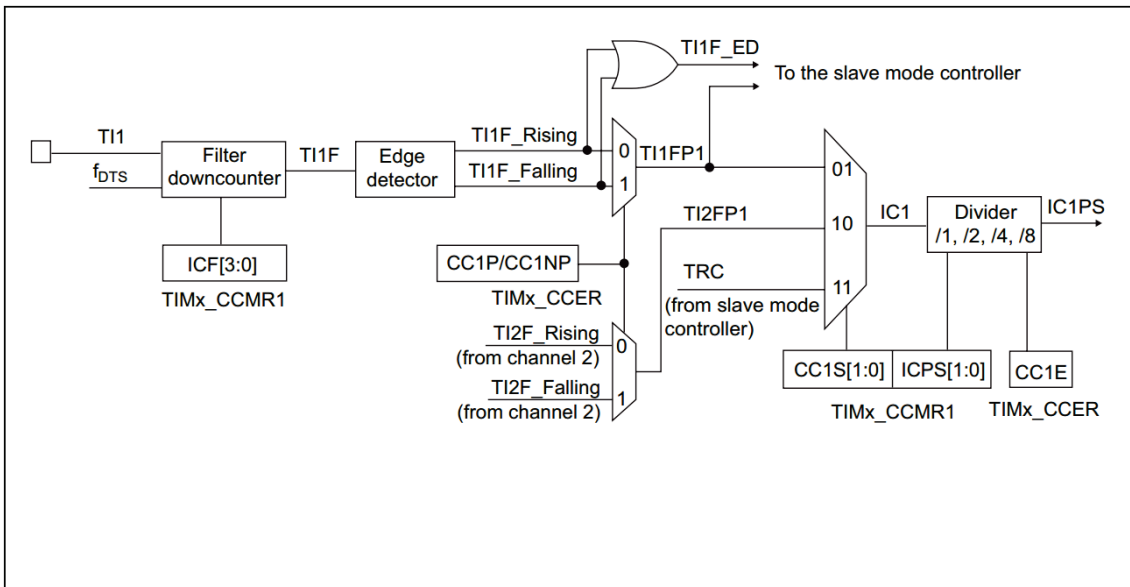


Figure74 Capture/compare channels (e.g. channel 1 input section)

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.

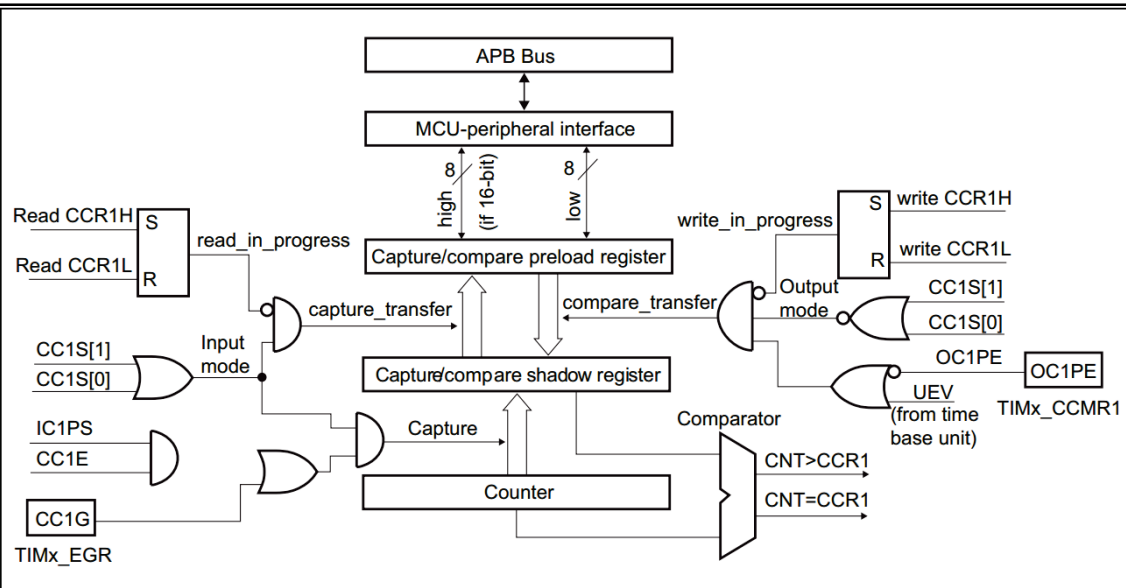


Figure75 Main circuit for capture/compare channel 1

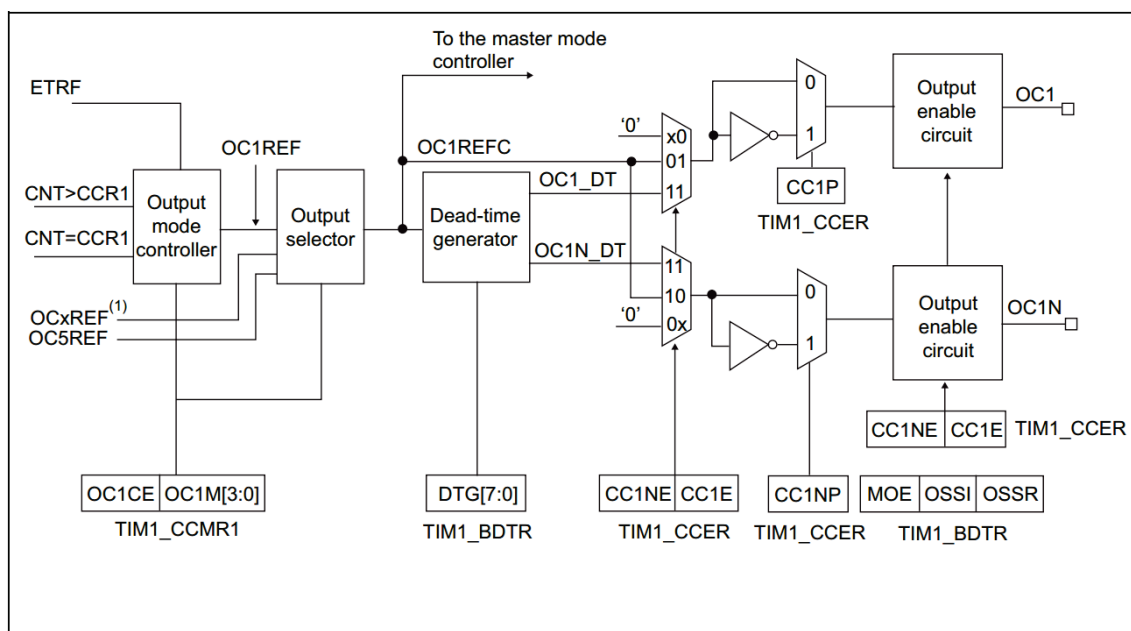


Figure76 Output section of capture/compare channels (channels 1 to 3)

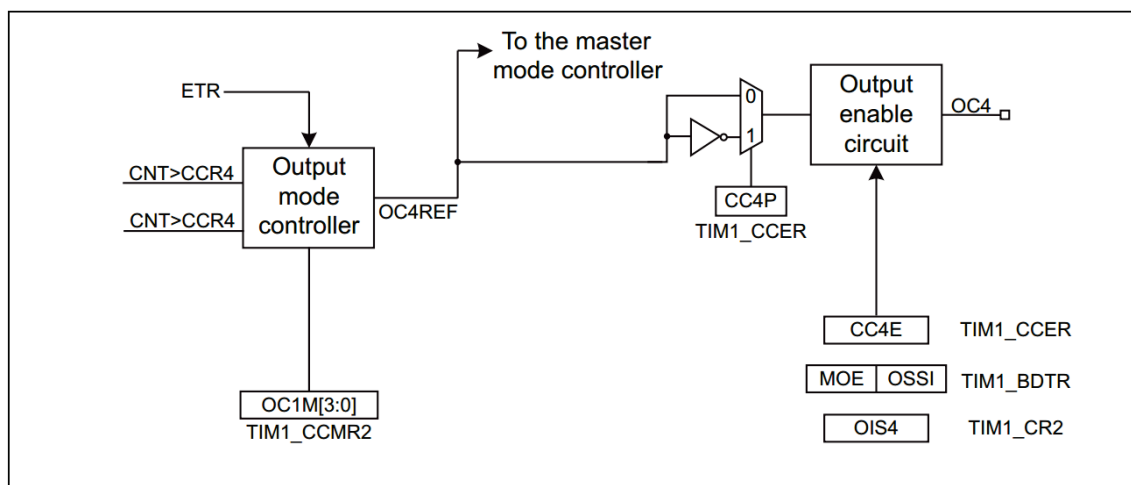


Figure77 Output section of the capture/compare channel (channel 4)

The capture/compare module consists of a preloaded register and a shadow register. The read/write process operates only on the preloaded register. In capture mode, the capture occurs on the shadow register, which is then copied to the preloaded register. 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.

### 13.3.6 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 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 an interrupt or DMA operation is open. 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 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 to the TIMx\_CCR1 register in the
- CC1S=01, 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 dithers over a period of up to 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 the input signal 8 times consecutively in order to confirm a true edge shift on TI1, i.e., by writing IC1F=0011 in the TIMx\_CCMR1 register.
- To select the active conversion edge of the TI1 channel, write 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 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 and CC1IF has not 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.

*Notes: Setting the corresponding CCxG bit in the TIMx\_EGR register allows you to generate input capture interrupts and/or DMA requests through software.*

### 13.3.7 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, you need to measure the length (TIMx\_CCR1 register) and duty cycle (TIMx\_CCR2 register) of the PWM signal input to TI1 as follows (depending on the frequency of CK\_INT and the value of the prescaler)

- To select the valid input of 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).
- 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.

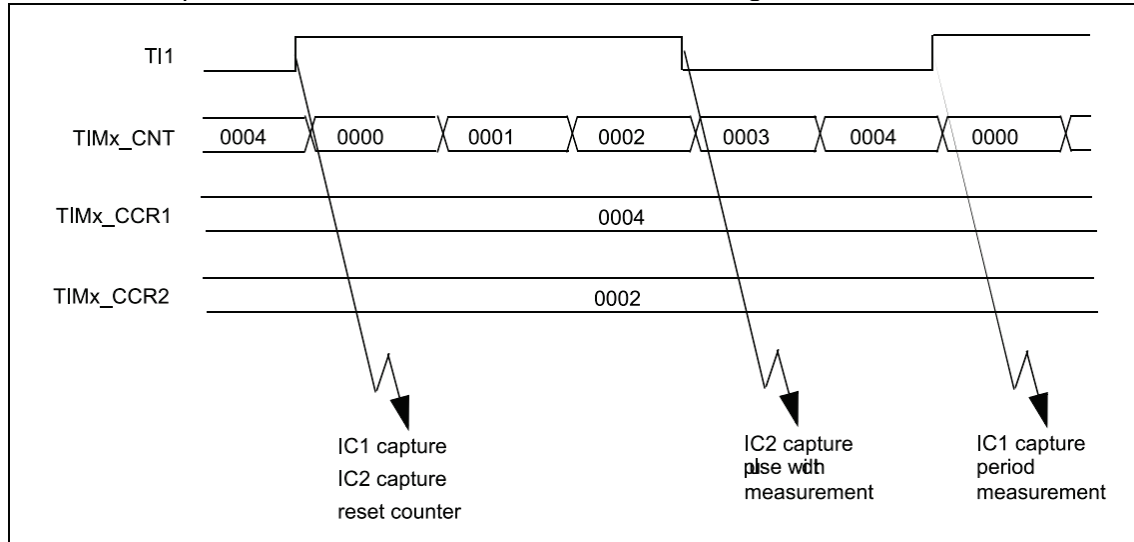


Figure 78 PWM Input Mode Timing

Since only TI1FP1 and TI2FP2 are connected to the slave mode controller, only the TIMx\_CH1/TIMx\_CH2 signals can be used for the PWM input mode.

### 13.3.8 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, independent of the result of the comparison between the Output Comparison Register and the counter.

The output compare signal (OCxREF/OCx) is forced active by setting the corresponding OCxM=101 in the TIMx\_CMRx register. In this way, OCxREF is forced high (OCxREF is always active high), while OCx gets the CCxP signal with opposite polarity.

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, comparisons between the TIMx\_CCRx shadow registers and counters are still performed and the corresponding flags are modified. Therefore corresponding interrupts and DMA requests are still generated. This will be described in the Output Compare Mode section below.

### 13.3.9 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. The output pin can hold its level while comparing the match (OCxM=000), is set to an active level (OCxM=001), is set to an inactive level (OCxM=010), or is flip-flopped (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 compare mode, the update event UEV has no effect on the OCxREF and OCx outputs.

The synchronization can be accurate to one counting cycle of the counter. The 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. To generate an interrupt request, set the CCxIE bit.
4. Select the output mode, for example:
  - Requires the counter to flip the output pin of OCx when it matches CCRx, set OCxM=011
  - Set OCxPE=0 to disable the preload registers.
  - Set CCxP=0 to select polarity as active high
  - Set CCxE=1 to enable the output.
5. Setting the CEN bit of the TIMx\_CR1 register starts 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 following figure.

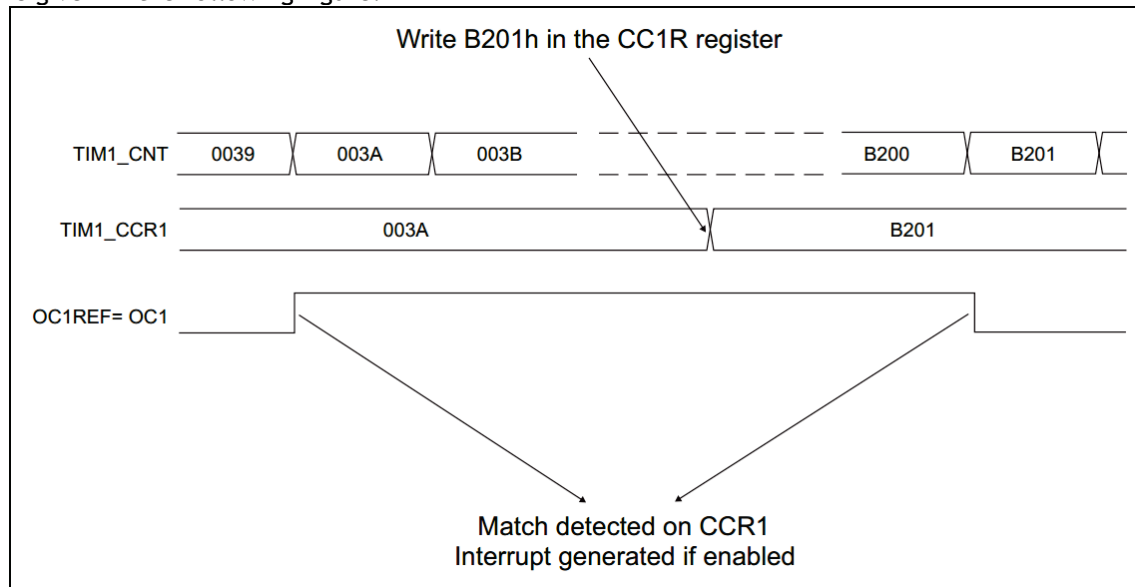


Figure79 Output Compare Mode, Flip OC1

### 13.3.10 PWM mode

Pulse width modulation mode can generate 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 PWM. the corresponding preload register must be enabled by setting the OCxPE bit of the TIMx\_CMRx register, and finally the TIMx\_CR1 register. Finally, the ARPE bit of the TIMx\_CR1 register must also be set to enable the auto-reload preload register (in count-up or center-symmetric mode).

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 count direction of the counter) to determine compliance with  $TIMx\_CCRx \leq TIMx\_CNT$  or  $TIMx\_CNT \leq TIMx\_CCRx$ .

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. Refer to section 13.3.2.

The following is an example of PWM mode 1. The PWM reference signal OCxREF is high when  $TIMx\_CNT < TIMx\_CCRx$  and low otherwise. If the comparison value in TIMx\_CCRx is greater than the auto-reload value (TIMx\_ARR), OCxREF remains '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.

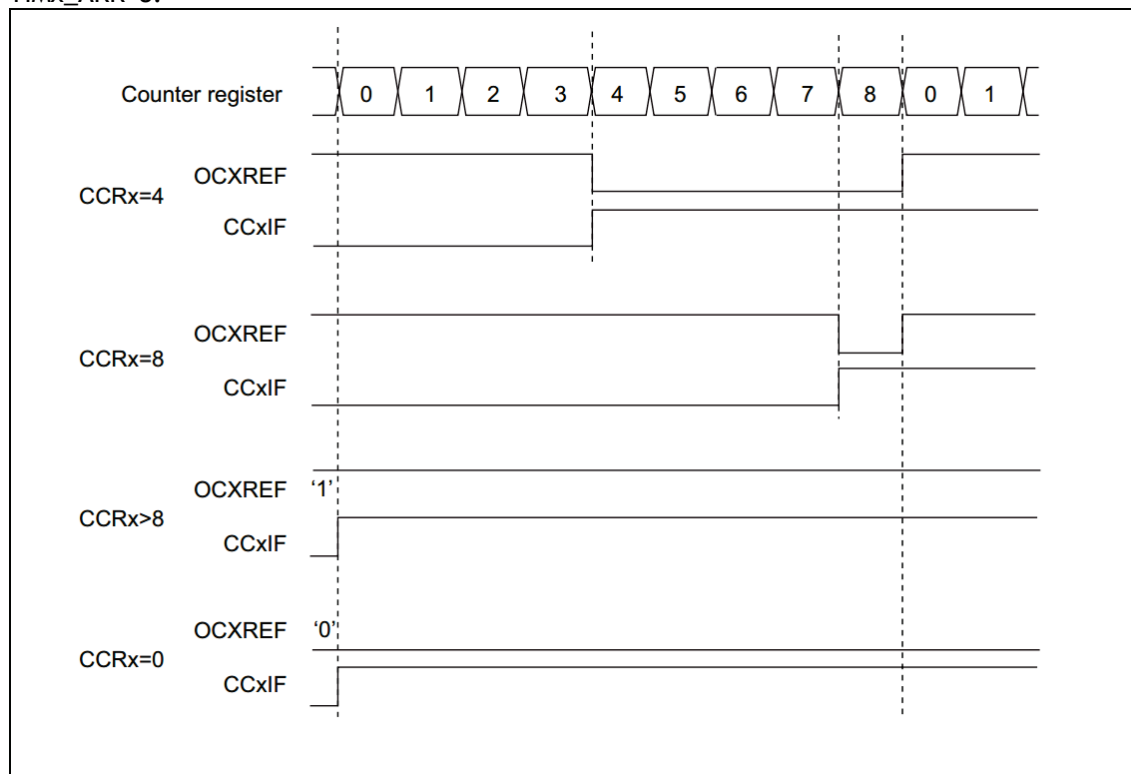


Figure 80 Edge-aligned PWM waveform (ARR=8)

### ● Down Count Configuration

Performs a down count when the DIR bit of the TIMx\_CR1 register is high. Refer to section 13.3.2. In PWM mode 1, the reference signal OCxREF is low when  $TIMx\_CNT > TIMx\_CCRx$ , otherwise it is high. The reference signal OCxREF is high if

The comparison value in TIMx\_CCRx is greater than the auto-reload value in TIMx\_ARR, then OCxREF is held at '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 can be set to 1 when the counter is counting up, 1 when the counter is counting down, or 1 when the counter is counting both up and down. The count direction bit (DIR) in the TIMx\_CR1 register is updated by hardware; do not modify it with software. Refer to the 13.3.2 section for Central Alignment Mode.

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 1.



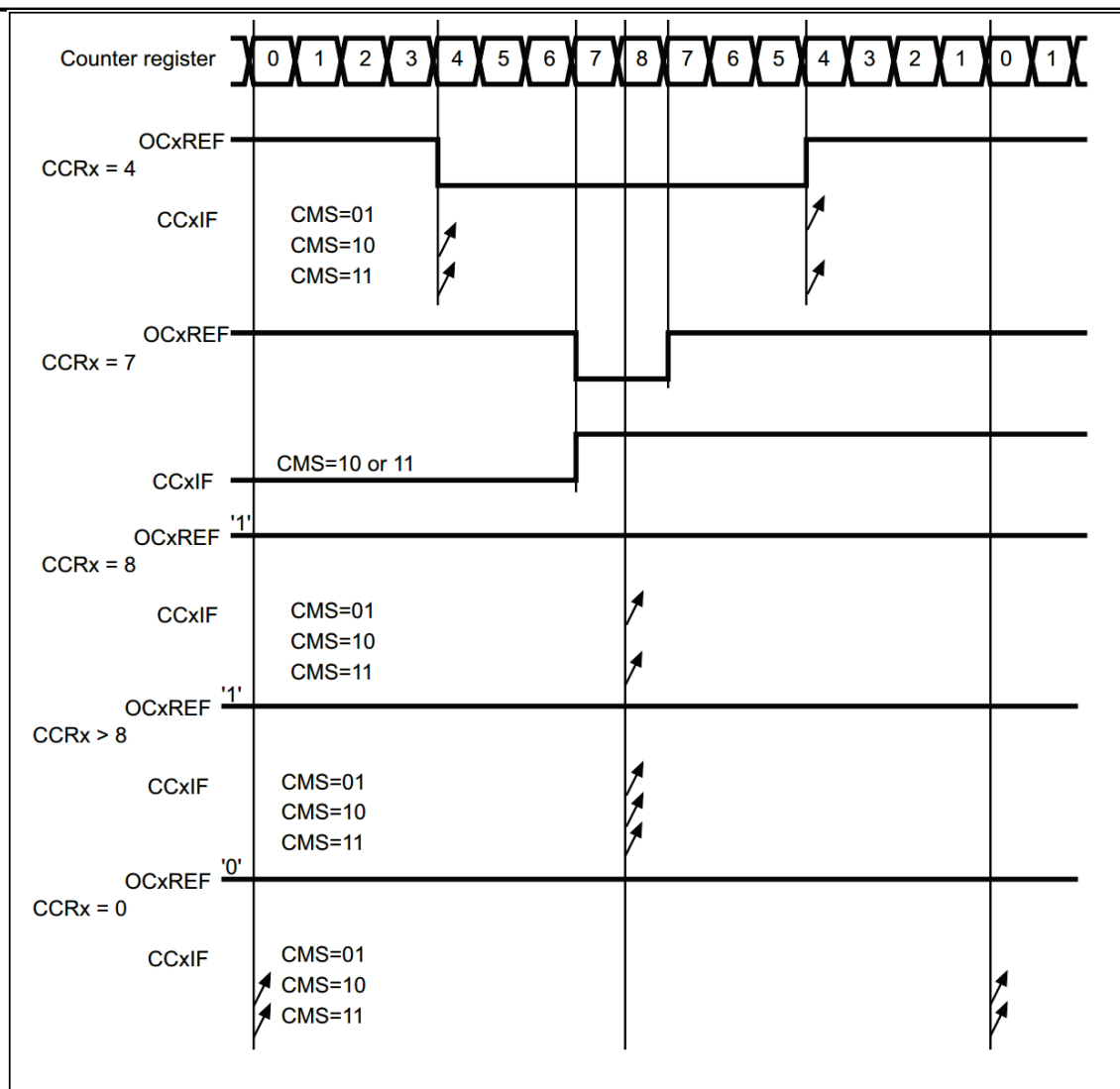


Figure81 Centrally aligned PWM waveform (APR=8)

#### 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 TIM<sub>x</sub>\_CR1 register. In addition, software cannot modify the DIR and CMS bits at the same time.
- It is not recommended to rewrite 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 (TIM<sub>x</sub>\_CNT>TIM<sub>x</sub>\_ARR), the direction is not updated. For example, if the counter is counting up, it continues to count up.
  - If a value of 0 or TIM<sub>x</sub>\_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 TIM<sub>x</sub>\_EGR bit) before starting the counter and not to modify the counter value while the count is in progress.

### 13.3.11 Complementary Outputs and Deadband Insertion

The advanced control timers (TIM1 and TIM8) are capable of outputting two complementary signals and managing 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 shift delay, power switch delay, 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, OSSI, and OSSR bits in the TIMx\_BDTR and TIMx\_CR2 registers, as detailed in the

Table73 Control Bits for Complementary Output Channels OCx and OCxN with Brake. In particular, the deadband is activated on transition to the IDLE state (MOE falling to 0).

Setting both the CCxE and CCxNE bits will insert a deadband, and the MOE bit will also be set if a brake circuit is present. Each channel has a 10-bit deadband generator. The reference signal OCxREF generates 2 outputs OCx and OCxN. if OCx and OCxN are active high:

- The OCx output signal is the same as the reference signal except 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 has a delay 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)

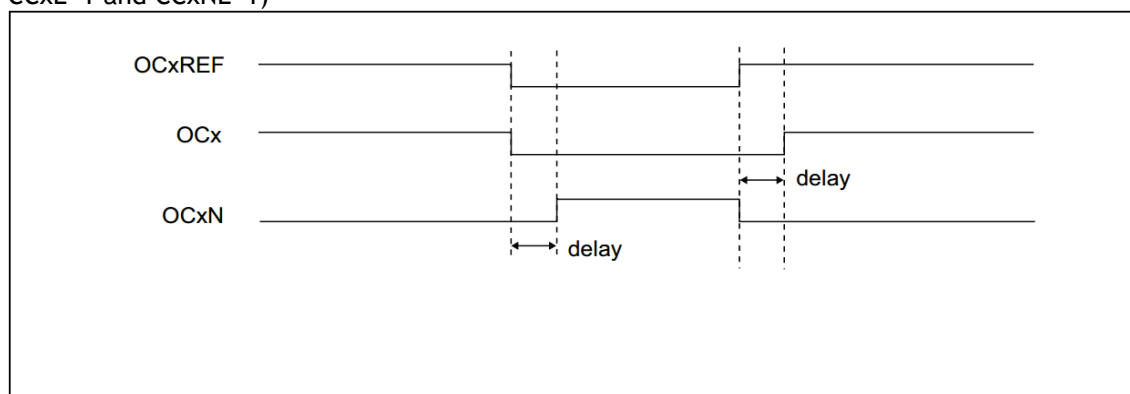


Figure82 Complementary output with deadband insertion

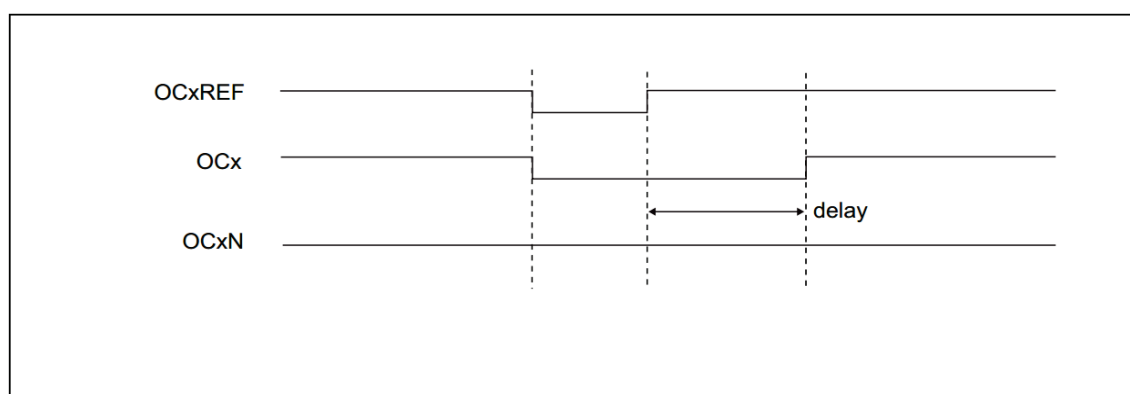


Figure83 Deadband waveform delay greater than negative pulse

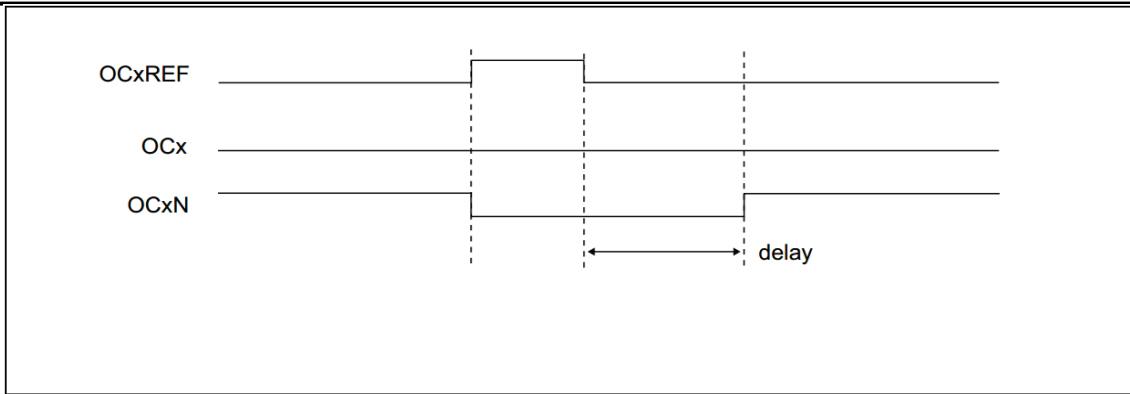


Figure84 Deadband waveform delay greater than positive pulse

The deadband delay is the same for each channel and is programmatically configured by the DTG bit in the TIMx\_BDTR register. For details, see  
 0Sections TIM1 and TIM8 brake and deadband register (TIMx\_BDTR)delay calculations in the .  
 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 outputs are 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.

*Notes: When only OCxN is enabled (CCxE=0,CCxNE=1), it is not inverted and goes high immediately when OCxREF is active. For example, OCxN=OCxREF if CCxNP=0. On the other hand, when both OCx and OCxN are enabled (CCxE=CCxNE=1), OCx is valid when OCxREF is high; and OCxN, on the contrary, becomes valid when OCxREF is low.*

### 13.3.12 Using the Brake Function

When the brake function is used, based on the corresponding control bits (MOE, OSS1, and OSSR bits in the TIMx\_BDTR register, and the OISx and OISxN bits in the TIMx\_CR2 register), the output enable signal and the invalid level are modified. However, the OCx and OCxN outputs cannot be on active levels at the same time at the same time. For details, see

Table73 Complementary Output Channels with BrakeControl Bits for OCx and OCxN .

The brake source can be either a brake input pin or a clock fail event. The clock failure event is generated by the clock safety system in the reset clock controller, as detailed in section Clock Safety System (CSS)6.2.7.

After 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, so it is necessary to wait for one APB clock cycle before the written bits can be 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 signal and the synchronization signal. 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 the write is to an asynchronous signal and the read is to a synchronous signal.

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, idle, or reset state (selected by the OSS1 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 OSS1=0, the timer releases the enable outputs, otherwise the enable outputs are always high.
- When using complementary outputs:
  - The output is first placed in a reset state i.e. an invalid state (depending on 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 levels 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 OSS1=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 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 at the next update event UEV; this can be used for shaping, for example. Otherwise, MOE always remains low until it is set '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.

*Notes: The brake input is level active. Therefore, MOE cannot be set at the same time (automatically or via software) when the brake input is active. At the same time, the status flag BIF cannot be cleared.*

The brake is generated by the BRK input, which has a programmable active polarity and is turned on by the BKE bit in the TIMx\_BDTR register.

In addition to brake input and output management, write protection is implemented in the brake circuit to secure the application. It allows the user to freeze several configuration parameters (deadband length, OCx/OCxN polarity and disabled state, 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, refer to 0 section TIM1 and TIM8 Brake and Deadband Registers (TIMx\_BDTR). The LOCK bit can only be modified once after an MCU reset.

The following figure shows an example of the output of the response brake.

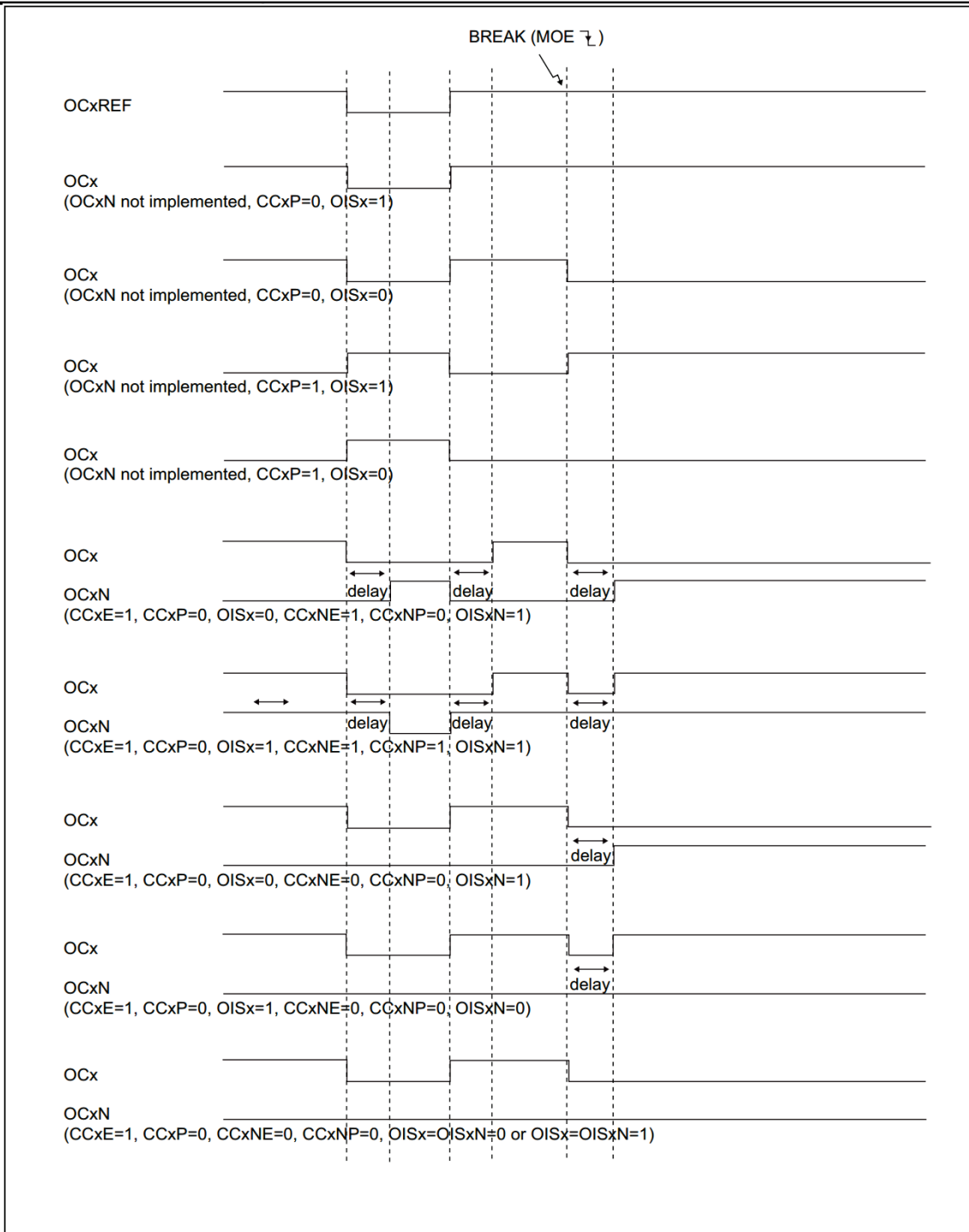


Figure85 Outputs in response to braking

### 13.3.13 Clearing the OCxREF Signal on an External Event

For a given channel, setting the corresponding OCxCE bit in the TIMx\_CMRx register to '1' is able to pull the OCxREF signal low with a high level on the ETRF input, and the OCxREF signal will remain low until the next update event, UEV, occurs.

This function can only be used in the output comparison and PWM modes, not in the strong-set mode.

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. The 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, the timer TIMx is placed in PWM mode.

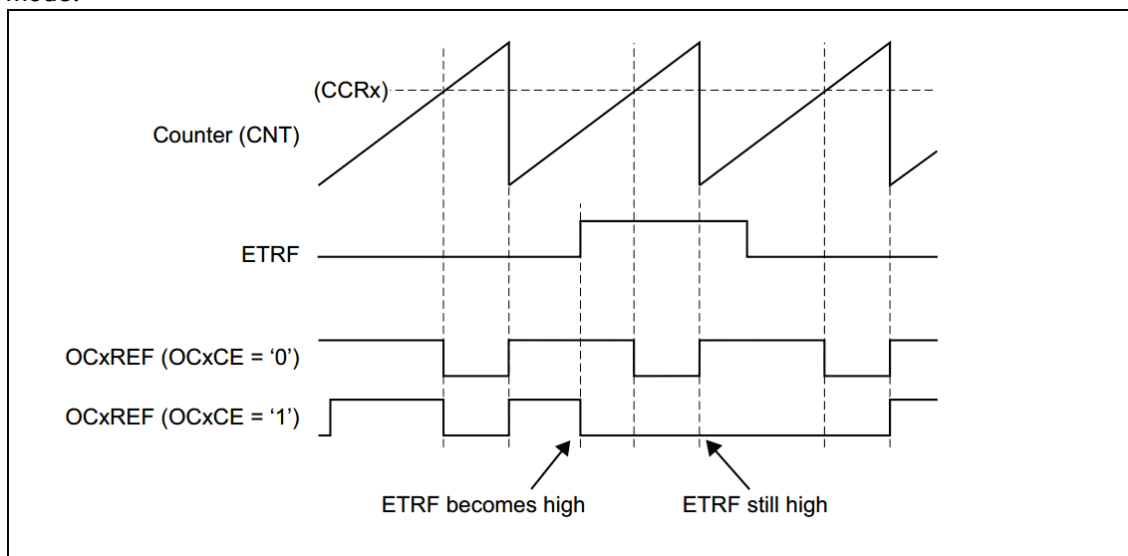


Figure86 Clear OCxREF for TIMx

### 13.3.14 Generates a Six-step PWM Output

When complementary outputs are required on a channel, the preload bits are OCxM, CCxE, and CCxNE. These preload bits are transferred to the shadow register bits when a COM phase change event occurs. This way you can pre-set the next step configuration and fix changes to all channels at the same time at the same moment. COM can be generated by software by setting the COM bit in the TIMx\_EGR register, or by hardware at 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.

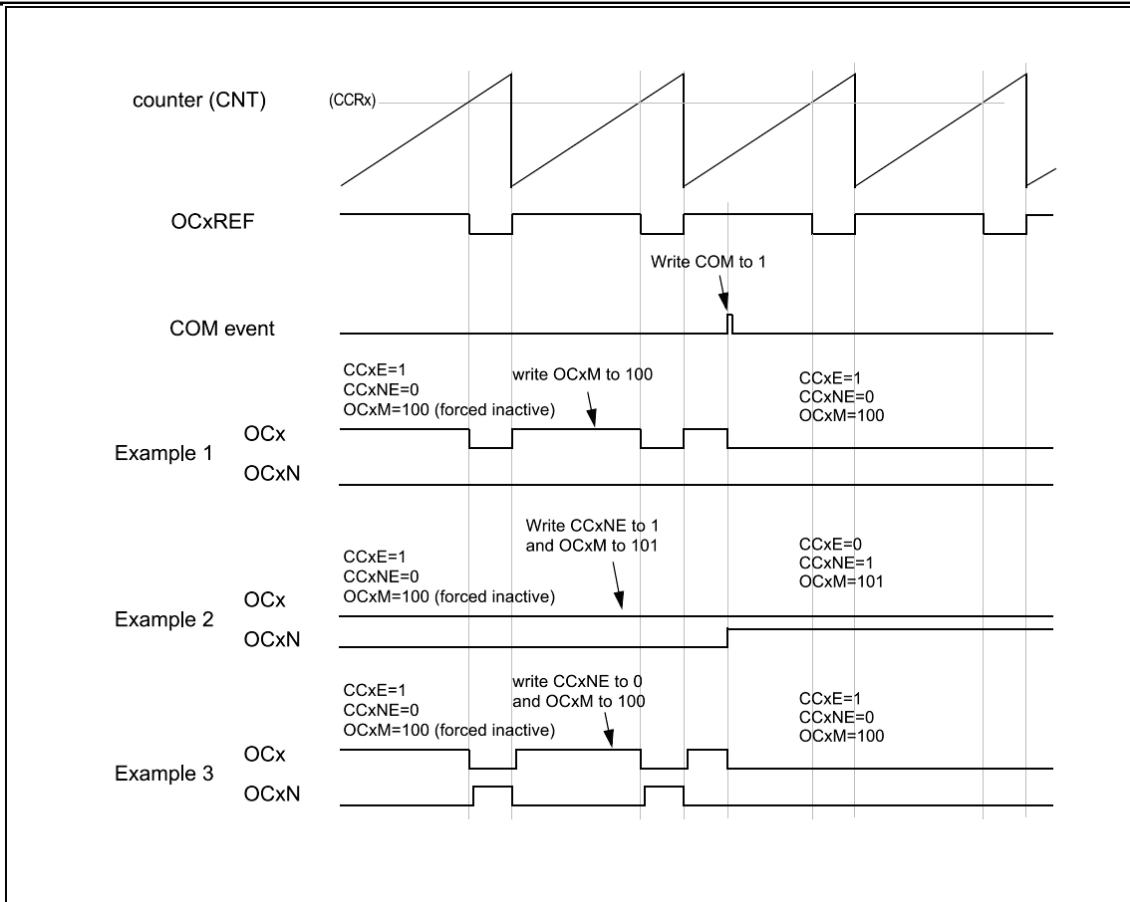


Figure87 Example of generating a six-step PWM, using COM (OSSR=1)

### 13.3.15 Single Pulse Mode

One-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 generate a pulse with a programmable pulse width after a programmable delay.

The counter can be started from the mode controller to generate a waveform in either output compare 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  $CNT < CCRx \leq ARR$  (specifically,  $0 < CCRx$ ).
- Downward counting mode: counter  $CNT > CCRx$ .

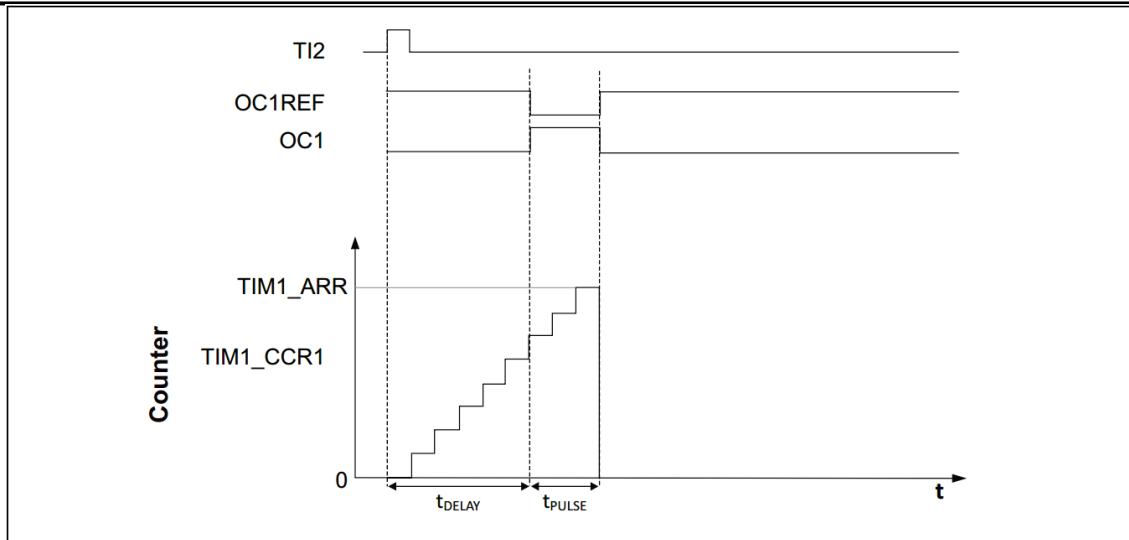


Figure88 Example of single pulse mode

For example, you need to generate a long on OC1 after delaying  $t_{\text{DELAY}}$  starting from a rising edge detected on the TI2 input pin positive pulse of degree  $t_{\text{PULSE}}$ .

Assuming 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 in the TIMx\_SMCR register (trigger mode), the TI2FP2 is used to start the counter. the waveform of the OPM is determined by the value written to the comparison register (taking into account the clock frequency and the counter prescaler)
- $t_{\text{DELAY}}$  is defined by the value in the TIMx\_CCR1 register.
- $t_{\text{PULSE}}$  is defined by the difference between the auto-load 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; first 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 by filling in the auto-load value in TIMx\_ARR register. then fill in the comparison value in the TIMx\_CCR1 register and the auto-load 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 required, 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-load 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. Comparison operations between the counter and the comparison value then produce the conversion of the output. However, these operations require a certain number of clock cycles, so it limits the minimum delay  $t_{\text{DELAY}}$  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 when the comparison is matched. OCxFE works only when the channel is configured for PWM1 and PWM2 modes.



### 13.3.16 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 Table71, assuming that the counter has been activated (CEN=1 in the TIMx\_CR1 register), the counter is driven by each valid hop on either TI1FP1 or TI2FP2. TI1FP1 and TI2FP2 are the signals of TI1 and TI2 after passing through the input filters and the polarity control; if there is no filtering and no phasing, then TI1FP1=TI1, TI2FP2=TI2. TI2FP2 = TI2. Based on the hopping order of the two input signals, count pulses and direction signals are generated. Depending on the hopping sequence 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 counts on TI1, on TI2, or on both TI1 and TI2, 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-loaded value of the TIMx\_ARR register (depending on the direction, either 0 to ARR counts or ARR to 0 counts). So TIMx\_ARR must be configured before counting starts; again, the capture, comparator, prescaler, repeat counter, trigger output characteristics, etc. still work as usual. Encoder mode and External Clock Mode 2 are not compatible and therefore cannot be operated at the same time.

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 shows all possible combinations, assuming that TI1 and TI2 are not changing at the same time.

Table71 Relationship between count direction and encoder signal

active edge	Relative signal level (TI1FP1 corresponds to TI2, TI2FP2 corresponds to TI1)	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 the need for 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 immunity to noise interference. The third signal from the encoder output represents a mechanical zero, which can be connected to an external interrupt input and trigger a counter reset.

The figure below is an example of counter operation, showing the generation of the counting signal 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, IC1FP1 mapped to TI1)
- CC2S='01' (TIMx\_CMR2 register, IC2FP2 mapped to TI2)
- CC1P='0' (TIMx\_CCER register, IC1FP1 not inverted, IC1FP1=TI1)
- CC2P='0' (TIMx\_CCER register, IC2FP2 not inverted, IC2FP2=TI2)
- SMS='011' (TIMx\_SMCR register, all inputs are valid on rising and falling edges).
- CEN='1' (TIMx\_CR1 register, counter enable)

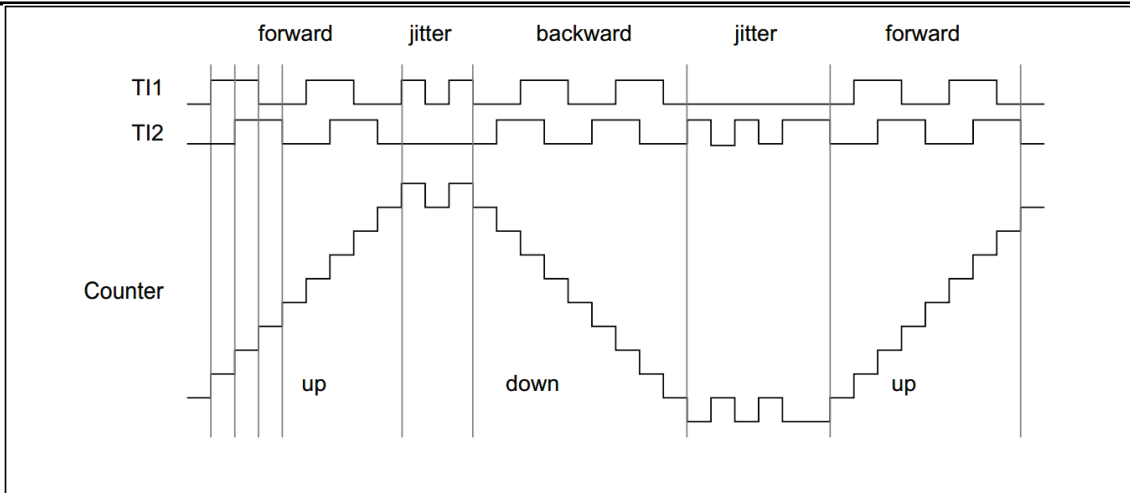


Figure89 Example of counter operation in encoder mode

The following figure shows an example of counter operation when IC1FP1 polarity is inverted (CC1P='1', other configurations are the same as the above example)

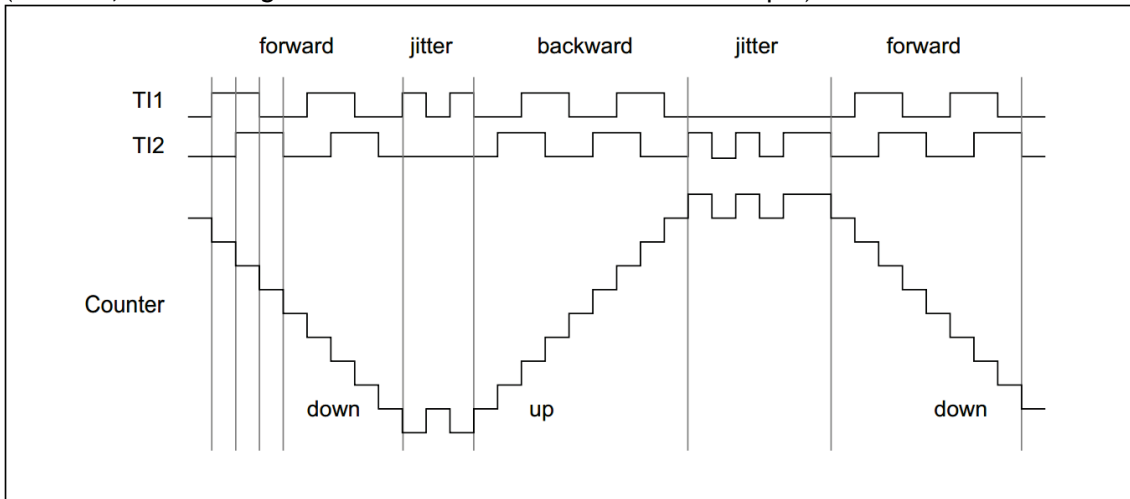


Figure90 Example of encoder interface mode for IC1FP1 inversion

When the timer is configured in encoder interface mode, it provides information about the current position of the sensor. Using a second timer configured in capture mode, the interval between two encoder events can be measured to obtain information about the dynamics (velocity, acceleration, deceleration). An 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, you can latch the value of the counter to a third input capture register (the capture signal must be periodic and can be generated by another timer); or you can read its value through a DMA request generated by the real-time clock.

### 13.3.17 Timer Input Heterodyne Function

The TI1S bit in 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 heterodyne output can be used for all timer input functions such as triggering or input capture. The following section 13.3.18 gives an example of this feature being used to connect a Hall sensor.

### 13.3.18 Interfacing with Hall Sensors

When using an advanced control timer (TIM1 or TIM8) to generate a PWM signal to drive the motor, another general-purpose TIMx (TIM2, TIM3, TIM4, or TIM5) timer can be used as an "interface timer" to connect to the Hall sensor, see Figure 91, with the three timer inputs (CC1, CC2, CC3) connected to the TI1 input channel through an isochronous gate (selected by setting the TI1S bit in the TIMx\_CR2 register). The three timer inputs (CC1, CC2, CC3) are connected to the TI1 input channel through an all-or-nothing 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 in 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 inputs.

Capture/compare channel 1 on the "Interface Timer" is configured in capture mode and the capture signal is TRC (see Figure 74). The capture value reflects the time delay between two input changes and gives information about the motor speed.

The "Interface Timer" can be used to generate a pulse in the output mode, which can be used (by triggering a COM event) to change the attributes of the individual channels of the Advanced Timer TIM1 or TIM8, which generates the PWM signal to drive the motor. The "Interface Timer" channel must therefore be programmed to generate a positive pulse after a specified delay (output comparison or PWM mode), which is sent to the Advanced Control Timer TIM1 or TIM8 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.

- Setting the TI1S bit of the TIMx\_CR2 register to '1' configures the three timer inputs to logic-or to the TI1 input, the
- 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 the PWM signal, and the capture/compare control signal is pre-loaded (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.

The following figure shows an example of this

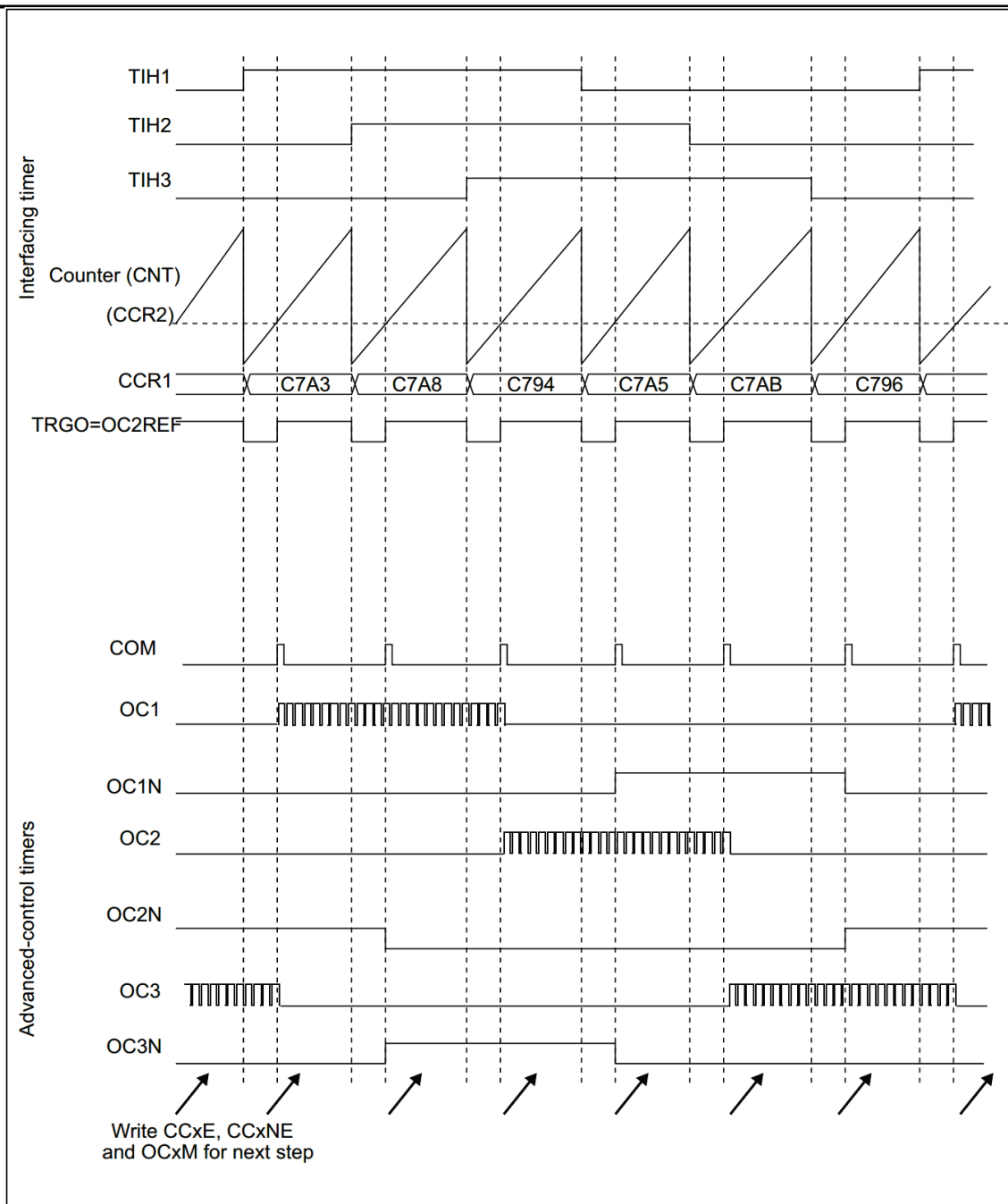


Figure91 Example of Hall sensor interface

### 13.3.19 Synchronization of TIMx 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 preloaded 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 the 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 based on the internal clock and then operates normally until a rising edge occurs at 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.

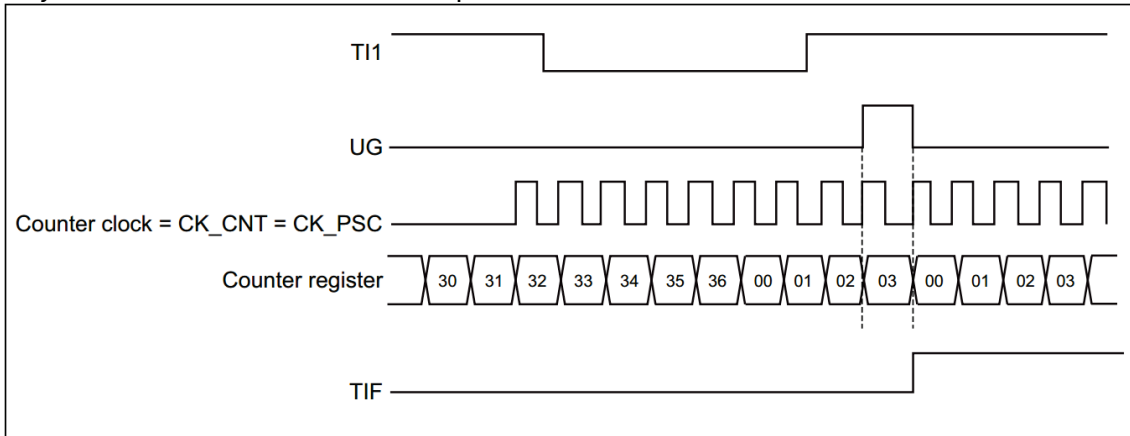


Figure92 Control circuit in reset mode

#### 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, set CC1S=01 in the TIMx\_CMR1 register. Set CC1P=1 in the TIMx\_CCER register to determine polarity (detect lows 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 be started, regardless of the trigger input level.

The counter starts counting according to the internal clock as long as TI1 is low, and stops counting once TI1 goes high. The TIF marker in TIMx\_SR is set when 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.

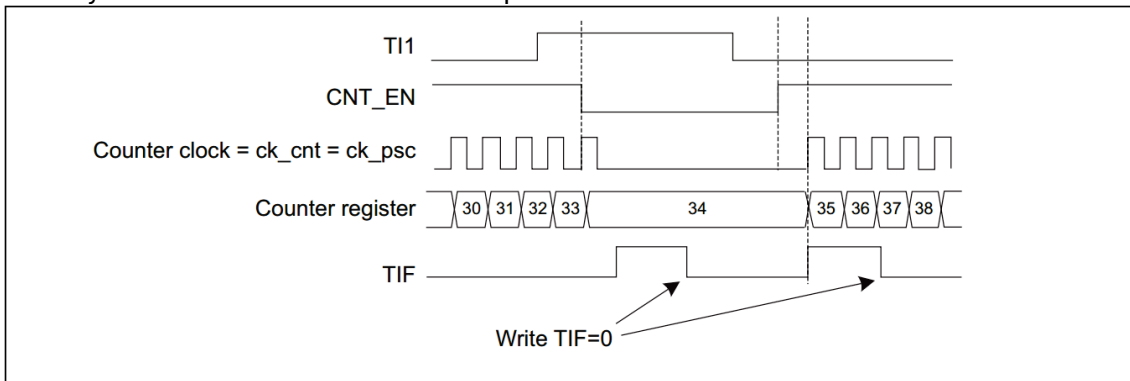


Figure93 Control circuit in gated mode

### 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 need to be configured. the CC2S bit is only used to select the input capture source, set CC2S=01 in the TIMx\_CMR1 register. set CC2P=1 in the TIMx\_CCER register to determine the 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 while the TIF flag is set. The delay between the rising edge of TI2 and the counter starting counting depends on the resynchronization circuit at the TI2 input.

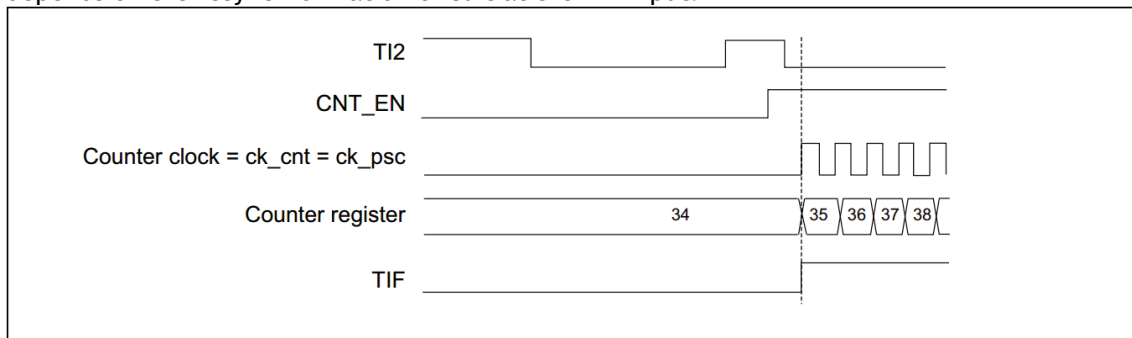


Figure94 Control Circuit in Trigger Mode

### 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 an input to the external clock, and another input can be selected as a 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 once on each rising edge of the ETR:

1. 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.
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 require configuration of the
  - 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 the rising edge)
3. Set SMS=110 in the TIMx\_SMCR register to configure the timer for trigger mode. Set TS=101 in TIMx\_SMCR register to select TI1 as 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.

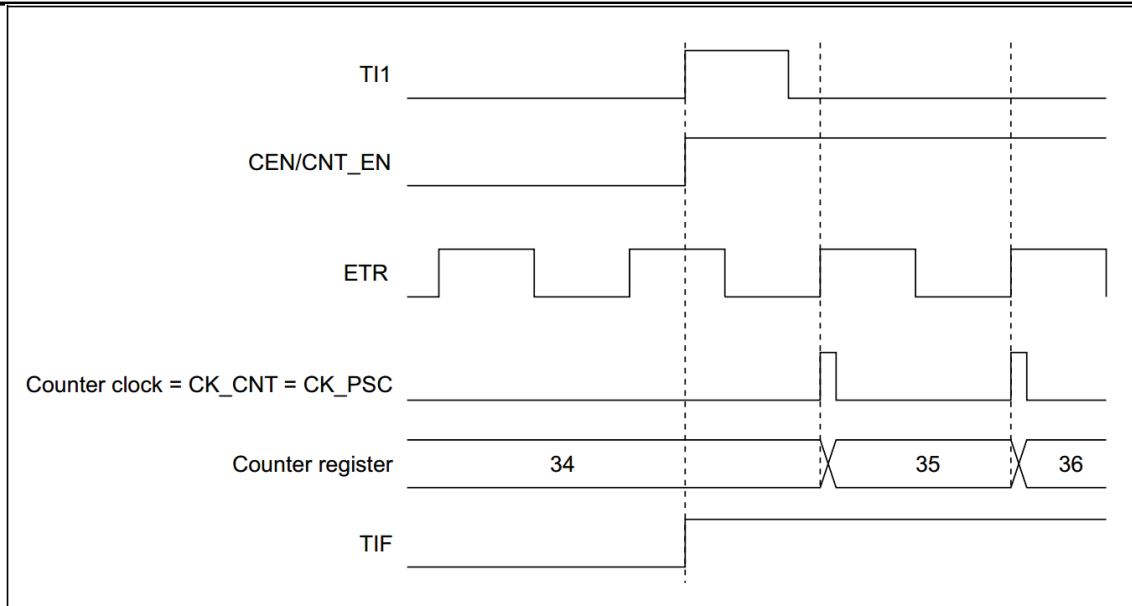


Figure95 Control Circuit in External Clock Mode 2 + Trigger Mode

### 13.3.20 Timer Synchronization

All TIM timers are connected internally for timer synchronization or linking. See the next chapter at 14.3.15 for more details.

### 13.3.21 Debug Mode

When the microcontroller enters debug mode (the Cortex-M3 core is stopped), the TIMx counter can either continue normal operation or stop, depending on the DBG\_TIMx\_STOP setting in the DBG module. See the subsequent 31.16.2 section for details.

## 13.4 TIM1 and TIM8 Register Descriptions

See Chapter 1 for details on the abbreviations used inside the register descriptions. These peripheral registers can be operated in half-word (16-bit) or word (32-bit) format.

### 13.4.1 TIM1 and TIM8 Control Register 1 (TIMx\_CR1)

Offset address: 0x00

Reset value: 0x0000

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

Bit	notation	clarification
15:10	Reserved	Reserved, always reads 0.
9:8	CKD[1:0]	<b>CKD[1:0]:</b> Clock division factor (Clock division) These 2 bits define the ratio between the timer clock (CK_INT) frequency, the dead time and the division between the dead time generator and the sampling clock used by the digital filter (ETR, TIx). 00: tDTS=tCK_INT 01: tDTS=2xtCK_INT 10: tDTS=4xtCK_INT 11: Reserved, do not use this configuration
7	ARPE	<b>ARPE:</b> Auto-reload preload enable bit 0: TIMx_ARR register is not buffered; 1: The TIMx_ARR register is loaded into the buffer.
6:5	CMS[1:0]	<b>CMS[1:0]:</b> Center-aligned mode selection 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. The output compare interrupt flag bit for the channel configured as an output (CCxS=00 in the TIMx_CMRx register) is set only when the counter is counting down. 10: Central Alignment Mode 2. the counter alternately counts up and down. The output compare interrupt flag bit for the channel configured as an output (CCxS=00 in

		the TIMx_CMRx register) is set only when the counter counts up. 11: Central Alignment Mode 3. the counter alternately counts up and down. The output compare interrupt flag bit for the channel configured as an output (CCxS=00 in the TIMx_CMRx register) is set for both upward and downward counter counts. Note: With the counter on (CEN=1), switching from edge-aligned mode to center-aligned mode is not allowed.
4	DIR	<b>DIR:</b> Direction 0: Counter counts up; 1: The counter counts down. Note: This bit is read-only when the counter is configured for Central Alignment Mode or Encoder Mode.
3	OPM	<b>OPM:</b> 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	<b>URS:</b> Update request source The software selects the source of UEV events with this bit 0: If an update interrupt or DMA request is enabled, either of the following events generates an update interrupt or DMA request: -Counter overflow/underflow -Setting the UG bit -Updates generated from the mode controller 1: If the update interrupt or DMA request is enabled, the update interrupt or DMA request is generated only for counter overflow/underflow.
1	UDIS	<b>UDIS:</b> Update disable Software allows/disallows the generation of UEV events with this bit 0: UEV is allowed. the update (UEV) event is generated by any of the following events: -Counter overflow/underflow -Setting the UG bit -Updates generated from the mode controller Registers with caches are loaded with their preloaded values. (Translation: updating shadow registers) 1: Disable UEV. no update event is generated and the shadow registers (ARR, PSC, CCRx) maintain their values. If the UG bit is set or a hardware reset is issued from the mode controller, the counters and prescalers are reinitialized.
0	CEN	<b>CEN:</b> Counter enable 0: Disable the counter; 1: Enable the counter. Note: External Clock, Gated Mode and Encoder Mode will not work until the CEN bit is set in software. Trigger mode can automatically set the CEN bit in hardware.

### 13.4.2 TIM1 and TIM8 Control Register 2 (TIMx\_CR2)

Offset address: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]		CCDS	CCUS	Reserved	CCPC	
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

Bit	notation	clarification
15	Reserved	Reserved, always reads 0.
14	OIS4	OIS4: Output Idle State 4 (OC4 output). See OIS1 bit.
13	OIS3N	OIS3N: Output idle state 3 (OC3N output). See the OIS1N bit.
12	OIS3	OIS3: Output Idle State 3 (OC3 output). See OIS1 bit.
11	OIS2N	OIS2N: Output Idle State 2 (OC2N output). See the OIS1N bit.
10	OIS2	OIS2: Output Idle State 2 (OC2 output). See OIS1 bit.
9	OIS1N	OIS1N: Output Idle state1 (OC1N output) (Output Idle state1) 0: When MOE=0, OC1N=0 after deadband; 1: When MOE=0, OC1N=1 after deadband. Note: This bit cannot be modified after LOCK (TIMx_BKR register) level 1, 2, or 3 has been set.
8	OIS1	OIS1: Output Idle state1 (OC1 output) (Output Idle state1) 0: When MOE=0, OC1=0 after deadband if OC1N is realized; 1: When MOE=0, if OC1N is realized, OC1=1 after deadband. Note: This bit cannot be modified after LOCK (TIMx_BKR register) level 1, 2, or 3 has been set.
7	TI1S	TI1S: TI1 Selection 0: The TIMx_CH1 pin is connected to the TI1 input; 1: The TIMx_CH1, TIMx_CH2 and TIMx_CH3 pins are connected to the TI1 inputs after heterodyne.



6:4	MMS[2:0]	<b>MMS[2:0]: Master mode selection</b> These 3 bits are used to select the synchronization information (TRGO) to be 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 reset is generated by a trigger input (from a mode controller in reset mode), the signal on TRGO is delayed relative to the actual reset. 001: The enable-counter enable signal CNT_EN is used as a trigger output (TRGO). Sometimes it is necessary to start several 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 by a 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). 010: Update-Update events are selected as trigger inputs (TRGO). For example, a master timer clock can be used as a prescaler for a slave timer. 011: Compare Pulse - When a capture or a successful comparison occurs, the trigger output sends a positive pulse (TRGO) when the CC1IF flag is to be set (even if it is already high). 100: The compare-OC1REF signal is used as a trigger output (TRGO). 101: The compare-OC2REF signal is used as a trigger output (TRGO). 110: The compare-OC3REF signal is used as a trigger output (TRGO). 111: The compare-OC4REF signal is used as a trigger output (TRGO).
3	CCDS	<b>CCDS: Capture/compare DMA selection</b> 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	<b>CCUS: Capture/compare control update selection</b> 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	Reserved	Reserved, always reads 0.
0	CCPC	<b>CCPC: Capture/compare reloaded control bits</b> 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.

### 13.4.3 TIM1 and TIM8 Slave Mode Control Register (TIMx\_SMCR)

Offset address: 0x08

Reset value: 0x0000

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]		Reserv ed		SMS[2:0]	
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW		RW	RW	RW

Bit	notation	clarification
15	ETP	<b>ETP: External trigger polarity</b> This bit selects whether to use ETR or the inverse of ETR as the trigger operation 0: ETR is not inverted and is active high or on the rising edge; 1: ETR is inverted, active low or falling edge.
14	ECE	<b>ECE: External clock enable bit</b> This bit enables external clock mode 2 0: Disable external clock mode 2; 1: Enable external clock mode 2. the counter is driven by any valid edge on the ETRF signal. 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). 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'). Note 3: When External Clock Mode 1 and External Clock Mode 2 are both enabled, the external clock input is ETRF.
13:12	ETPS[1:0]	<b>ETPS[1:0]: External trigger prescaler</b> The frequency of the external trigger signal, ETRP, must be at most 1/4 of the TIMxCLK frequency. When a faster external clock is input, the frequency of ETRP can be reduced using prescaling. 00: Turns off the prescaler; 01: ETRP frequency divided by 2; 10: ETRP frequency divided by 4; 11: ETRP frequency divided by 8.
11:8	ETF[3:0]	<b>ETF[3:0]: External trigger filter</b> These bits define the frequency at which the ETRP signal is sampled and the

		<p>bandwidth at which the ETRP is digitally filtered. In effect, the digital filter is an event counter that records N events to produce a jump in the output.</p> <p>0000: no filter, sampled at fDTS  0001: Sampling frequency fSAMPLING=fCK_INT, N=2  0010: Sampling frequency fSAMPLING=fCK_INT, N=4  0011: Sampling frequency fSAMPLING=fCK_INT, N=8  0100: Sampling frequency fSAMPLING=fDTS/2, N=6  0101: Sampling frequency fSAMPLING=fDTS/2, N=8  0110: Sampling frequency fSAMPLING=fDTS/4, N=6  0111: Sampling frequency fSAMPLING=fDTS/4, N=8</p> <p>1000: Sampling frequency fSAMPLING=fDTS/8, N=6  1001: Sampling frequency fSAMPLING=fDTS/8, N=8  1010: Sampling frequency fSAMPLING=fDTS/16, N=5  1011: Sampling frequency fSAMPLING=fDTS/16, N=6  1100: Sampling frequency fSAMPLING=fDTS/16, N=8  1101: Sampling frequency fSAMPLING=fDTS/32, N=5  1110: Sampling frequency fSAMPLING=fDTS/32, N=6  1111: Sampling frequency fSAMPLING=fDTS/32, N=8</p>
7	MSM	<p><b>MSM:</b> Master/slave mode  0: No effect;  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[2:0]	<p><b>TS[2:0]:</b> Trigger selection  These 3-bit selections are used to synchronize the trigger input of the counter.</p> <p>000: Internal trigger 0 (ITR0)      100: Edge detector for TI1 (TI1F_ED)  001: Internal Trigger 1 (ITR1)      101: Filtered Timer Input 1 (TI1FP1)  010: Internal Trigger 2 (ITR2)      110: Filtered Timer Input 2 (TI2FP2)  011: Internal Trigger 3 (ITR3)      111: External Trigger Input (ETRF)</p> <p>For more details on ITRx, see Table72 .  Note: These bits can only be changed when they are not used (e.g. SMS=000) to avoid false edge detection when changed.</p>
3	Reserved	Reserved, always reads 0.
2:0	SMS[2:0]	<p><b>SMS[2:0]:</b> Slave mode selection  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 Input Control Register and Control Register)</p> <p>000: Off Slave Mode - If CEN=1, the prescaler is driven directly from 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 - Depending on the level of TI2FP2, the counter counts up/down on the edge of TI1FP1.  011: Encoder Mode 3 - Depending on the input level of another signal, the counter counts up/down on the edges of TI1FP1 and TI2FP2.  100: Reset Mode-The rising edge of the selected trigger input (TRGI) reinitializes the counter and generates a signal to update the register.  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 are controlled.  110: Trigger Mode - 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.  111: External Clock Mode 1-The rising edge of the selected trigger input (TRGI) drives the counter.</p> <p>Note: Do not use the 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 gating mode is to check the level of the trigger input.</p>

Table72 TIMx Internal Trigger Connections

From Timer	ITR0(TS=000)	ITR1 (TS=001)	ITR2 (TS=010)	ITR3 (TS=011)
TIM1	TIM5_TROG	TIM2_TROG	TIM3_TROG	TIM4_TROG
TIM8	TIM1_TROG	TIM2_TROG	TIM4_TROG	TIM5_TROG

## 13.4.4 TIM1 and TIM8 DMA/Interrupt Enable Registers (TIMx\_DIER)

Offset address: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserv ed	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	notation	clarification
15	Reserved	Reserved, always reads 0.
14	TDE	TDE: Trigger DMA request enable 0: Disable triggering of DMA request; 1: Allow triggering of DMA requests.
13	COMDE	COMDE: COM DMA request enable 0: DMA request from COM is disabled; 1: Allow COM's DMA request.
12	CC4DE	CC4DE: Capture/Compare 4 DMA requests enable 0: Capture/compare 4 DMA requests are disabled; 1: Allow capture/compare 4 DMA requests.
11	CC3DE	CC3DE: Capture/Compare 3 DMA request enable 0: Capture/compare 3 DMA requests are disabled; 1: Allow capture/comparison of DMA requests for 3.
10	CC2DE	CC2DE: Capture/Compare 2 DMA request enable 0: Capture/compare 2 DMA requests are disabled; 1: Allow capture/compare 2 DMA requests.
9	CC1DE	CC1DE: Capture/Compare 1 DMA request enable 0: Capture/compare 1 DMA requests are disabled; 1: Allow capture/compare 1 for DMA requests.
8	UDE	UDE: Update DMA request enable 0: DMA requests for updates are disabled; 1: Allow updated DMA requests.
7	BIE	BIE: Break interrupt enable 0: Disable brake interruption; 1: Allow the brake to be interrupted.
6	TIE	TIE: Trigger interrupt enable 0: Disable triggering of interrupts; 1: Enable trigger interrupt.
5	COMIE	COMIE: COM interrupt enable 0: Disable COM interrupt; 1: Allow COM interrupt.
4	CC4IE	CC4IE: Capture/Compare 4 interrupt enable 0: Capture/compare 4 interrupt disabled; 1: Allow capture/compare 4 interrupts.
3	CC3IE	CC3IE: Capture/Compare 3 interrupt enable 0: Capture/Compare 3 interrupt disabled; 1: Capture/compare 3 interrupts are allowed.
2	CC2IE	CC2IE: Capture/Compare 2 interrupt enable 0: Capture/Compare 2 interrupt disabled; 1: Allow capture/compare 2 interrupts.
1	CC1IE	CC1IE: Capture/Compare 1 interrupt enable 0: Capture/Compare 1 interrupt disabled; 1: Allow capture/compare 1 interrupt.
0	UIE	UIE: Update interrupt enable 0: Disable update interrupt; 1: Allow updates to be interrupted.

### 13.4.5 TIM1 and TIM8 Status Registers (TIMx\_SR)

Offset address: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	CC4OF	CC3OF	CC2OF	CC1OF	Reserved	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF		
	rc w0	rc w0	rc w0	rc w0		rc w0	rc w0	rc w0	rc w0	rc w0	rc w0	rc w0	rc w0	rc w0	rc w0

Bit	notation	clarification
15:13	Reserved	Reserved, always reads 0.
12	CC4OF	<b>CC4OF</b> : Capture/Compare 4 overcapture flag See CC1OF description.
11	CC3OF	<b>CC3OF</b> : Capture/Compare 3 overcapture flag See CC1OF description.
10	CC2OF	<b>CC2OF</b> : Capture/Compare 2 overcapture flag See CC1OF description.
9	CC1OF	<b>CC1OF</b> : Capture/Compare 1 overcapture flag This flag can be set to 1 by hardware only when the corresponding channel is configured for input capture. writing 0 clears this 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	Reserved	Reserved, always reads 0.
7	BIF	<b>BIF</b> : Break interrupt flag Once the brake input is valid, '1' to this bit by hardware. If the brake input is not valid, the bit can be cleared '0' by software. 0: No brake events are generated; 1: A valid level is detected on the brake input.
6	TIF	<b>TIF</b> : Trigger interrupt flag This position is '1' by hardware when a trigger event occurs (valid 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 software. 0: No trigger event is generated; 1: Trigger an interrupt to wait for a response.
5	COMIF	<b>COMIF</b> : COM interrupt flag This bit is set '1' by hardware as soon as a COM event is generated (when the capture/compare control bits: CCxE , CCxNE , OCxM have been updated). It is cleared '0' by software. 0: No COM events are generated; 1 : COM interrupt waiting for response.
4	CC4IF	<b>CC4IF</b> : Capture/Compare 4 interrupt flag Refer to the CC1IF description.
3	CC3IF	<b>CC3IF</b> : Capture/Compare 3 interrupt flag Refer to the CC1IF description.
2	CC2IF	<b>CC2IF</b> : Capture/Compare 2 interrupt flag Refer to the CC1IF description.
1	CC1IF	<b>CC1IF</b> : Capture/Compare 1 interrupt flag If channel CC1 is configured for output mode: This bit is set to 1 by hardware when 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 software. 0: No match occurred; 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 under a counter overflow in up or counting modes, or under a counter underflow condition in down counting mode If channel CC1 is configured for input mode: This bit is set '1' by hardware when a capture event occurs, it is cleared '0' by software or by reading TIMx_CCR1. 0: No input capture is generated; 1: The counter value has been captured (copied) to TIMx_CCR1 (an edge of the same polarity as selected is detected on IC1).
0	UIF	<b>UIF</b> : Update interrupt flag This bit is set '1' by hardware when an update event is generated. It is cleared '0' by software. 0: No update event is generated; 1: Update interrupt wait response. This bit is set '1' by hardware when the register is updated: - If UDIS=0 in the TIMx_CR1 register, when the repeat counter value overflows or underflows (repeat counter=0 generates an event). - If URS=0 and UDIS=0 in the TIMx_CR1 register, an update is generated when UG=1 in the TIMx_EGR register is set, and when the counter CNT is re-initialized by software. - If URS=0 , UDIS=0 of TIMx_CR1 register, when counter CNT is reinitialized by the trigger event. (Refer to 12.4.3: TIM1 and TIM8 Slave Mode Control Register (TIMx_SMCR)).

### 13.4.6 TIM1 and TIM8 Event Generation Registers (TIMx\_EGR)

Offset address: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
								W	W	W	W	W	W	W	W

Bit	notation	clarification
15:8	Reserved	Reserved, always reads 0.
7	BG	<b>BG</b> : Break generation This bit is set '1' by software to generate a brake event and is automatically cleared '0' by hardware. 0: No action; 1: Generate a brake event. At this time, MOE=0, BIF=1, if the corresponding interrupt and DMA are turned on, the corresponding interrupt and DMA will be generated.
6	TG	<b>TG</b> : Trigger generation This bit is set '1' by software to generate a trigger event that is automatically cleared '0' by hardware. 0: No action; 1 : TIF=1 in the TIMx_SR register generates the corresponding interrupt and DMA if they are turned on.
5	COMG	<b>COMG</b> : Capture/Compare control update generation This bit is set '1' by software and cleared '0' automatically by hardware. 0: No action; 1: When CCPC = 1, the CCxE, CCxNE, and OCxM bits are allowed to be updated. Note: This bit is only valid for channels that have complementary outputs.
4	CC4G	<b>CC4G</b> : Generate Capture/Compare 4 generation Refer to the CC1G description.
3	CC3G	<b>CC3G</b> : Generate Capture/Compare 3 generation Refer to the CC1G description.
2	CC2G	<b>CC2G</b> : Generate Capture/Compare 2 generation Refer to the CC1G description.
1	CC1G	<b>CC1G</b> : Capture/Compare 1 generation This bit is set '1' by software to generate a capture/compare event and is automatically cleared '0' by hardware. 0: No action; 1: Generate a capture/compare event on channel CC1: If channel CC1 is configured as an output: Set CC1IF=1 to generate the corresponding interrupt and DMA if they are turned on. If channel CC1 is configured as an input: The current counter value is captured to the TIMx_CCR1 register; set CC1IF=1 to generate the corresponding interrupts and DMAs if they are turned on. set CC1OF=1 if CC1IF is already 1.
0	UG	<b>UG</b> : Update generation This bit is set '1' by software and cleared '0' automatically by hardware. 0: No action; 1: Reinitialize the counter and generate an update event. Note that the prescaler counter is also cleared '0' (but the prescaler count remains unchanged). The counter is cleared '0' if in centrosymmetric mode or if DIR=0 (counting up); if DIR=1 (counting down) the counter takes the value of TIMx_ARR.

### 13.4.7 TIM1 and TIM8 Capture/Compare Mode Register 1 ( TIMx\_CCMR1 )

Offset address: 0x18

Reset value: 0x0000

The channel can be used as input (capture mode) or output (compare mode), and the direction of the channel is defined by the corresponding CCxS bit. The other bits of this register function differently in input and output modes; OCxx describes the function of the channel in output mode and ICxx describes the function of the channel in input mode. Therefore, it is important to note that the same bit functions differently in output and input modes.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2:0]		OC2PE	OC2FE	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	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

## Output comparison mode:

Bit	notation	clarification
15	OC2CE	OC2CE: Output Compare 2 clear enable
14:12	OC2M[2:0]	OC2M[2:0]: Output Compare 2 mode
11	OC2PE	OC2PE: Output Compare 2 preload enable
10	OC2FE	OC2FE: Output Compare 2 fast enable
9:8	CC2S[1:0]	<p><b>CC2S[1:0]:</b> Capture/Compare 2 selection. (Capture/Compare 2 selection) This bit defines the direction of the channel (input/output), and the selection of the input pin:</p> <p>00 : 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 : CC2 channel is configured as an input and IC2 is mapped on TI1;</p> <p>11 : The CC2 channel is configured as an input with IC2 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: CC2S is writable only when the channel is closed (CC2E=0 in the TIMx_CCER register).</p>
7	OC1CE	<p><b>OC1CE:</b> Output Compare 1 clear '0' enable</p> <p>0 : OC1REF is not affected by the ETRF input;</p> <p>1 : Clear OC1REF=0 once the ETRF input is detected high.</p>
6:4	OC1M[2:0]	<p><b>OC1M[2:0]:</b> Output Compare 1 mode</p> <p>These 3 bits define the action of the output reference signal OC1REF, which determines the value of OC1, OC1N. OC1REF is active high, and the effective level of OC1, OC1N depends on the CC1P, CC1NP bits.</p> <p>000: Freeze. Comparison between output comparison register TIMx_CCR1 and counter TIMx_CNT does not work for OC1REF;</p> <p>001: Set channel 1 to active level when matched. Force OC1REF high when counter TIMx_CNT is the same value as capture/compare register 1 (TIMx_CCR1).</p> <p>010: Set channel 1 to invalid level when matched. Force OC1REF low when counter TIMx_CNT is the same value as capture/compare register 1 (TIMx_CCR1).</p> <p>011: Flip. Flips the level of OC1REF when TIMx_CCR1 = TIMx_CNT.</p> <p>100: Forces an invalid level. Forces OC1REF to low. 101: Forced to valid level. Forces OC1REF to be high.</p> <p>110: PWM Mode 1 - In up count, channel 1 is valid once TIMx_CNT &lt; TIMx_CCR1, otherwise it is invalid; in down count, channel 1 is invalid once TIMx_CNT &gt; TIMx_CCR1 (OC1REF=0), it is valid ( OC1REF=1).</p> <p>111: PWM Mode 2 - In up count, channel 1 is invalid once TIMx_CNT &lt; TIMx_CCR1, otherwise it is valid; in down count, channel 1 is valid once TIMx_CNT &gt; 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 for 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 Output Compare Mode.</p>
3	OC1PE	<p><b>OC1PE :</b> Output Compare 1 preload enable</p> <p>0: Disable the preload function of TIMx_CCR1 register, it can be written to TIMx_CCR1 register at any time, and the newly written value takes effect immediately.</p> <p>1: Enable the preload function of TIMx_CCR1 register, read/write operation is only operated on the preloaded register, and 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 for output).</p> <p>Note 2: Only in single pulse mode (OPM=1 in TIMx_CR1 register), PWM mode can be used without checking the preload register, otherwise its action is uncertain.</p>
2	OC1FE	<p><b>OC1FE :</b> Output Compare 1 fast enable</p> <p>This bit is used to speed up the response of the CC outputs 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 a edge.</p> <p>1: The active edge input to the flip-flop acts as if a comparison match has occurred. Therefore, OC is set to the comparison level independent of the comparison result. The delay between the active edge of the sampling flip-flop and the CC1 output is reduced to 3 clock cycles.</p> <p>OCFE only works when the channel is configured for PWM1 or PWM2 mode.</p>
1:0	CC1S[1:0]	<p><b>CC1S[1:0]:</b> Capture/Compare 1 selection. (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 input and IC1 is mapped on TI1; 10 : CC1 channel is configured as input and IC1 is mapped on TI2;</p> <p>11 : The CC1 channel is configured as an input and IC1 is mapped on TRC. This mode operates only when the internal trigger input is selected (by the TS bit of the</p>



		TIMx_SMCR register). Note: CC1S is writable only when the channel is closed (CC1E=0 in the TIMx_CCER register).
--	--	--

#### Input Capture Mode:

Bit	notation	instructions
15:12	IC2F[3:0]	IC2F[3:0]: Input capture 2 filter
11:10	IC2PSC [1:0]	IC2PSC[1:0]: Input capture 2 prescaler
9:8	CC2S[1:0]	<b>CC2S[1:0]: Capture/Compare 2 selection</b> These 2 bits define the direction of the channel (input/output), and the selection of the input pin: 00 : CC2 channel is configured as an output; 01 : CC2 channel is configured as an input and IC2 is mapped on TI2 10 : CC2 channel is configured as an input and IC2 is mapped on TI1; 11 : The CC2 channel is configured as an input with IC2 mapped on TRC. This mode operates only when the internal trigger input is selected (by the TS bit of the TIMx_SMCR register). Note: CC2S is writable only when the channel is closed (CC2E=0 in the TIMx_CCER register).
7:4	IC1F[3:0]	<b>IC1F[3:0]: Input capture 1 filter</b> These bits define the sampling frequency of the TI1 input and the length of the digital filter. The digital filter consists of an event counter that registers N events before generating a jump in the output: 0000: no filter, sampled at fDTS 0001: Sampling frequency fSAMPLING=fDTS/8, N=6 0010: Sampling frequency fSAMPLING=fDTS/8, N=8 0011: Sampling frequency fSAMPLING=fDTS/16, N=5 0100: Sampling frequency fSAMPLING=fDTS/16, N=6 0101: Sampling frequency fSAMPLING=fDTS/2, N=6 0110: Sampling frequency fSAMPLING=fDTS/2, N=8 0111: Sampling frequency fSAMPLING=fDTS/4, N=6 1000: Sampling frequency fSAMPLING=fDTS/8, N=6 1001: Sampling frequency fSAMPLING=fDTS/8, N=8 1010: Sampling frequency fSAMPLING=fDTS/16, N=5 1011: Sampling frequency fSAMPLING=fDTS/16, N=6 1100: Sampling frequency fSAMPLING=fDTS/16, N=8 1101: Sampling frequency fSAMPLING=fDTS/32, N=5 1110: Sampling frequency fSAMPLING=fDTS/32, N=6 1111: Sampling frequency fSAMPLING=fDTS/32, N=8
3:2	IC1PSC [1:0]	<b>IC1PSC[1:0]: Input capture 1 prescaler</b> These 2 bits define the prescaler coefficient for the CC1 input (IC1). Once CC1E=0 (in the TIMx_CCER register), the prescaler is reset. 00: No prescaler, each edge detected on the capture input port 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]	<b>CC1S[1:0] : Capture/Compare 1 Selection</b> 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 : The CC1 channel is configured as an input and IC1 is mapped on TRC. This mode operates only when the internal trigger input is 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).

### 13.4.8 TIM1 and TIM8 Capture/Compare Mode Register 2 (TIMx\_CCMR2)

Offset address: 0x1C

Reset value: 0x0000

Refer to the description of the CCMR1 register above

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4CE	OC4M [2:0]			OC4PE	OC4FE	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	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

#### Output comparison mode:

Bit	notation	clarification
15	OC4CE	OC4CE: Output compare 4 clear enable
14:12	OC4M [2:0]	OC4M[2:0]: Output compare 4 mode
11	OC4PE	OC4PE: Output compare 4 preload enable
10	OC4FE	OC4FE: Output compare 4 fast enable
9:8	CC4S[1:0]	CC4S[1:0]: Capture/Compare 4 selection These 2 bits define the direction of the channel (input/output), and the selection of the input pin: 00 : CC4 channel is configured as an output; 01 : CC4 channel is configured as input and IC4 is mapped on TI4; 10 : CC4 channel is configured as input and IC4 is mapped on TI3; 11 : The CC4 channel is configured as an input with IC4 mapped on TRC. This mode operates only when the internal trigger input is selected (by the TS bit of the TIMx_SMCR register). Note: CC4S is writable only when the channel is closed (CC4E=0 in the TIMx_CCER register).
7	OC3CE	OC3CE: Output compare 3 clear enable
6:4	OC3M [2:0]	OC3M[2:0]: Output compare 3 mode
3	OC3PE	OC3PE: Output compare 3 preload enable
2	OC3FE	OC3FE: Output compare 3 fast enable
1:0	CC3S[1:0]	CC3S[1:0]: Capture/Compare 3 selection These 2 bits define the direction of the channel (input/output), and the selection of the input pin: 00 : CC3 channel is configured as an output; 01 : CC3 channel is configured as an input and IC3 is mapped on TI3 10 : CC3 channel is configured as an input and IC3 is mapped on TI4; 11 : The CC3 channel is configured as an input with IC3 mapped on TRC. This mode operates only when the internal trigger input is selected (by the TS bit of the TIMx_SMCR register). Note: CC3S is writable only when the channel is closed (CC3E=0 in the TIMx_CCER register).

#### Input Capture Mode:

Bit	notation	clarification
15:12	IC4F[3:0]	IC4F[3:0]: Input capture 4 filter
11:10	IC4PSC [1:0]	IC4PSC[1:0]: Input capture 4 prescaler
9:8	CC4S[1:0]	CC4S[1:0]: Capture/Compare 4 selection These 2 bits define the direction of the channel (input/output), and the selection of the input pin: 00 : CC4 channel is configured as an output; 01 : CC4 channel is configured as an input and IC4 is mapped on TI4 10 : CC4 channel is configured as an input and IC4 is mapped on TI3; 11 : The CC4 channel is configured as an input with IC4 mapped on TRC. This mode operates only when the internal trigger input is selected (by the TS bit of the TIMx_SMCR register). Note: CC4S is writable only when the channel is closed (CC4E=0 in the TIMx_CCER register).
7:4	IC3F[3:0]	IC3F[3:0]: Input capture 3 filter
3:2	IC3PSC [1:0]	IC3PSC[1:0]: input capture 3 prescaler
1:0	CC3S[1:0]	CC3S[1:0]: Capture/compare 3 selection These 2 bits define the direction of the channel (input/output), and the selection of the input pin:



		00 : CC3 channel is configured as an output; 01 : CC3 channel is configured as an input and IC3 is mapped on TI3 10 : CC3 channel is configured as an input and IC3 is mapped on TI4; 11 : The CC3 channel is configured as an input with IC3 mapped on TRC. This mode operates only when the internal trigger input is selected (by the TS bit of the TIMx_SMCR register). Note: CC3S is writable only when the channel is closed (CC3E=0 in the TIMx_CCER register).
--	--	--

### 13.4.9 TIM1 and TIM8 Capture/Compare Enable Registers (TIMx\_CCER)

Offset address: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4NP	Reserv ed	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	rW

Bit	notation	clarification
15	CC4NP	<b>CC4NP</b> : Input/capture 4 output polarity, refer to the description of CC1NP.
14	Reserved	Reserved, always reads 0.
13	CC4P	<b>CC4P</b> : Input/Compare 4 output polarity Refer to the description of CC1P.
12	CC4E	<b>CC4E</b> : Capture/Compare 4 output enable Refer to the description of CC1E.
11	CC3NP	<b>CC3NP</b> : Input/Capture 3 complementary output polarity Refer to the description of CC1NP.
10	CC3NE	<b>CC3NE</b> : Capture/Compare 3 complementary output enable Refer to the description of CC1NE.
9	CC3P	<b>CC3P</b> : Input/Compare 3 output polarity Refer to the description of CC1P.
8	CC3E	<b>CC3E</b> : Capture/Compare 3 output enable Refer to the description of CC1E.
7	CC2NP	<b>CC2NP</b> : Input/Capture 2 complementary output polarity Refer to the description of CC1NP.
6	CC2NE	<b>CC2NE</b> : Capture/Compare 2 complementary output enable Refer to the description of CC1NE.
5	CC2P	<b>CC2P</b> : Input/Compare 2 output polarity Refer to the description of CC1P.
4	CC2E	<b>CC2E</b> : Capture/Compare 2 output enable Refer to the description of CC1E.
3	CC1NP	<b>CC1NP</b> : Input/Capture 1 complementary output polarity 0 : OC1N active high; 1 : OC1N active low. Note: Once the LOCK level (LOCK bit in the TIMx_BDTR register) is set to 3 or 2 and CC1S=00 (channel configured for output) this bit cannot be modified.
2	CC1NE	<b>CC1NE</b> : Input/Capture 1 complementary output enable 0: Off - OC1N disables the output, so the level of OC1N is dependent on the values of the MOE, OSS1, OSSR, OIS1, OIS1N and CC1E bits. 1 : On - The OC1N signal is output to the corresponding output pin, and its output level is dependent on the values of the MOE, OSS1, OSSR, OIS1, OIS1N, and CC1E bits.
1	CC1P	<b>CC1P</b> : Input/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: This bit selects whether IC1 or the inverted signal of IC1 is used as the trigger or capture signal. 0: not inverted: capture occurs on the rising edge of IC1; when used as an external trigger, IC1 is not inverted. 1: Inverted: Capture occurs on the falling edge of IC1; when used as an external trigger, IC1 is inverted. Note: Once the LOCK level (LOCK bit in the TIMx_BDTR register) is set to 3 or 2, this bit cannot be modified.
0	CC1E	<b>CC1E</b> : Capture/Compare 1 output enable The CC1 channel is configured as an output: 0 : Off - OC1 disables output, so the output level of OC1 is dependent on the values of the MOE, OSS1, OSSR, OIS1, OIS1N and CC1NE bits. 1 : ON - The OC1 signal is output to the corresponding output pin, and its output level depends on the values of the MOE, OSS1, OSSR, OIS1, OIS1N and CC1NE bits. The CC1 channel is configured as an input: This bit determines whether the counter value can be captured into the TIMx_CCR1 register. 0: Capture disabled;

0: Capture enable.

Table73 Control bits for complementary output channels OCx and OCxN with brake function

control bits					Output state <sup>(1)</sup>	
MOE position	OSSI bit	OSSR bit	CCxE bit	CCxNE bit	OCx Output Status	OCxN Output Status
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= OCxREF xor CCxP, OCx_EN=1	Output disabled (disconnected from timer) OCxN=0, OCxN_EN=0
		0	1	1	OCxREF + Polarity + Deadband, OCx_EN=1	OCxREF Inverted + 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= OCxREF xor CCxNP, OCxN_EN=1
		1	1	0	OCxREF + polarity. OCx= OCxREF xor CCxP, OCx_EN=1	Off state (output enabled and disabled) OCxN=CCxNP, OCxN_EN=1
		1	1	1	OCxREF + Polarity + Deadband, OCx_EN=1	OCxREF Inverted + Polarity + Deadband, OCxN_EN=1
0	0	X	0	0	Output inhibit (disconnected from timer) Asynchronously: OCx=CCxP, OCx_EN=0, OCxN=CCxNP, OCxN_EN=0; If the clock exists: OCx=OISx and OCxN=OISxN after a dead time, assume that OISx and OISxN do not both correspond to the valid levels of OCx and OCxN.	
	0		0	1		
	0		1	0		
	0		1	1		
	1		0	0	Off state (output enabled and invalid level) Asynchronously: OCx=CCxP, OCx_EN=1, OCxN=CCxNP, OCxN_EN=1; If the clock exists: OCx=OISx and OCxN=OISxN after a dead time, assume that OISx and OISxN do not both correspond to the valid levels of OCx and OCxN.	
	1		0	1		
	1		1	0		
	1		1	1		

(1).If neither of the 2 outputs of a channel is used (CCxE = CCxNE = 0), then OISx, OISxN, CCxP and CCxNP must all be cleared to zero.

Notes: The status of the external I/O pins whose pins are connected to the complementary OCx and OCxN channels depends on the OCx and OCxN channel status and the GPIO and AFIO registers.

### 13.4.10 TIM1 and TIM8 Counters (TIMx\_CNT)

Offset address: 0x24

Reset value: 0x0000

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

### 13.4.11 TIM1 and TIM8 Prescalers (TIMx\_PSC)

Offset address: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC [15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit	notation	clarification													
15:0	PSC [15:0]	PSC[15:0]: Prescaler value The counter clock frequency (CK_CNT) is equal to fCK_PSC/( PSC[15:0]+1). The PSC contains the value loaded into the current prescaler register each time an update													

event is generated; an update event consists of the counter being cleared '0' by the UG bit of TIM\_EGR or by a slave controller operating in reset mode.

### 13.4.12 TIM1 and TIM8 Auto-Reload Registers (TIMx\_ARR)

Offset Address: 0x2C

Reset value: 0xFFFF

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	notation	clarification													
15:0	ARR[15:0]	<b>ARR[15:0]:</b> Prescaler value The ARR contains the values that will be loaded into the actual Auto-Reload Register. Refer to the 13.3.1 section for details: updates and actions related to the ARR. The counter does not work when the value of Auto-Reload is empty.													

### 13.4.13 TIM1 and TIM8 Repeat Counter Registers (TIMx\_RCR)

Offset address: 0x30

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								REP[7:0]							
								rW	rW	rW	rW	rW	rW	rW	rW
Bit	notation	clarification													
15:8	Reserved	Reserved, always reads 0.													
7:0	REP[7:0]	<b>REP[7:0]:</b> Repetition counter value With preload enabled, these bits allow the user to set the update rate of the comparison registers (i.e., periodic transfers from the preload register to the current register); if generating update interrupts is permitted, this also affects the rate at which update interrupts are generated. 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 will only take effect when the next cycle update event occurs. This means that in PWM mode, (REP+1) corresponds to: - The number of PWM cycles in edge-aligned mode; - Number of PWM half-cycles in centrosymmetric mode;													

### 13.4.14 TIM1 and TIM8 Capture/Compare Register 1 (TIMx\_CCR1)

Offset address: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW
Bit	notation	clarification													
15:0	CCR1[15:0]	<b>CCR1[15:0]:</b> Capture/Compare Channel 1 value 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 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/Compare1 register only when an update event occurs. The current capture/compare register participates in the comparison with counter TIMx_CNT and generates an output signal on port OC1. If the CC1 channel is configured as an input: CCR1 contains the counter value transmitted by the last input capture 1 event (IC1).													

### 13.4.15 TIM1 and TIM8 Capture/Compare Register 2 (TIMx\_CCR2)

Offset address: 0x38

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit	notation	clarification													
15:0	CCR2[15:0]	<b>CCR2[15:0]: Capture/Compare 2 value</b> If the CC2 channel is configured as an output: CCR2 contains the value loaded into the current Capture/Compare 2 register (preloaded value). If the preload feature is not selected in the TIMx_CMR2 register (OC2PE bit), the written value is immediately transferred to the current register. Otherwise, this preloaded value is transferred to the current Capture/Compare2 register only when an update event occurs. The current capture/compare register participates in the comparison with counter TIMx_CNT and generates an output signal on port OC2. If the CC2 channel is configured as an input: CCR2 contains the counter value transmitted by the last input capture 2 event (IC2).													

### 13.4.16 TIM1 and TIM8 Capture/Compare Register 3 (TIMx\_CCR3)

Offset address: 0x3C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro
Bit	notation	instructions													
15:0	CCR3[15:0]	<b>CCR3[15:0]: Capture/Compare 3 value</b> If the CC3 channel is configured as an output: CCR3 contains the value loaded into the current Capture/Compare 3 register (preloaded value). If the preload feature is not selected in the TIMx_CMR3 register (OC3PE bit), the written value is immediately transferred to the current register. Otherwise, this preloaded value is transferred to the current capture/compare3 register only when an update event occurs. The current capture/comparison register participates in the comparison with counter TIMx_CNT and generates an output signal on port OC3. If the CC3 channel is configured as an input: CCR3 contains the counter value transmitted by the last input capture 3 event (IC3).													

### 13.4.17 TIM1 and TIM8 Capture/Compare Registers (TIMx\_CCR4)

Offset address: 0x40

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit	notation	clarification													
15:0	CCR4[15:0]	<b>CCR4[15:0]: Capture/Compare Channel 4 value</b> If the CC4 channel is configured as an output: CCR4 contains the value loaded into the current Capture/Compare 4 register (preloaded value). If the preload feature is not selected in the TIMx_CMR4 register (OC4PE bit), the written value is immediately transferred to the current register. Otherwise this preloaded value is transferred to the current capture/compare4 register only when an update event occurs. The current capture/comparison register participates in the comparison with counter TIMx_CNT and generates an output signal on port OC4. If the CC4 channel is configured as an input: CCR4 contains the counter value transmitted by the last input capture 4 event (IC4).													

## 13.4.18 TIM1 and TIM8 Brake and Deadband Registers (TIMx\_BDTR)

Offset address: 0x44

Reset value: 0x0000

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

*Comment: Depending on the lock setting, the AOE, BKP, BKE, OSSI, OSSR, and DTG[7:0] bits can be write-protected, and it is necessary to configure them the first time the TIMx\_BDTR register is written.*

Bit	notation	clarification
15	MOE	<b>MOE:</b> Main output enable This bit is cleared '0' asynchronously by hardware once the brake input is active. Depending on the value set for the AOE bit, this bit can be cleared '0' by software or automatically set to 1. It is only valid for channels configured as outputs. 0: Disable OC and OCN output or force to idle state; 1: If the corresponding enable bits (CCxE, CCxNE bits of TIMx_CCER register) are set, OC and OCN outputs are enabled. See section 13.4.9, TIM1 and TIM8 Capture/Compare Enable Registers (TIMx_CCER) for details on OC/OCN enable.
14	AOE	<b>AOE:</b> Automatic output enable 0 : MOE can only be set to '1' by the software; 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 invalid). Note: Once the LOCK level (LOCK bit in the TIMx_BDTR register) is set to '1', this bit cannot be modified.
13	BKP	<b>BKP:</b> Brake input polarity (Break polarity) 0: Brake input active low; 1: Brake input active high. Note: Once the LOCK level (LOCK bit in the TIMx_BDTR register) is set to '1', this bit cannot be modified. Note: Any write operation to this bit requires a delay of one APB clock before it works.
12	BKE	<b>BKE:</b> Break enable 0: Disable brake input (BRK and CCS clock failure event); 1: Turn on the brake input (BRK and CCS clock failure event). Note: When LOCK level 1 is set (LOCK bit in the TIMx_BDTR register), this bit cannot be modified. Note: Any write operation to this bit requires a delay of one APB clock before it works.
11	OSSR	<b>OSSR:</b> Off-stateselection for Run mode 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. Refer to OC/OCN Enable for a detailed description (13.4.9 section, TIM1 and TIM8 Capture/Compare Enable Registers (TIMx_CCER)). 0: Disable OC/OCN output when the timer is not operating (OC/OCN enable output signal = 0); 1: When the timer is not working, once CCxE=1 or CCxNE=1, first turn on OC/OCN and output the invalid level, and then set OC/OCN enable output signal=1. Note: Once the LOCK level (LOCK bit in the TIMx_BDTR register) is set to 2, this bit cannot be modified.
10	OSSI	<b>OSSI:</b> Off-stateselection for Idle mode This bit is used when MOE=0 and the channel is set to output. Refer to OC/OCN Enable for a detailed description (13.4.9 section, TIM1 and TIM8 Capture/Compare Enable Registers (TIMx_CCER)). 0: Disable OC/OCN output when the timer is not operating (OC/OCN enable output signal = 0); 1 : When the timer is not working, once CCxE=1 or CCxNE=1, OC/OCN first outputs its idle level, and then OC/OCN enables the output signal=1. Note: Once the LOCK level (LOCK bit in the TIMx_BDTR register) is set to 2, this bit cannot be modified.
9:8	LOOK[1:0]	<b>LOOK[1:0]:</b> Lock configuration This bit provides write protection against software errors. 00: Lock off, registers are not write-protected; 01: Lock level 1, cannot write to the DTG, BKE, BKP, AOE bits of the TIMx_BDTR register and the OISx/OISxN bits of the TIMx_CR2 register; 10: Latch level 2, cannot write to each bit in Latch level 1, nor can it write to the CC polarity bit (once the channel in question is set to output via the CCxS bit, the CC polarity bit is the CCxP/CCNxP bit of the TIMx_CCER register) and the OSSR/OSSI bits; 11: Latch level 3, cannot write to each bit in Latch level 2, and cannot write to the CC control bit (once the channel in question is set to output via the CCxS bit, the CC control bit is the OCxM/OCxPE bit of the TIMx_CMRx register);

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

Bit	notation	clarification
15:13	Reserved	Reserved, always reads 0.
12:8	DBL [4:0]	<b>DBL[4:0]:</b> DMA burst length These bits define the length of the DMA transfer in continuous mode (when a read or write is made to the TIMx_DMAR register, the timer makes one continuous transfer), i.e.: defines the number of transfers, which can be half-words (double bytes) or bytes: 00000: 1

[illegible]

Bit	notation	clarification
31:0	DMAB [31:0]	<b>DMAB[31:0]:</b> DMA register for burst accesses A read or write to the TIMx_DMAR register results in an access operation to the register where the following address is located TIMx_CR1 address + DBA + DMA index, where: "TIMx_CR1 Address" is the address where control register 1 (TIMx_CR1) is located "DBA" is the base address defined in the TIMx_DCR register; The "DMA Index" is an offset that is automatically controlled by the DMA and depends on the DBL defined in the TIMx_DCR register.

#### Example of how to use the DMA burst feature

In this example, the timer DMA burst feature is used to update the contents of the CCRx registers (x = 2, 3, 4), and the DMA sends half a word of the Transferred to the CCRx register.

This is accomplished through the following steps.

1. Configure the corresponding DMA channels as follows.
  - The DMA channel peripheral address is the DMAR register address
  - The DMA channel memory address is the address of the buffer in RAM that contains the data to be transferred by DMA to the CCRx register.
  - Number of data to be transferred = 3 (see note below).
  - Turns off circular mode.
1. Configure the DCR registers by configuring the DBA and DBL bit fields as follows:DBL=3 transmissions, DBA=0xE
2. 3. enable TIMx update DMA request (set UDE bit in DIER register).
3. 4. Enable TIMx
4. Enabling DMA channels

**WARNING:** *This example applies if each CCRx register needs to be updated once. If each CCRx register needs to be updated twice, then the number of data to be transferred should be 6. Take the example of a buffer in RAM containing data 1, data 2, data 3, data 4, data 5 and data 6. Transfer data to the CCRx registers as follows:On the first update DMA request, data 1 is transferred to CCR2, data 2 is transferred to CCR3, and data 3 is transferred to CCR4; on the second update DMA request, data 4 is transferred to CCR2, data 5 is transferred to CCR3, and data 6 is transferred to CCR4.*

## 13.4.21 TIM1 and TIM8 Registers

The following table maps all registers of TIM1 and TIM8 into a 16-bit addressable (addressed) space.

Table74 TIM1 and TIM8 Register and Reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																		
000h	TIMx_CR1	Reserved																						CKD [1:0]		APRE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN																		
	Reset value																							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
004h	TIMx_CR2	Reserved																		OIS4		OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS [2:0]		CCDS	CCUS	Reserved	CCPC																	
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						
008h	TIMx_SMCR	Reserved																		ETP	ECE	ETPS [1:0]		EFT [3:0]			MSM		TS [2:0]		Reserved	SMS [2:0]																			
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
00Ch	TIMx_DIER	Reserved																		TDE	COMDE		CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE																
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
010h	TIMx_SR	Reserved																CC4OF		CC3OF	CC2OF	CC1OF	Reserved	BIF	TIF	COM1F	CC41F	CC31F	CC21F	CC11F	UTF																				
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									
014h	TIMx_EGR	Reserved																						BG	TG	COM	CC4G	CC3G	CC2G	CC1G	UG																				
	Reset value																							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
018h	TIMx_CCMR1 Output Compare Mode	Reserved																		OC2CE	OC2M [2:0]		OC2PE	OC2FE	CC2S [1:0]		OC1CE	OC1M [2:0]		OC1PE	OC1FE	CC1S [1:0]																			
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							
	TIMx_CCMR1 Input Capture Mode	Reserved																		IC2F [3:0]		IC2 PSC [1:0]		CC2S [1:0]		IC1F [3:0]		IC1 PSC [1:0]		CC1S [1:0]																					
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
01Ch	TIMx_CCMR2 Output Compare Mode	Reserved																		OC4CE	OC4M [2:0]		OC4PE	OC4FE	CC4S [1:0]		OC3CE	OC3M [2:0]		OC3PE	OC3FE	CC3S [1:0]																			
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
	TIMx_CCMR2 Input Capture Mode	Reserved																		IC4F [3:0]		IC4 PSC [1:0]		CC4S [1:0]		IC3F [3:0]		IC3 PSC [1:0]		CC3S [1:0]																					
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
020h	TIMx_CCER	Reserved																		CC4NP	Reserved	CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E																
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
024h	TIMx_CNT	Reserved																		CNT [15:0]																															
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				



Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
028h	TIMx_PSC	Reserved																PSC [15:0]																			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
02Ch	TIMx_ARR	Reserved																ARR[15:0]																			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
030h	TIMx_RCR	Reserved																								REP[7:0]											
	Reset value																									0	0	0	0	0	0	0	0	0	0	0	0
034h	TIMx_CCR1	Reserved																CCR1[15:0]																			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
038h	TIMx_CCR2	Reserved																CCR2[15:0]																			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
03Ch	TIMx_CCR3	Reserved																CCR3[15:0]																			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
040h	TIMx_CCR4	Reserved																CCR4[15:0]																			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
044h	TIMx_BDTR	Reserved																MOE	AOE	BKP	BKE	OSSI	OSSI	LOCK [1:0]	DT[7:0]												
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
048h	TIMx_DCR	Reserved																				DBL [4:0]				Reserved				DBA [4:0]							
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
04Ch	TIMx_DMAR	DMAB [31:0]																																			
	Reset value																																				

SeeTable1 for register start addresses.

## 14 General-Purpose Timer (TIM2 to TIM5)

### 14.1 Introduction to TIM2 to TIM5

The general-purpose timer is a 16-bit auto-load counter constructed by driving it through 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.

Each timer is completely independent and does not share any resources with each other. They can operate synchronously together, see section 14.3.15.

### 14.2 TIM2 to TIM5 Main Functions

General purpose TIMx (TIM2, TIM3, TIM4, and TIM5) timer functions include:

- 16-bit up, down, up/down auto-load counter
- 16-bit programmable (can be modified in real time) prescaler, counter clock frequency division factor is any value between 1 ~ 65536
- 4 independent channels:
  - Input Capture
  - Output Comparison
  - PWM generation (edge or center alignment mode)
  - Single pulse mode output
- Synchronization circuits using external signals to control timers and timer interconnections
- An interrupt/DMA is generated when the following events occur:
  - Updates: Counter overflow up/down, counter initialization (via software or internal/external trigger)
  - Trigger event (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

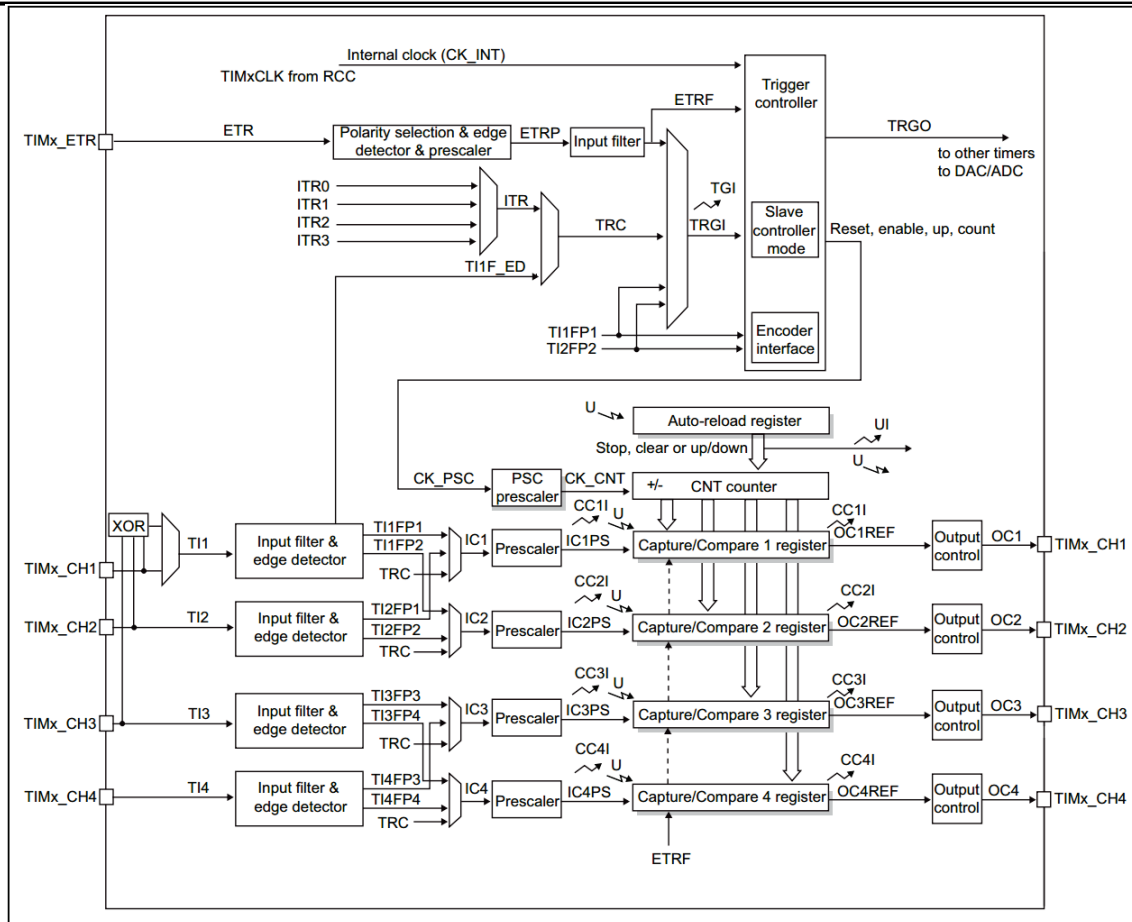




Figure 96 General Purpose Timer Block Diagram

Notes:  The contents of the preload register are transmitted to the working register at the U event according to the setting of the control bit

 event

 Interrupt and DMA output

## 14.3 TIM2 to TIM5 Functional Description

### 14.3.1 Time Base Unit (In Computing)

The main part of the programmable general purpose timer is a 16-bit counter and its associated auto-load register. The counter count up, count down, or count up and down in both directions. The counter clock is divided by a prescaler.

The counter, auto-load registers, and prescaler registers can be read and written by software and remain read and written while the counter is running. The time base unit contains:

- Counter Register (TIMx\_CNT)
- Prescaler register (TIMx\_PSC)
- Autoload Register (TIMx\_ARR)

The autoloader registers are preloaded, and writing or reading the auto-reload registers will access the preload registers. Depending on the setting of the Auto-Load Pre-Load Enable bit (ARPE) in the TIMx\_CR1 register, the contents of the Pre-Load Register are transferred to the Shadow Register either immediately or at each update event UEV. An update event is generated when the counter reaches an overflow condition (underflow condition when counting down) and when the UDIS bit in the TIMx\_CR1 register is equal to '0'. The update event can also be generated by software. The generation of update events for each configuration is described in detail later.

The counter is driven by the prescaler's clock output, CK\_CNT, which is valid only when the counter bit (CEN) in the counter's TIMx\_CR1 register is set. (See the Slave Mode description of the controller for details on counter enable).

Notes: The true counter enable signal, CNT\_EN, is set after one clock cycle of CEN.

## Prescaler Description

The prescaler divides the counter clock frequency by any value between 1 and 65536. It is based on a 16-bit counter controlled by a 16-bit register (in the TIMx\_PSC register). This control register is buffered and can be during operation. The new prescaler parameter is used when the next update event arrives.

Figure97 and Figure98 give examples of changing counter parameters while the prescaler is running.

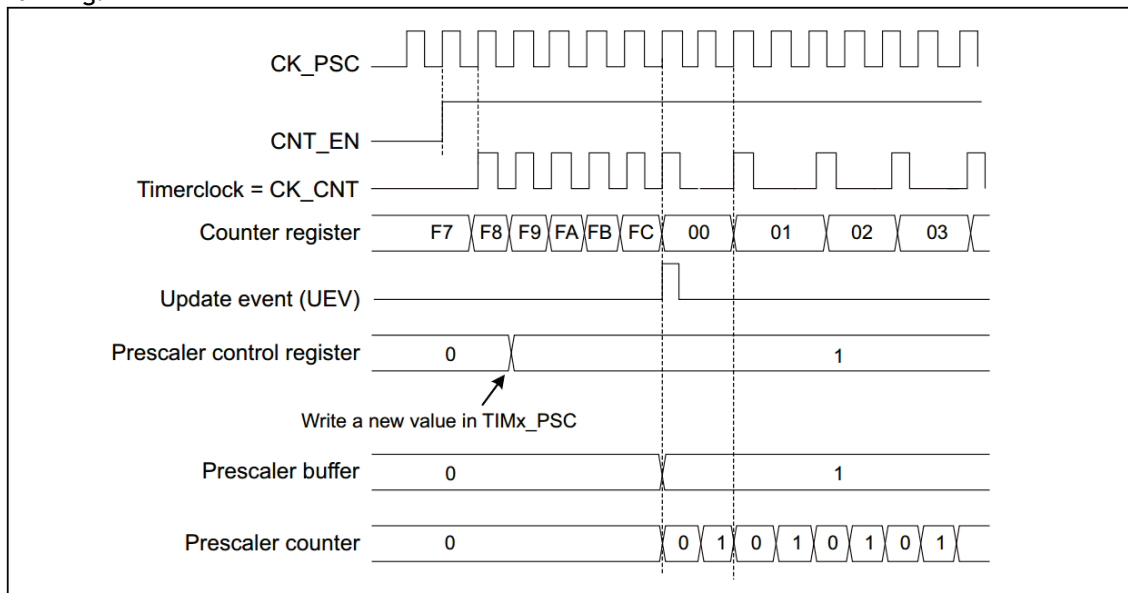


Figure97 Timing diagram of the counter when the parameter of the prescaler is changed from 1 to 2

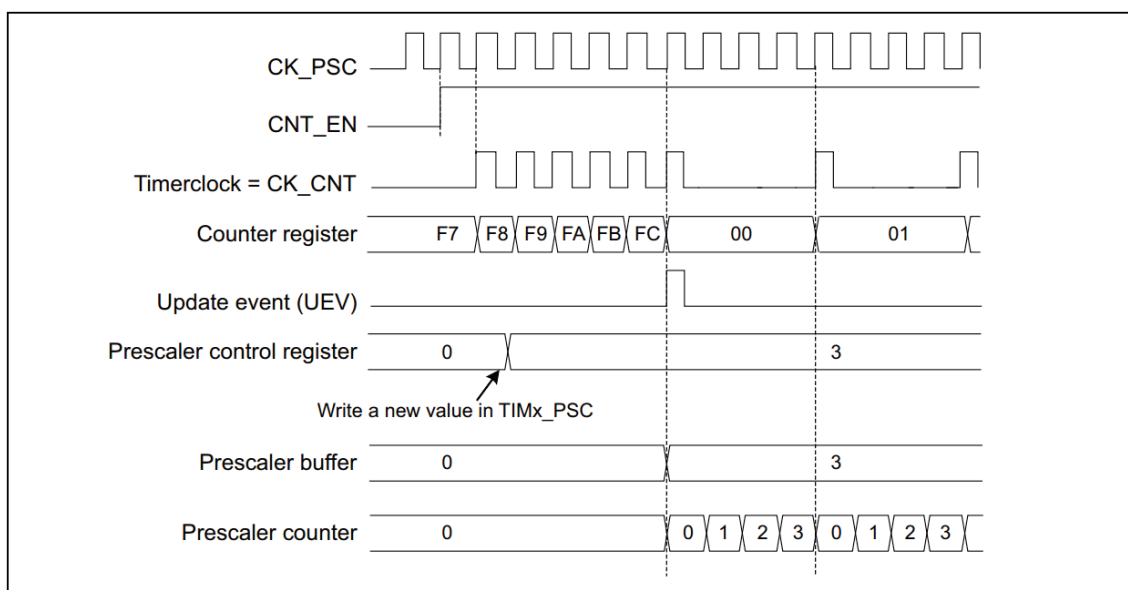


Figure98 Timing diagram of the counter when the parameter of the prescaler is changed from 1 to 4

## 14.3.2 Counter Mode

### Up Count Mode

In count-up mode, the counter counts from 0 to the auto-load value (the contents of the TIMx\_ARR counter), then starts counting from 0 again and generates a counter overflow event. An update event can be generated each time the counter overflows, and an update event can also be generated by setting the UG bit in the TIMx\_EGR register (either in software or using the slave mode controller).

Setting the UDIS bit in the TIMx\_CR1 register disables the update event; this prevents the shadow register from being updated when a new value is written to the preload register. No update event is generated until the UDIS bit is cleared '0'. However, when an update should be generated, the counter will still be cleared to '0' and the prescaler count will be reset to 0 (but the prescaler factor will remain unchanged). In addition, if the URS bit in the TIMx\_CR1 register is set (select update request), setting the UG bit will generate an update event UEV, but the UIF flag is not set by the hardware (i.e., no interrupt or DMA request will be); this is to avoid simultaneous update and capture interrupts when the counter is cleared 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 prescaler buffer is set to the value of the preload register (the contents of the TIMx\_PSC register).
  - The autoloader shadow register is reset to the value of the preload register (TIMx\_ARR).
- The following figure gives some examples of how the counter acts at different clock frequencies when TIMx\_ARR=0x36.

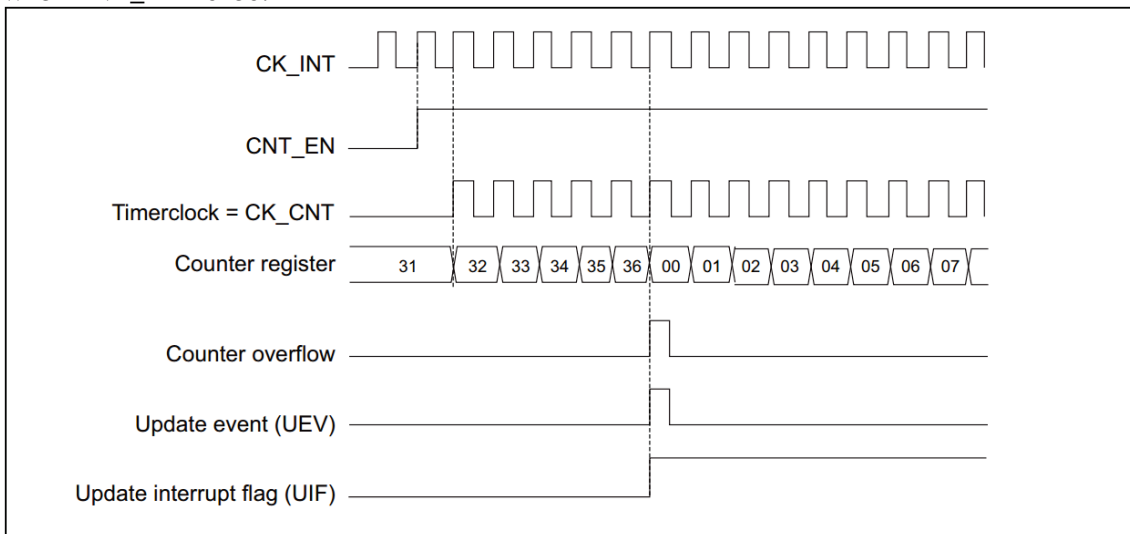


Figure99 Timing diagram of the counter with internal clock division factor of 1

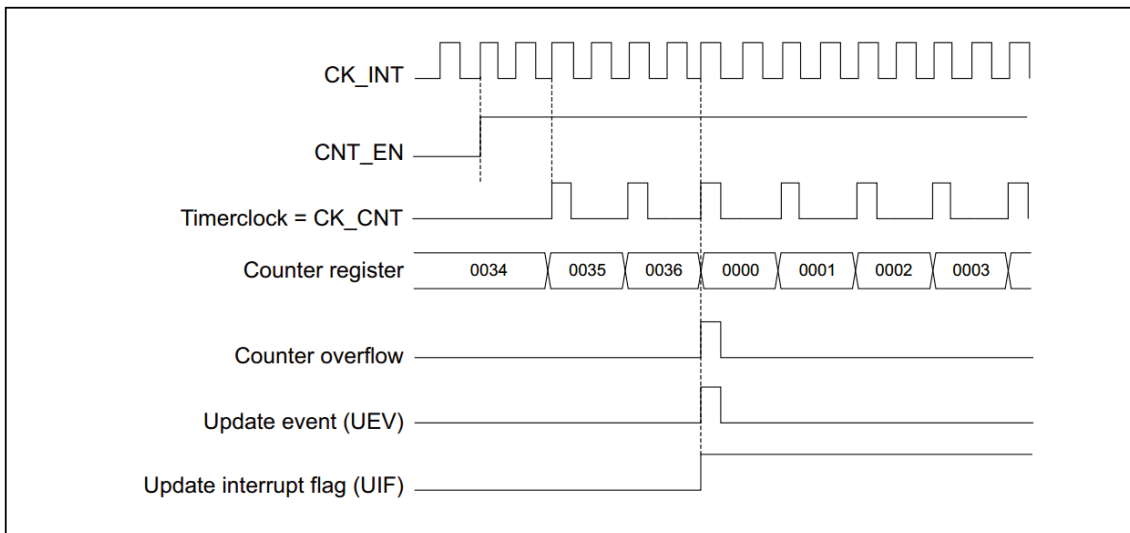


Figure100 Timing diagram of the counter with internal clock division factor of 2

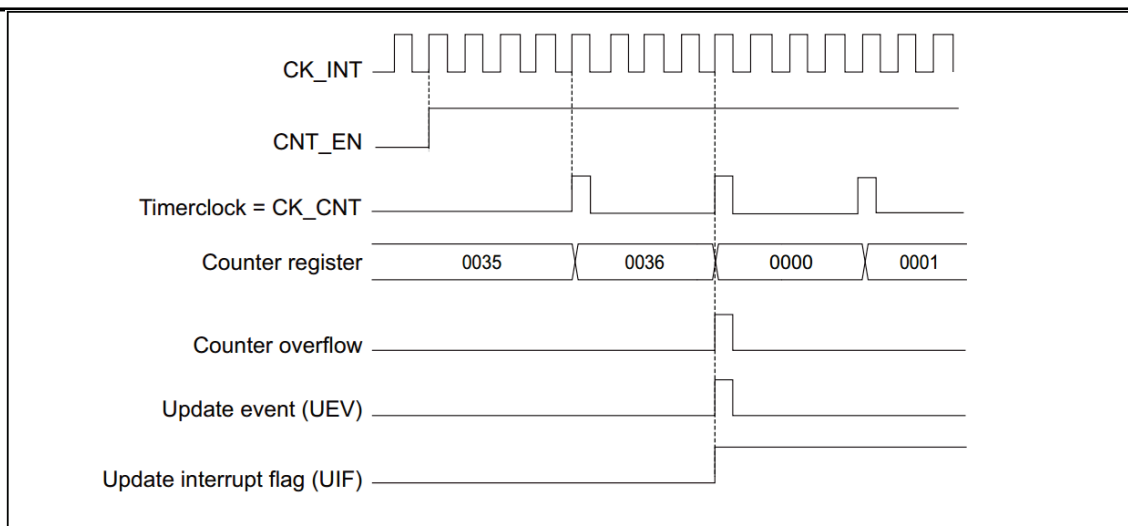


Figure101 Timing diagram of the counter with internal clock division factor of 4

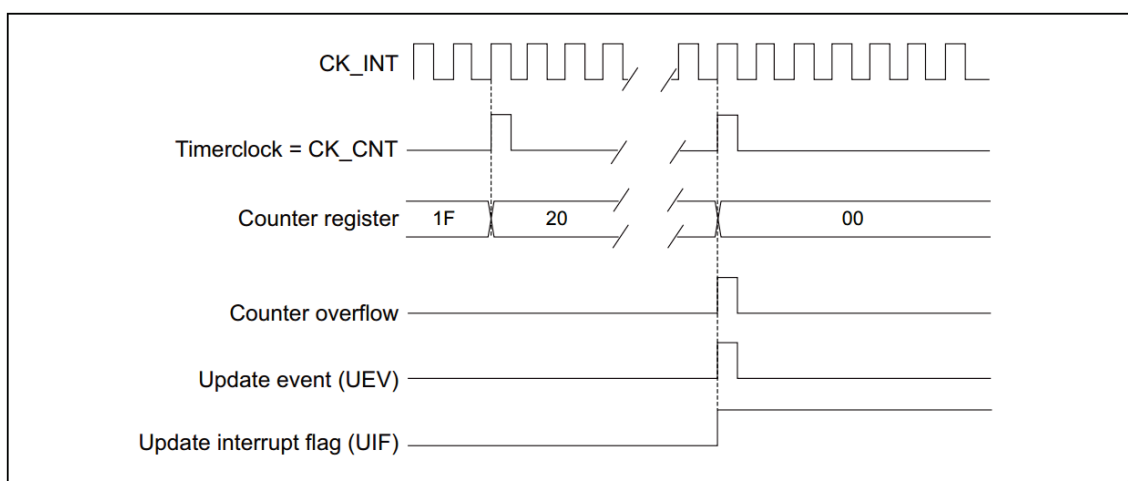


Figure102 Timing diagram of the counter with internal clock division factor N

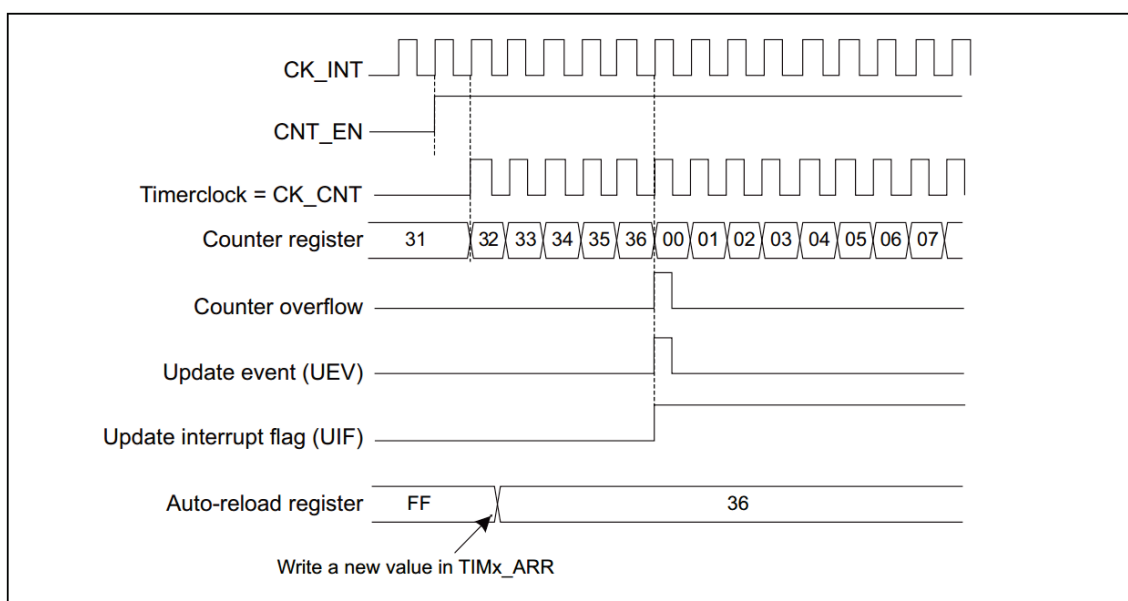


Figure103 Counter timing diagram, update event when ARPE=0 (TIMx\_ARR is not preloaded)

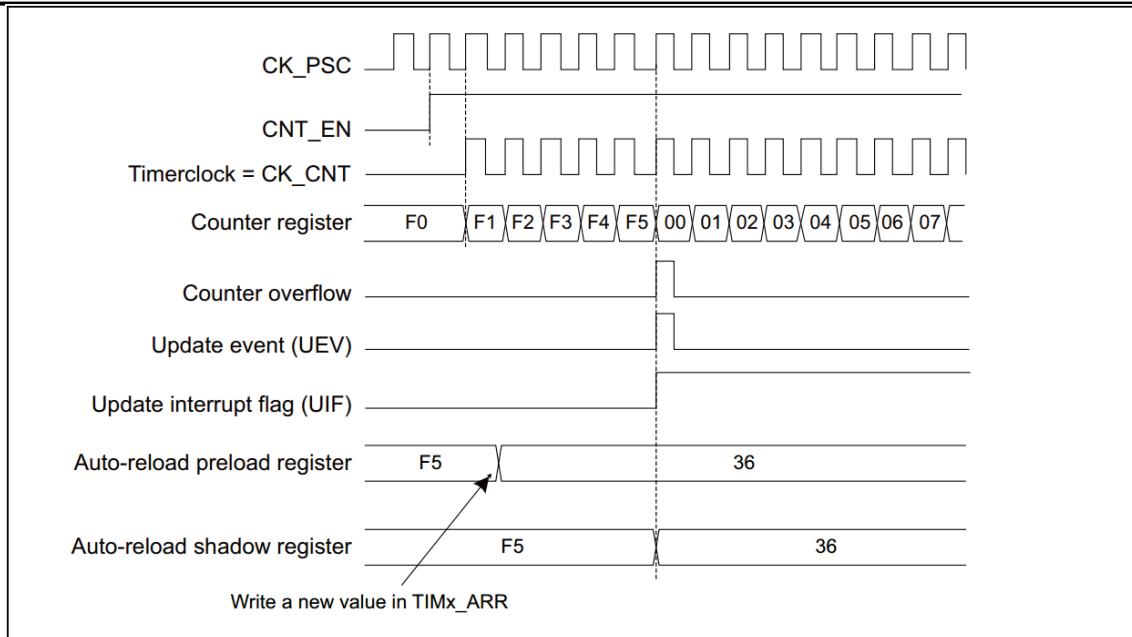


Figure104 Counter timing diagram, update event when ARPE=1 (preloaded with TIMx\_ARR)

### Down Count Mode

In down mode, the counter counts down from the auto-loaded value (the value of the TIMx\_ARR counter) to 0, then restarts from the value and generates a counter down overflow event.

An update event can be generated each time the counter overflows, and an update event can also be generated by setting the UG bit in the TIMx\_EGR register (either in software or using a slave mode controller).

Setting the UDIS bit of the TIMx\_CR1 register disables UEV events. This prevents the shadow register from being updated when a new value is written to the preload register. Therefore no update event is generated until the UDIS bit is cleared to '0'. However, the counter will still restart counting from the current value and the prescaler counter will restart from 0 (but the prescaler factor remains unchanged).

In addition, if the URS bit (select update request) in the TIMx\_CR1 register is set, 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 both an update and a capture interrupt when a capture event occurs and clears the .

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 ) is set.

- The prescaler buffer is set to the value of the preload register (the value of the TIMx\_PSC register).

- The current autoloader register is updated to the preloaded value (the contents of the TIMx\_ARR register). Note: The autoloader is updated before the counter is reloaded, so the next cycle will be the expected value.

Here are some examples of counter operation at different clock frequencies when TIMx\_ARR=0x36.

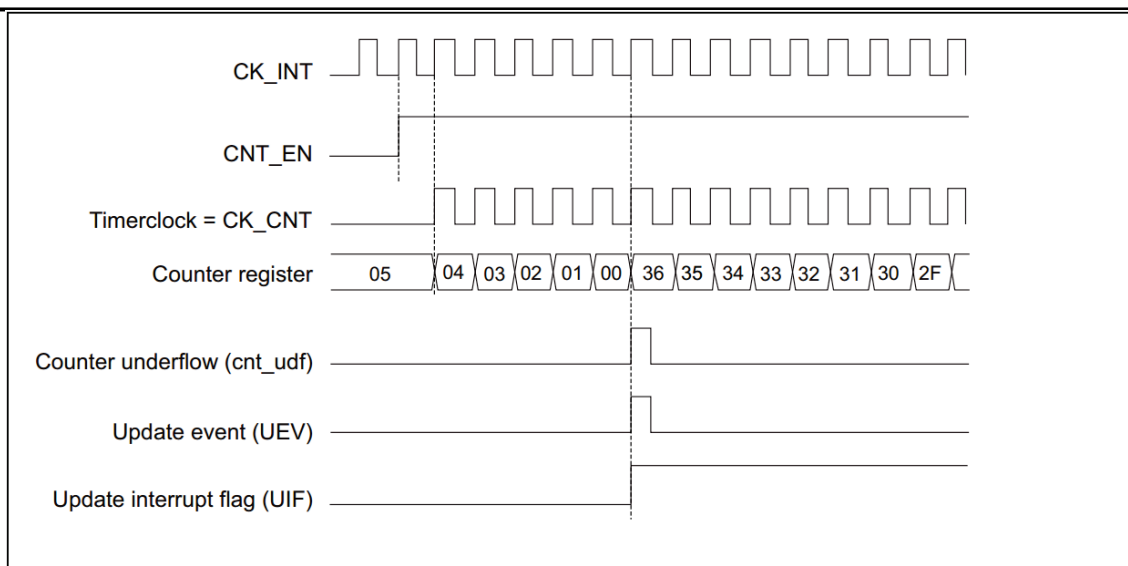


Figure105 Timing diagram of the counter with internal clock division factor of 1

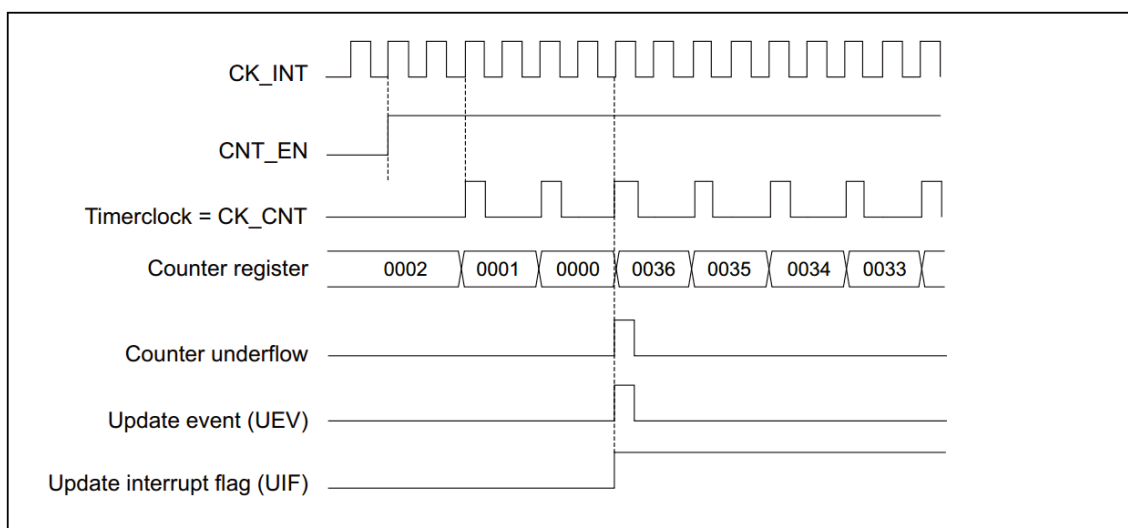


Figure106 Timing diagram of the counter with internal clock division factor of 2

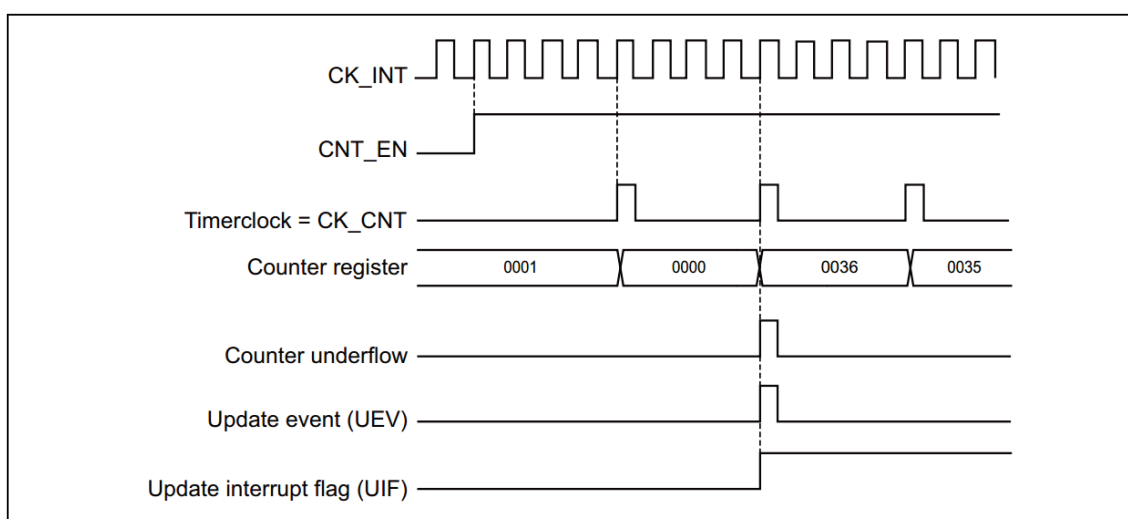


Figure107 Timing diagram of the counter with internal clock division factor of 4



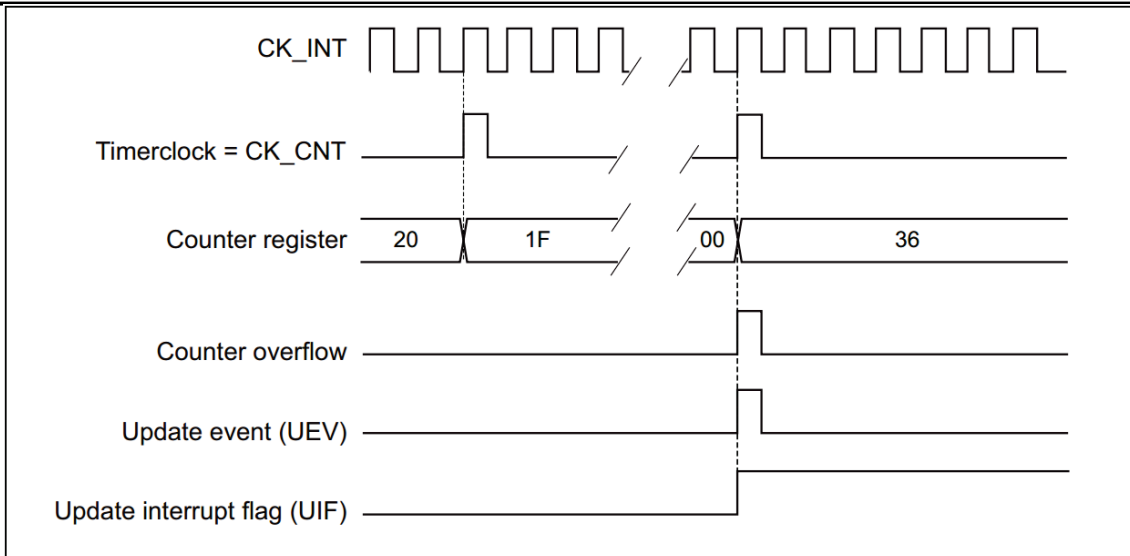


Figure108 Timing diagram of the counter with internal clock division factor N

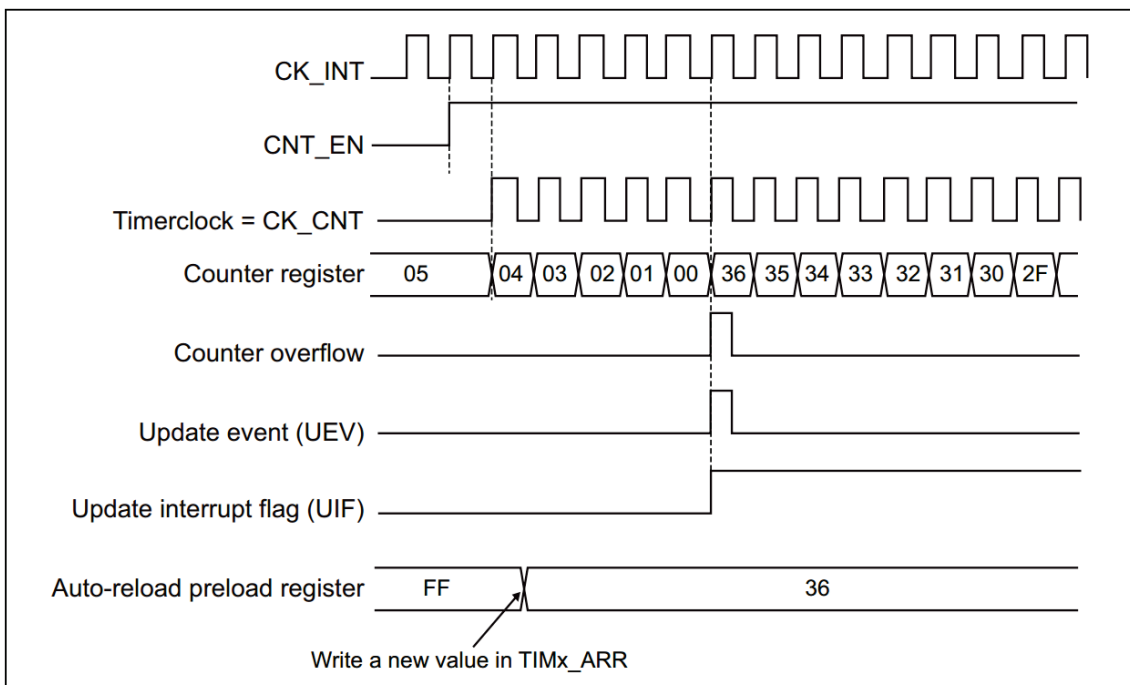


Figure109 Counter Timing Diagram, Update Events When Repeat Counter is Not Used

### 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; and then counts from 0 again.

In this mode, the DIR direction bit in TIMx\_CR1 cannot be written. It is updated by hardware and indicates the current count direction.

The update event can be generated on every count overflow and every count underflow; it can also be generated by setting the UG bit in the TIMx\_EGR register (either in software or using a slave mode controller). The counter then resumes counting from 0, and the prescaler resumes counting from 0.

Setting the UDIS bit in the TIMx\_CR1 register disables UEV events. This prevents the shadow register from being updated when a new value is written to the preload register. Therefore no update event is generated until the UDIS bit is cleared to '0'. However, the counter will continue to count up or down depending on the current auto-reload value.

In addition, if the URS bit (select update request) in the TIMx\_CR1 register is set, 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 both an update and a capture interrupt when a capture event occurs and clears the .

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 ) is set.

- The prescaler buffer is loaded with the value of the preload (TIMx\_PSC register).
- The current autoloader register is updated to the preloaded value (the contents of the TIMx\_ARR register).

**Notes:** *If an update occurs due to a counter overflow, Auto Reload will be updated before the counter is reloaded, so the next cycle will be expected value (the counter is loaded with the new value).*

The following are some examples of counter operation at different clock frequencies:

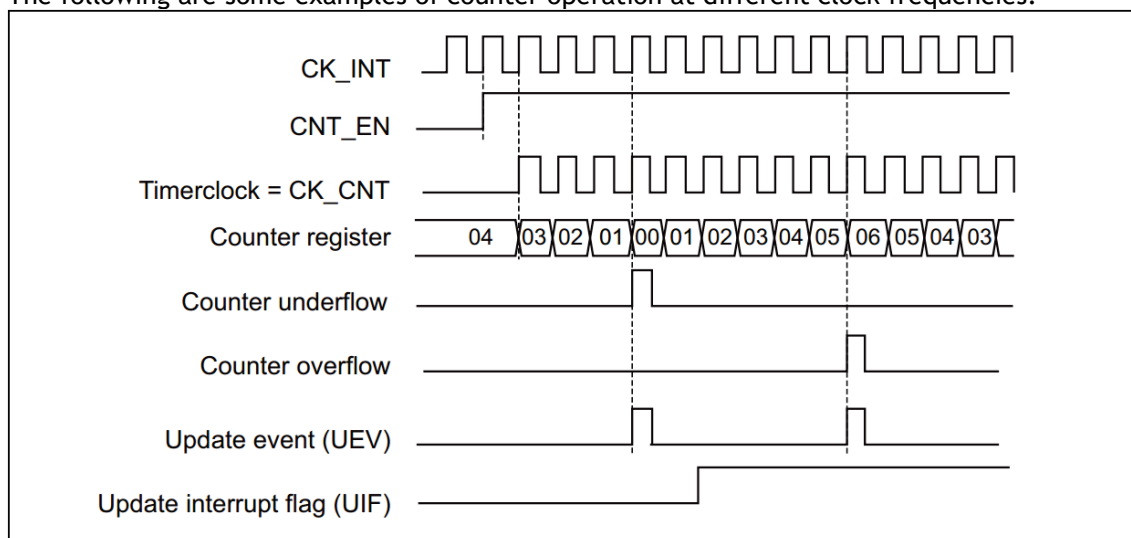


Figure110 Counter Timing Diagram with Internal Clock Division Factor of 1 and TIMx\_ARR=0x6

1. Center alignment mode 1 is used here (see section14.4.1 for details).

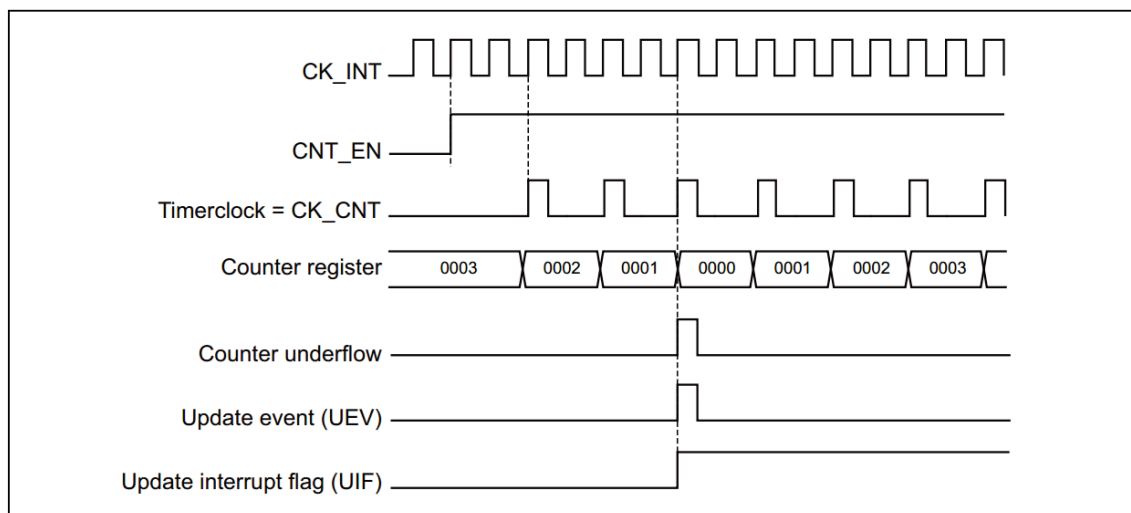


Figure111 Timing diagram of the counter with internal clock division factor of 2

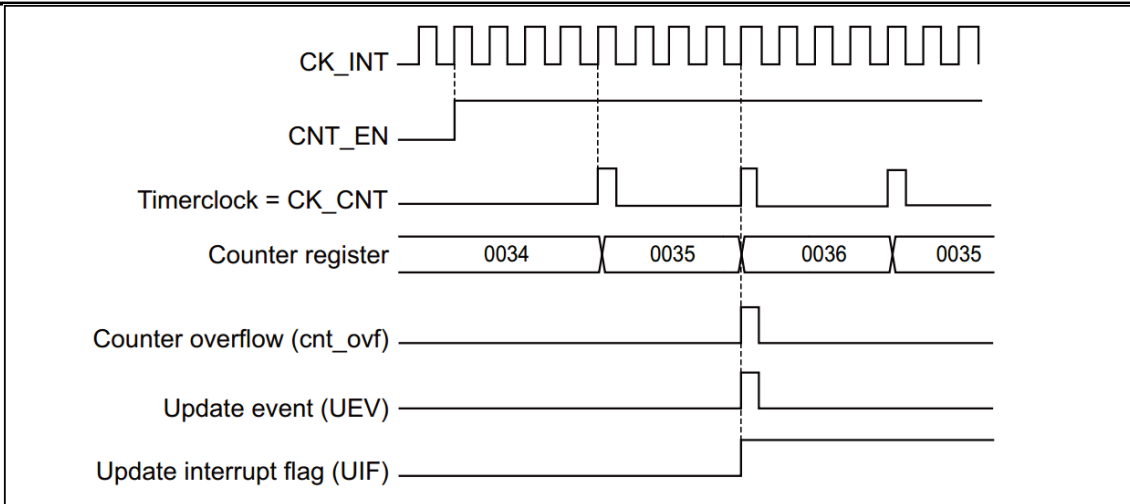


Figure112 Counter Timing Diagram with Internal Clock Division Factor of 4, TIMx\_ARR=0x36

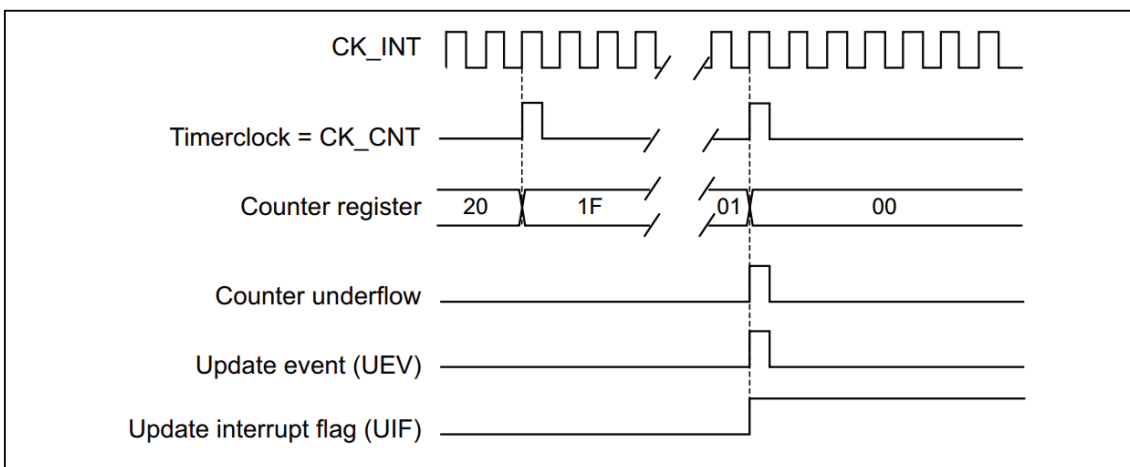


Figure113 Timing diagram of the counter with internal clock division factor N

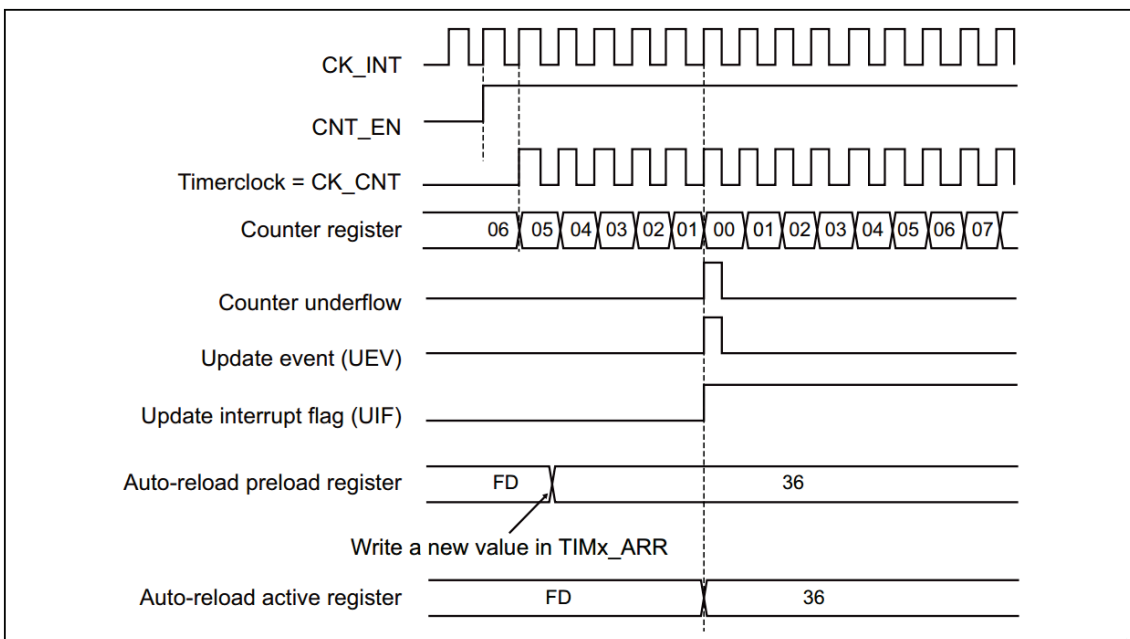


Figure114 Counter Timing Diagram, Update Event at ARPE=1 (Counter Underflow)

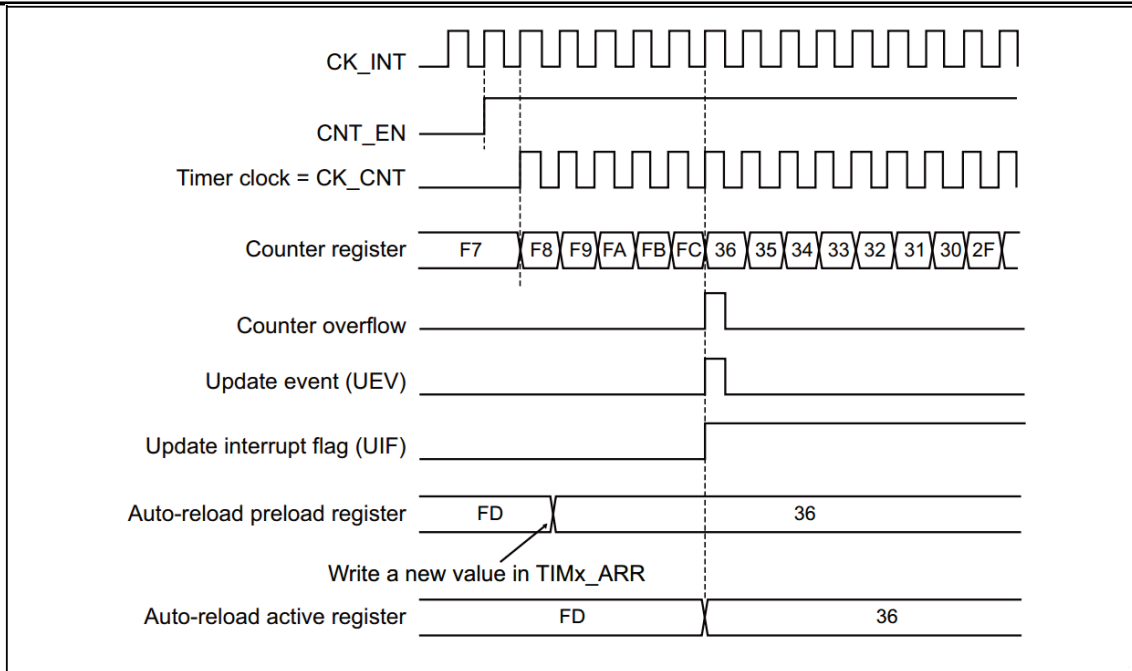


Figure115 Counter Timing Diagram, Update Event at ARPE=1 (Counter Overflow)

### 14.3.3 Clock Selection

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

- Internal clock (CK\_INT)
- External Clock Mode 1: External Input Pin (TIx)
- External Clock Mode 2: External Trigger Input (ETR)
- Internal Trigger Input (ITRx): Uses one timer as a prescaler for another timer, e.g. you can configure one timer Timer1 as a prescaler for another timer Timer2. See 13.3.15.

#### Internal Clock Source (CK\_INT)

If the slave mode controller is disabled (SMS=000 in the TIMx\_SMCR register), the CEN, DIR (TIMx\_CR1 register) and UG bits (TIMx\_EGR register) are de facto control bits and can only be modified by software (the UG bit is still cleared automatically). As long as the CEN bit is written to '1', the prescaler clock is provided by the internal clock CK\_INT.

The following figure shows the operation of the control circuit and up counter in the general mode without prescaler.

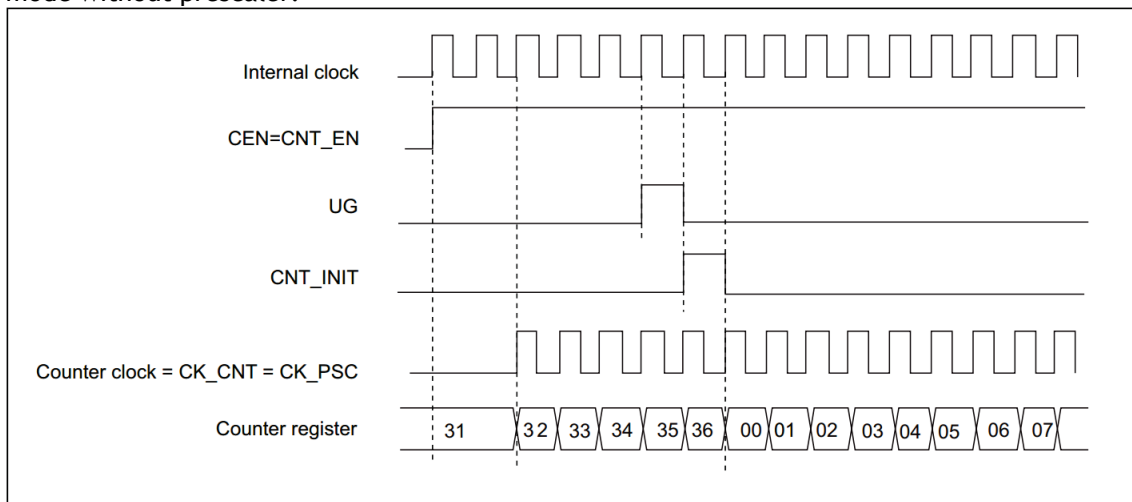


Figure116 Control circuit in general mode with internal clock division factor of 1

#### 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.

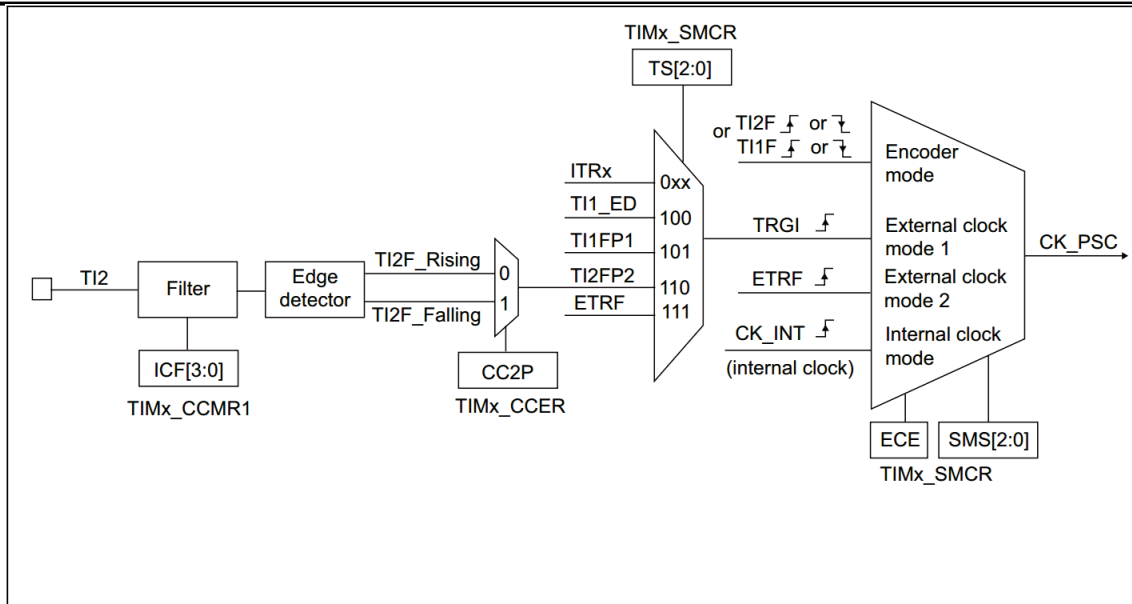


Figure117 TI2 External Clock Connection Example

For example, to configure the up counter to count on the rising edge of the TI2 input, use the following procedure:

1. Configure TIMx\_CMR1 register CC2S='01' to configure channel 2 to detect the rising edge of the TI2 input
2. Configure IC2F[3:0] of the TIMx\_CMR1 register to select the input filter bandwidth (if no filter is required, keep IC2F=0000)

- Notes: The capture prescaler is not used as a trigger, so there is no need to configure it
3. Configure CC2P='0' in the TIMx\_CCER register to select the rising edge polarity
  4. Configure the TIMx\_SMCR register with SMS='111' to select timer 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

When the rising edge occurs at TI2, the counter counts 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.

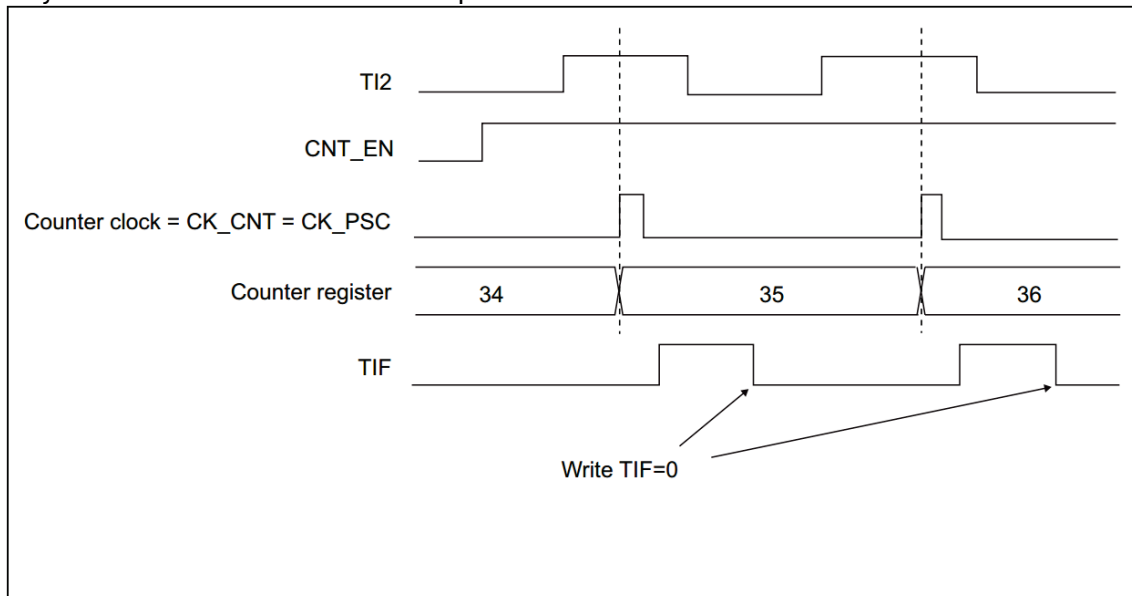


Figure118 Control circuit in external clock mode 1

### External Clock Source Mode 2

This mode is selected by making ECE=1 in the TIMx\_SMCR register. The counter is able to count on every rising or falling edge of the externally triggered ETR. The following figure shows the block diagram of the externally triggered input

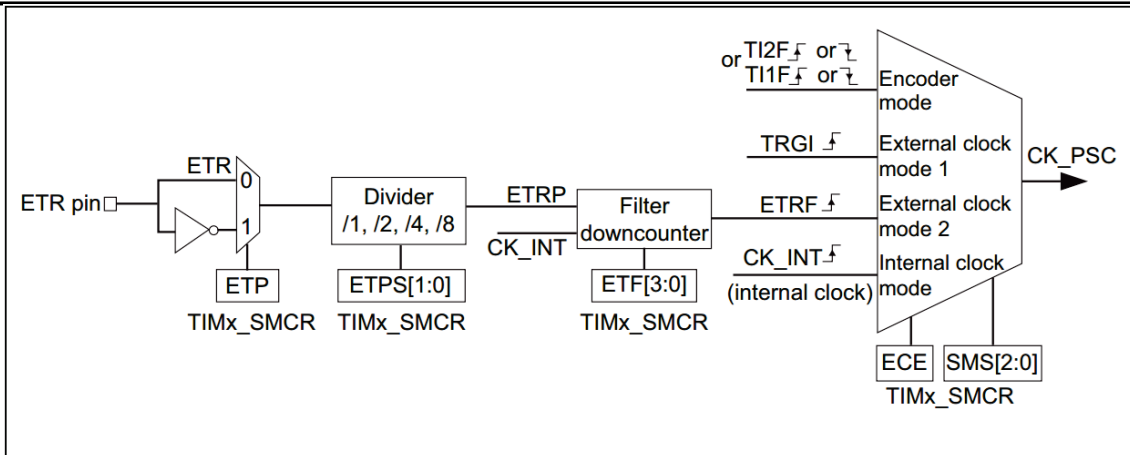


Figure 119 External Trigger Input Block Diagram

For example, to configure an up counter that counts every 2 rising edges under ETR, use the following procedure:

1. No filter is needed in this example, set  $ETF[3:0] = 0000$  in the TIMx\_SMCR registers
2. Set the prescaler, set  $ETPS[1:0] = 01$  in the TIMx\_SMCR registers
3. Set the detection on the rising edge of the ETR, set  $ETP = 0$  in the TIMx\_SMCR register
4. Enable external clock mode 2, set  $ECE = 1$  in TIMx\_SMCR register
5. Start the counter by setting  $CEN = 1$  in the TIMx\_CR1 register. The counter counts every 2 ETR rising edges.

The delay between the rising edge of ETR and the actual clock of the counter depends on the resynchronization circuitry at the ETRP signal end.

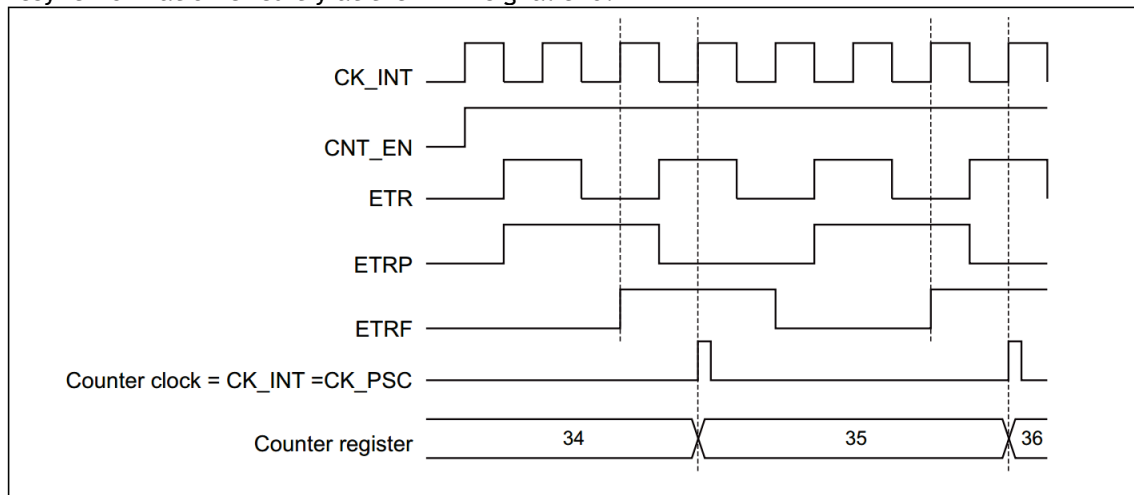


Figure 120 Control Circuit in External Clock Mode 2

### 14.3.4 Capture/Compare Channel

Each capture/compare channel is centered around a capture/compare register (containing shadow registers), including the input portion of the capture

(digital filtering, multiplexing, and prescaler), and the output section (comparator and output control). The following diagrams give an overview of the capture/compare channels.

The input section samples the corresponding TIx input signal and generates a filtered signal TIxF. An edge detector with polarity selection then generates a signal (TIxFPx) that can be used as an input trigger from the mode controller or as a capture control. This signal is pre-divided into the capture register (ICxPS).

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.

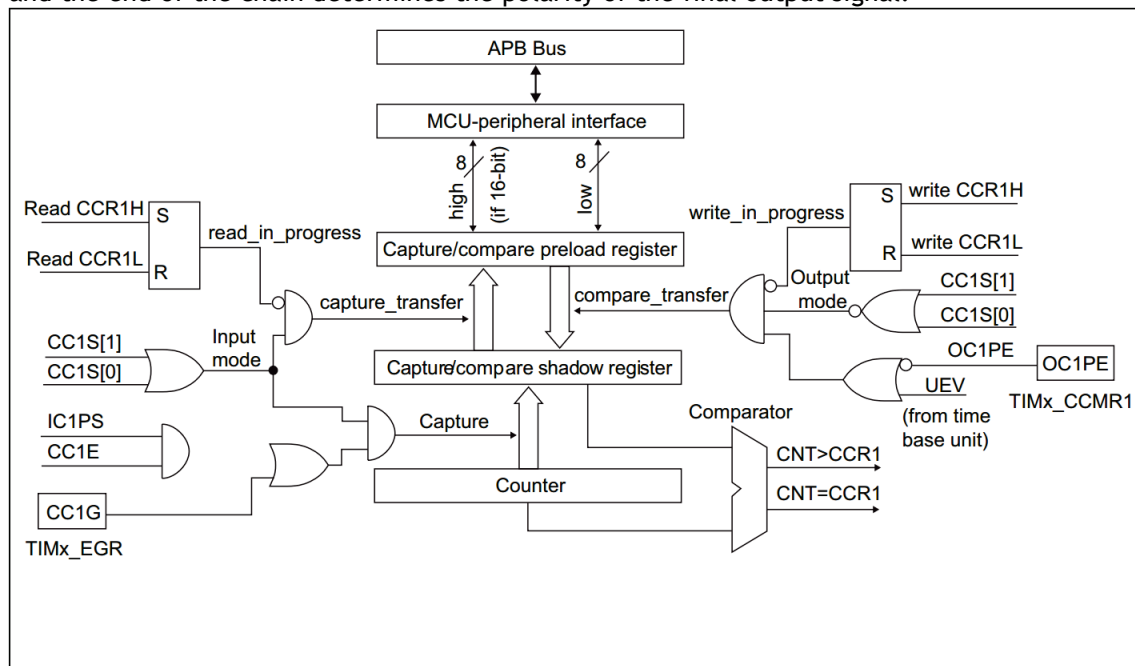


Figure122 Main circuit for capture/compare channel 1

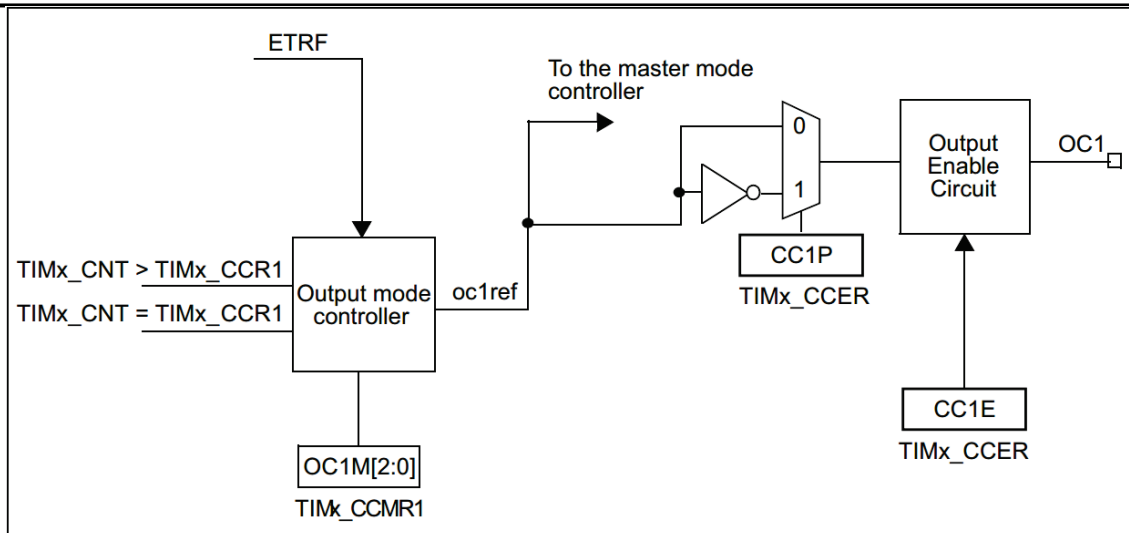


Figure123 Output section of the capture/compare channel (channel 1)

The capture/compare module consists of a preloaded register and a shadow register. The read/write process operates only on the preloaded register.

In capture mode, the capture occurs on the shadow registers, which are then copied to the preloaded registers.

In compare 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.

### 14.3.5 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 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 operation is generated if enabled. If the CCxIF flag is already high when the capture event occurs, the repeat capture flag CCxOF (TIMx\_SR register) is set to '1'. CCxIF is cleared by writing CCxIF=0, or by reading the capture data stored in the TIMx\_CCRx register. Write 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 must configure the filter bandwidth to be longer than 5 clock cycles. Therefore we can sample the input signal 8 times (at fDTS frequency) to confirm a true edge shift on TI1, i.e., by writing IC1F=0011 in the TIMx\_CMR1 register.
- To select the active conversion edge of the TI1 channel, write 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 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 and CC1IF has not 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 in order to avoid losing capture overflow information that may be generated after the capture flag is read and before the data is read.

*Notes: Setting the corresponding CCxG bit in the TIMx\_EGR register allows you to generate input capture interrupts and/or DMA requests through software.*

### 14.3.6 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, you need to measure the length (TIMx\_CCR1 register) and duty cycle (TIMx\_CCR2 register) of the PWM signal input to TI1 as follows (depending on the frequency of CK\_INT and the value of the prescaler)

- To select a valid input for TIMx\_CCR1: Set CC1S=01 in the TIMx\_CMR1 register (selects TI1).
- 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 a valid input for TIMx\_CCR2: Set CC2S=10 in the TIMx\_CMR1 register (selects TI1).
- 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

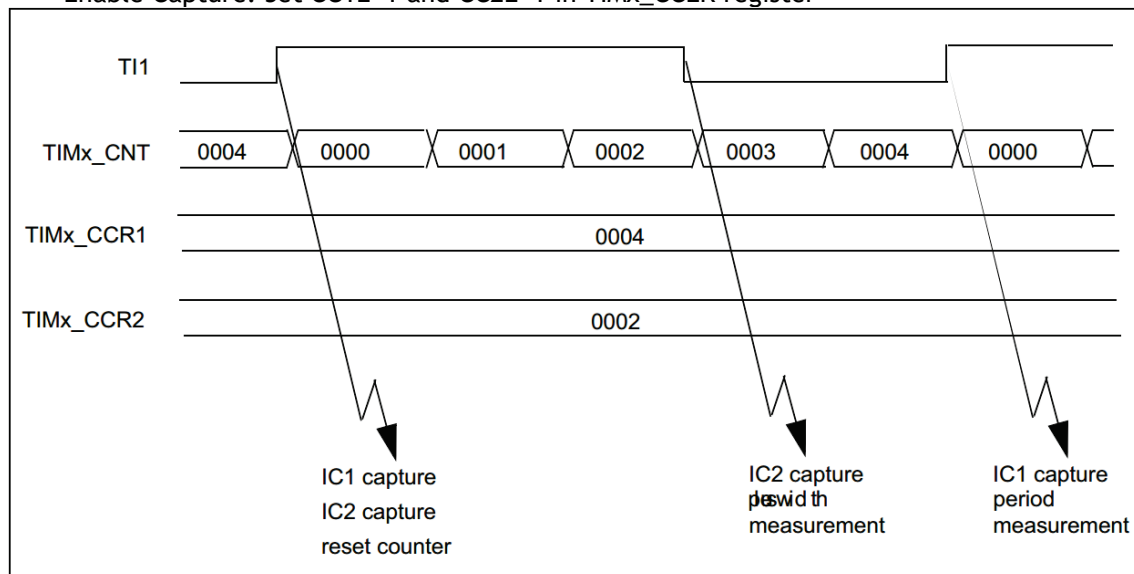


Figure 124 PWM Input Mode Timing

Since only TI1FP1 and TI2FP2 are connected to the Slave Mode Controller, only TIMx\_CH1/TIMx\_CH2 signals can be used for PWM input mode.

### 14.3.7 Forced Output Mode

In output mode (CCxS=00 in the TIMx\_CMRx register), the output compare signals (OCxREF and the corresponding OCx) can be forced to active or inactive state directly by the software, independently of the result of the comparison between the output compare register and the counter.

The output compare signal (OCxREF/OCx) is forced to active by setting the corresponding OCxM=101 in the TIMx\_CMRx register. In this way, OCxREF is forced high (OCxREF is always active high), and OCx gets the value of CCxP polarity bit reversed.

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 registers and counters is still performed and the corresponding flags are modified. Therefore the corresponding interrupts

and DMA requests are still generated. This will be described in the Output Compare Mode section below.

### 14.3.8 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. 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) when comparing matches.
- 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 compare mode, the update event UEV has no effect on the OCxREF and OCx outputs.

Synchronization can be done with an accuracy of up to one counting cycle of the counter. The 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 counter clock (internal, external, prescaler)
2. Write the corresponding data to the TIMx\_ARR and TIMx\_CCRx registers
3. To generate an interrupt request and/or a DMA request, set the CCxIE bit and/or the CCxDE bit.
4. To select the output mode, e.g. to flip the OCx output pins when counter CNT matches CCRx, CCRx preload is not used, turn on the OCx outputs and high is active, you must set OCxM='011', OCxPE='0', CCxP='0' and CCxE='1'.
5. Setting the CEN bit of the TIMx\_CR1 register starts the counter

The TIMx\_CCRx register can be updated by software at any time to control the output waveform, provided that no preloaded registers are used (OCxPE='0', otherwise the TIMx\_CCRx shadow register can only be updated when the next update event occurs). An example is given in the following figure.

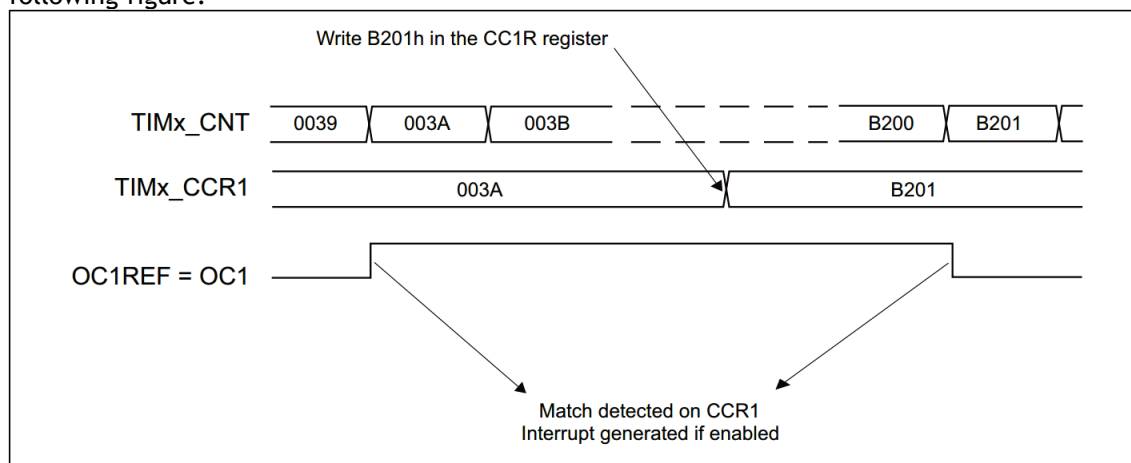


Figure125 Output Compare Mode, Flip OC1

### 14.3.9 PWM Mode

Pulse width modulation mode can generate 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 each of the OCx output channels to generate one PWM. the OCxPE bit of the TIMx\_CMRx register must be set to enable the corresponding preload registers, and finally the ARPE bit of the TIMx\_CR1 register must be set (in up-count or center-symmetric mode) to enable the auto-reload preload registers. Finally, the ARPE bit of the TIMx\_CR1 register must be set to enable the auto-reload preload register (in count-up or center-symmetric mode).

The 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 .

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 CCxE bit in the TIMx\_CCER register controls the OCx output enable. The CCxE bit in the TIMx\_CCER register controls the OCx output enable. See the description of the TIMx\_CCERx register for details.

In PWM mode (Mode 1 or Mode 2), TIMx\_CNT and TIMx\_CCRx are always compared (based on the counter count direction) to determine if  $TIMx\_CCRx \leq TIMx\_CNT$  or  $TIMx\_CNT \leq TIMx\_CCRx$  is satisfied. However, in order to be consistent with the function of OCREF\_CLR (an external event on the ETR signal clears OCxREF before the next PWM cycle), the OCxREF signal can only be generated under the following conditions:

- When the result of the comparison changes
- When the output compare mode (OCxM bit in the TIMx\_CMRx register) switches from "frozen" (no compare, OCxM='000') to some PWM mode (OCxM='110' or '111').

This allows the PWM output to be forced by software during operation.

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. Refer to section0 . The following is an example of PWM mode 1. The PWM signal reference OCxREF is high when  $TIMx\_CNT < TIMx\_CCRx$  and low . If the comparison value in TIMx\_CCRx is greater than the auto-reload value (TIMx\_ARR), OCxREF remains '1'.

If the comparison value is 0, OCxREF is held at '0'. The following figure shows an example of an edge-aligned PWM waveform with TIMx\_ARR=8.

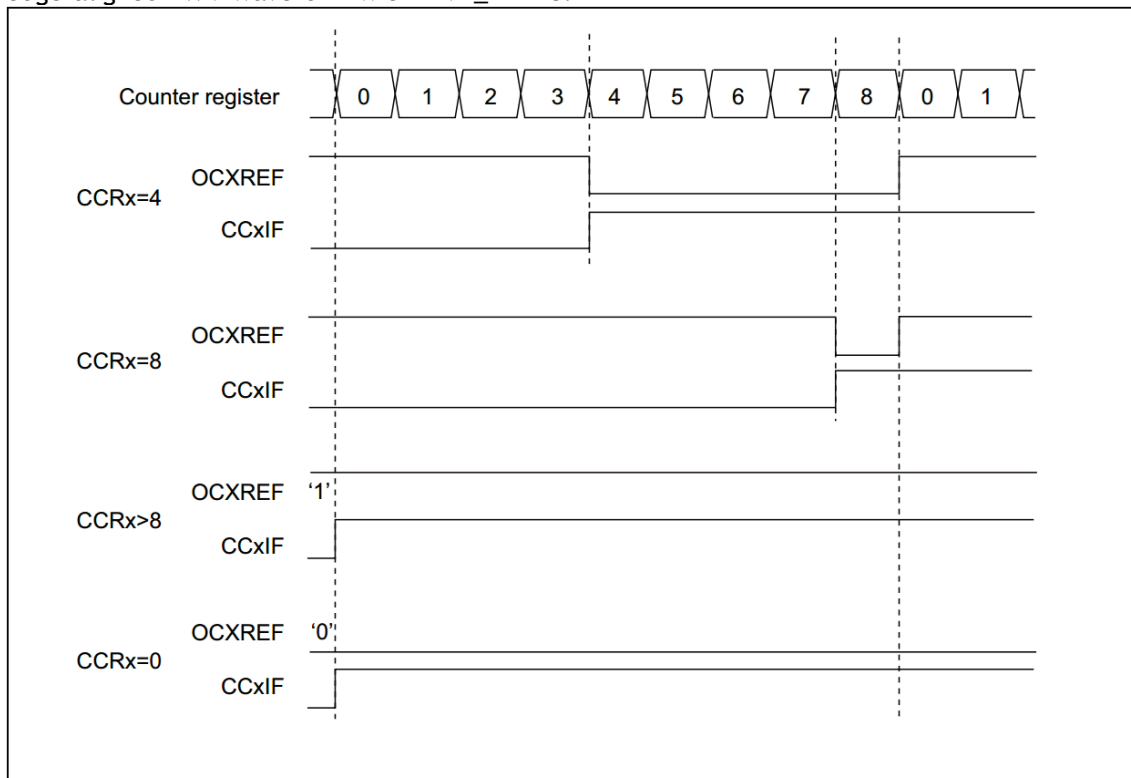


Figure126 Edge-aligned PWM waveform (ARR=8)

#### Down Count Configuration

Performs a down count when the DIR bit of the TIMx\_CR1 register is high. Refer to section0 . 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 kept as '1'. A 0% PWM waveform be generated in this mode.

## PWM Central Alignment Mode

When the CMS bit in the TIMx\_CR1 register is not '00', it is center-aligned mode (all other configurations have the same effect on the OCxREF/OCx signals). Depending on the CMS bit setting, the compare flag can be set to '1' when the counter is counting up, '1' when the counter is counting down, or '1' when the counter is counting both up and down. The count direction bit (DIR) is updated by hardware; do not modify it with software. See 0 in the TIMx\_CR1 register for central alignment mode.

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 when the counter counts down in central alignment mode 1.

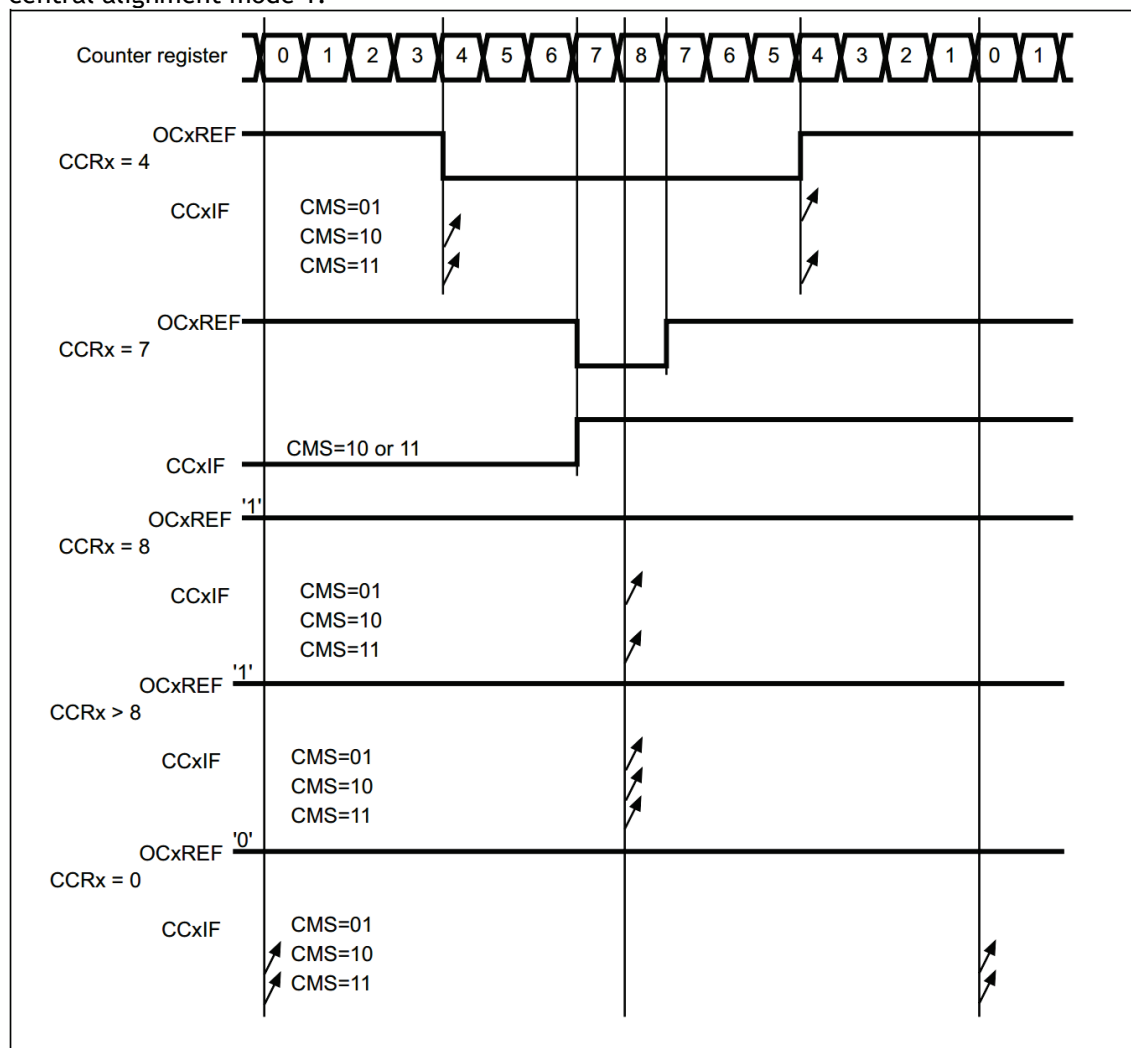


Figure127 Centrally aligned PWM waveform (APR=8)

### Tips for Using the Center Alignment Mode:

- When entering Central Alignment mode, the current count up/down configuration is used; 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, software cannot modify the DIR and CMS bits at the same time.
- It is not recommended to rewrite 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 is not updated. For example, if the counter is counting up, it continues 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.

### 14.3.10 Single Pulse Mode

One-pulse mode (OPM) is a special case of one of the many modes described above. This mode allows the counter to respond to an excitation and, after a programmable delay, generate a pulse with a programmable pulse width.

The counter can be started from the mode controller to generate waveforms in output compare mode or PWM mode. Setting the OPM bit in the TIMx\_CR1 register selects 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:  $CNT < CCRx \leq ARR$  (specifically,  $0 < CCRx$ ).

Downward counting mode:  $CNT > CCRx$

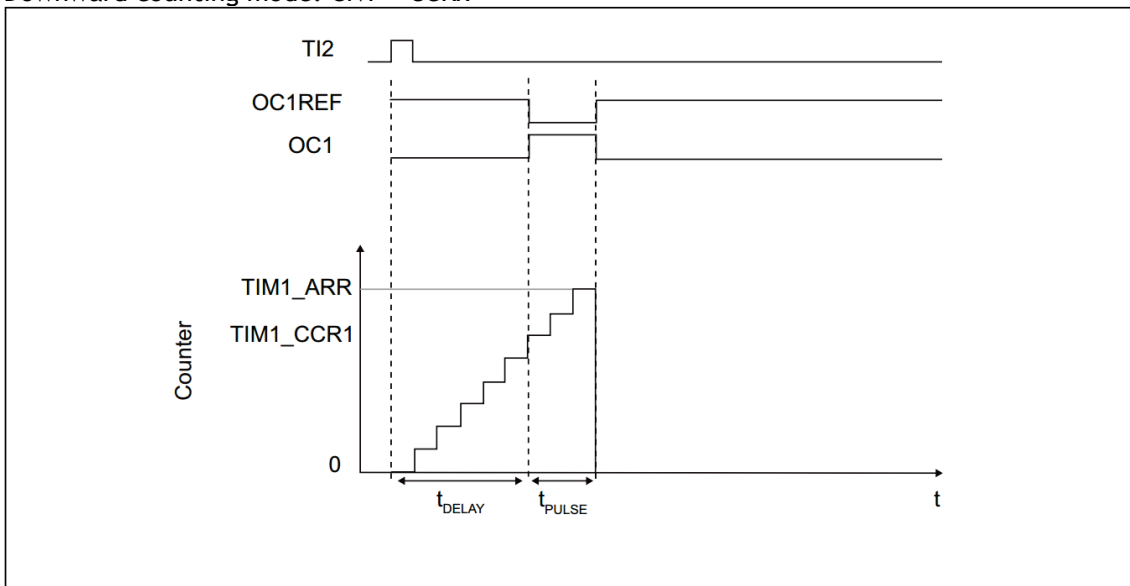


Figure128 Example of single pulse mode

For example, you need to generate a long on OC1 after delaying  $t_{DELAY}$  starting from a rising edge detected on the TI2 input pin positive pulse of degree  $t_{PULSE}$ .

Assuming 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.
- Set TS='110' in the TIMx\_SMCR register and the TI2FP2 triggers 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 the counter prescaler).
- $t_{DELAY}$  is defined by the value written to the TIMx\_CCR1 register.
- $t_{PULSE}$  is defined by the difference between the auto-load 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; first, set OC1M="111" in TIMx\_CCMR1 register to enter PWM mode 2; selectively enable the preload registers according to the needs: set OC1PE="1" in TIMx\_CMR1 and ARPE in TIMx\_CR1 register; then fill in the comparison value in TIMx\_CCR1 register, and fill in the auto-load value in TIMx\_ARR register; then fill in the auto-load value in TIMx\_CCR1 register and the auto-load value in TIMx\_ARR register; then, set the comparison value in TIMx\_CCR1 register, and fill in the auto-load value in TIMx\_ARR register. and ARPE in the TIMx\_CR1 register; then fill in the comparison value in the TIMx\_CCR1 register, fill in the auto-

load value in the TIMx\_ARR register, modify the UG bit to generate an update event, and then wait for an external trigger event on the 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 required, 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-load 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. Comparison operations between the counter and the comparison value then generate the output transition. However, these operations require a certain number of clock cycles, which limits the minimum delay time tDELAY that can be obtained.

If you want to output waveforms with minimum delay, you can set the OCxFE bit in the TIMx\_CMRx register; at this point, OCxREF (and OCx) are forced to respond to the excitation without relying on the result of the comparison anymore, and the output waveform is the same as that when the comparison is matched. OCxFE works only when the channel is configured for PWM1 and PWM2 modes.

### 14.3.11 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 at the ETRF input, and the OCxREF signal will remain low until the next update event, UEV, occurs.

This function can only be used in the output comparison and PWM modes, not in the strong-set mode.

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. The 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 goes high, corresponding to different OCxCE values. this example, timer TIMx is placed in PWM mode.

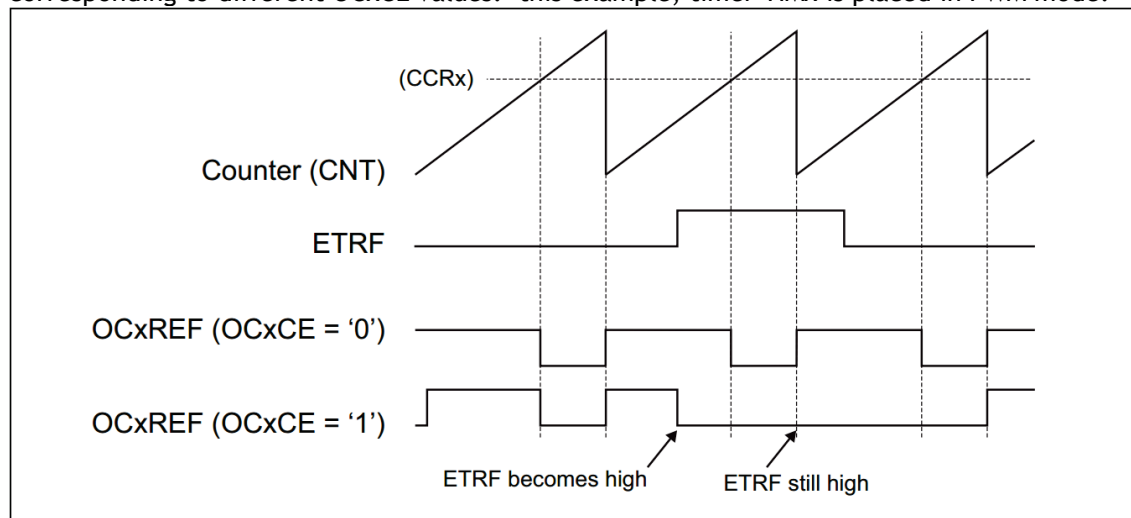


Figure129 Clear OCxREF for TIMx

### 14.3.12 Encoder Interface Mode

The encoder interface mode is selected by setting SMS=001 in the TIMx\_SMCR register if the counter only counts on the TI2 edge, SMS=010 if it only counts 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 Table75 , it is assumed that the counter is activated (CEN='1' in TIMx\_CR1 register) and 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 filters and polarity control; if there is no filter and no

phasing, then  $T11FP1 = T11$ ,  $T12FP2 = T12$ . Depending on the hopping sequence of the two input signals, count pulses and direction signals are generated. Depending on the hopping sequence of the two input signals, the counter counts up or down and the hardware sets the DIR bit of the TIMx\_CR1 register accordingly. It does not matter whether the counter counts on T11, on T12, or on both T11 and T12. A trip on either input (T11 or T12) 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 from 0 to the auto-loaded value in the TIMx\_ARR register (depending on the direction, either 0 to ARR count or ARR to 0 count). Therefore, TIMx\_ARR must be configured before counting starts; again, the captures, comparators, prescalers, trigger outputs, etc. work as usual.

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 shows all possible combinations, assuming that T11 and T12 do not change at the same time.

Table75 Relationship between count direction and encoder signal

active edge	Relative signal level (T11FP1 corresponds to T12, T12FP2 corresponds to T11)	T11FP1 signal		T12FP2 signal	
		go up	go down	go up	go down
T11 count only	your (honorific)	down-count	Upward Counting	disregard	disregard
	lower (one's head)	Upward Counting	down- count	disregard	disregard
T12 count only	your (honorific)	disregard	disregard	Upward Counting	down count
	lower (one's head)	disregard	disregard	down count	Upward Counting
Counting on T11 and T12	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 noise immunity. The third of the encoder output represents the mechanical zero point, which can be connected to an external interrupt input and trigger a counter reset.

The figure below is an example of counter operation, showing the generation of the count signal and direction control. It also shows how input jitter is suppressed when a double edge is selected; jitter may be generated when the sensor position is close to a transition point. In this example, we assume the following configuration:

- CC1S='01' (TIMx\_CMR1 register, IC1FP1 mapped to T11)
- CC2S='01' (TIMx\_CMR2 register, IC2FP2 mapped to T12)
- CC1P='0' (TIMx\_CCER register, IC1FP1 not inverted, IC1FP1=T11)
- CC2P='0' (TIMx\_CCER register, IC2FP2 not inverted, IC2FP2=T12)
- SMS='011' (TIMx\_SMCR register, all inputs are valid on rising and falling edges).
- CEN='1' (TIMx\_CR1 register, counter enable)



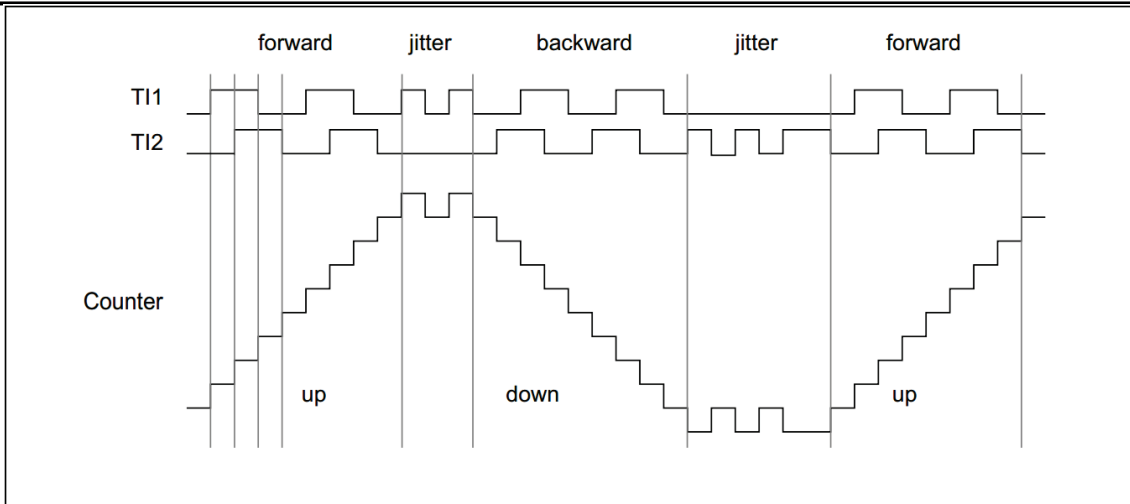


Figure130 Example of counter operation in encoder mode

The following figure shows an example of counter operation when IC1FP1 polarity is inverted (CC1P='1', other configurations are the same as the above example)

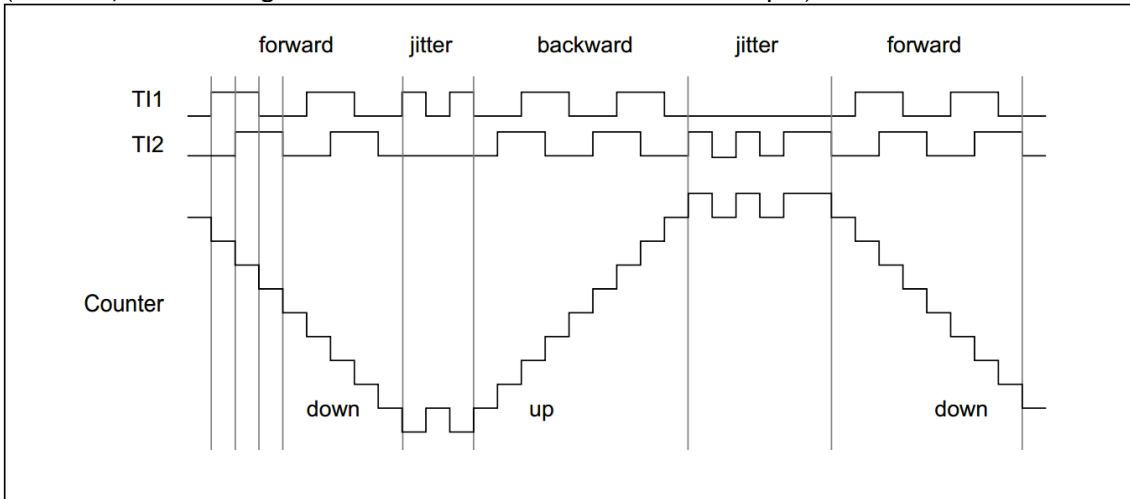


Figure131 Example of encoder interface mode for IC1FP1 inversion

When the timer is configured in encoder interface mode, it provides information about the current position of the sensor. Using second timer configured in capture mode, the interval between two encoder events can be measured to obtain dynamic information (velocity, acceleration, deceleration). The encoder output indicating the mechanical zero point can be used for this purpose. Depending on the interval between the two events, can be read out at a fixed time. If possible, you can latch the counter value to a third input capture register (the capture signal must be periodic and can be generated by another timer); it can also be read by a DMA request generated by the real-time clock.

### 14.3.13 Timer Input Heterodyne Function

The TI1S bit in the TIMx\_CR2 register allows the input filter of channel 1 to be connected to the output of an iso-gate whose 3 inputs are TIMx\_CH1, TIMx\_CH2, and TIMx\_CH3.

The heterodyne output can be used for all timer input functions such as triggering or input capture. An example of how this feature can be used to connect a Hall sensor is given in the previous chapter 13.3.18.

### 14.3.14 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 generated if the URS bit of the TIMx\_CR1 register is low; all preloaded 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 it does not need to be configured. the CC1S bit selects the input capture source only, i.e., CC1S=01 in the TIMx\_CMR1 register. set CC1P=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 of TI1 occurs; at this point, the counter is cleared to zero counts again 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.

The following figure shows the action when the auto-reload register TIMx\_ARR = 0x36. The delay the rising edge of TI1 and the actual reset of the counter depends on the resynchronization circuitry at the TI1 input.

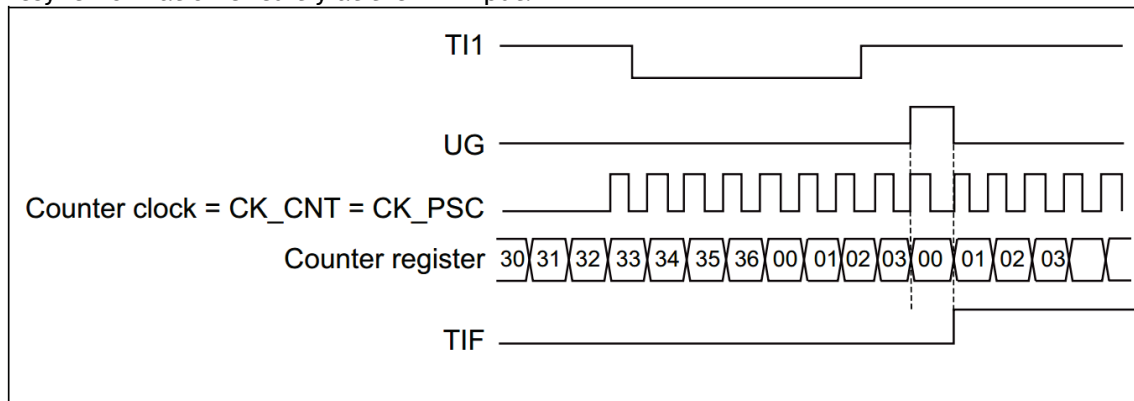


Figure132 Control circuit in reset mode

#### 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 example, 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, set CC1S=01 in the TIMx\_CMR1 register. set CC1P=1 in the TIMx\_CCER register to determine the polarity (detects a low level 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.
- Setting CEN=1 in the TIMx\_CR1 register starts the counter. In gated mode, if CEN=0, the counter will not start, regardless of the trigger input level.

The counter starts counting according to the internal clock as long as TI1 is low and stops counting when TI1 goes high. The TIF flag in TIMx\_SR is set when 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.

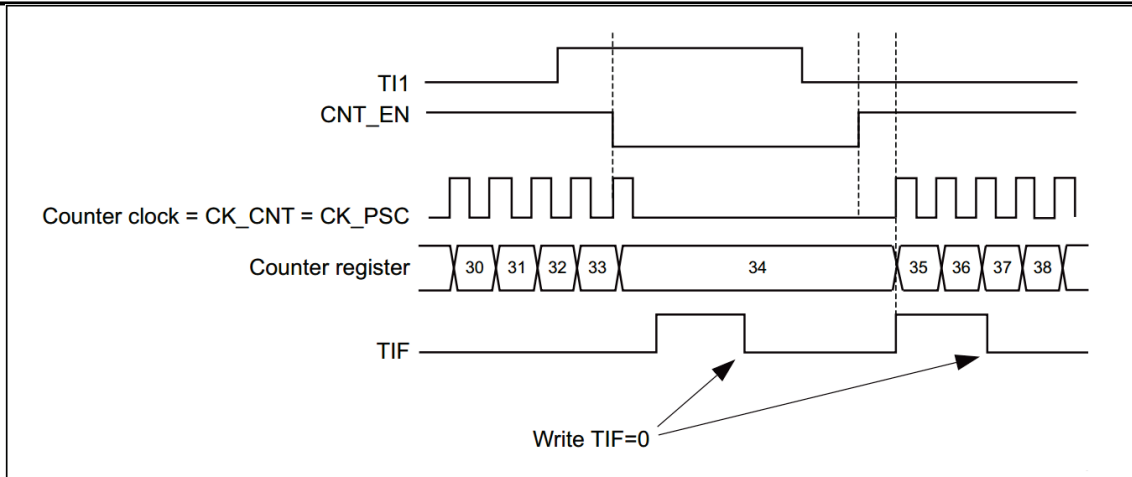


Figure133 Control circuit in gated mode

### 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 need to be configured. the CC2S bit is only used to select the input source, set CC2S=01 in the TIMx\_CMR1 register. set CC2P=1 in the TIMx\_CCER register to determine the polarity (detects low levels only).
- 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.

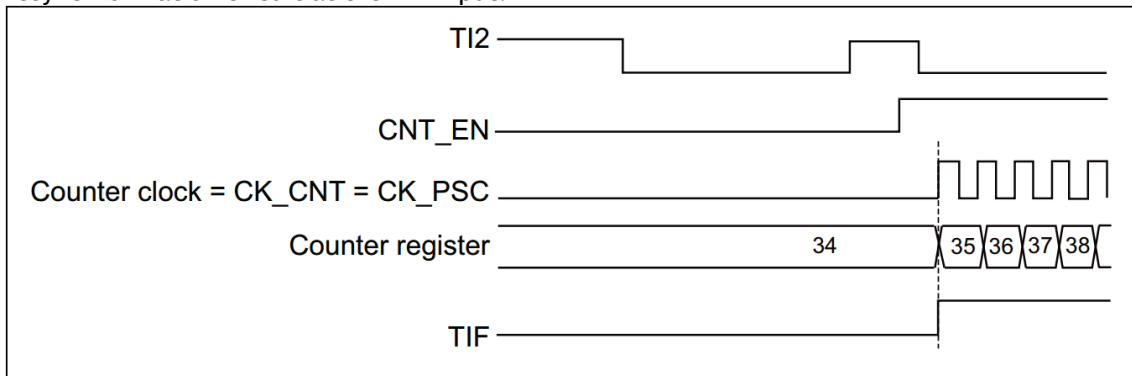


Figure134 Control Circuit in Trigger Mode

### 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 an input to the external clock, and another input can be selected as the trigger input during 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, after a rising edge occurs on TI1, the counter counts upward on every rising edge of ETR:

1. 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
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 require configuration of the
  - 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 rising edge only)
- 3. Set SMS=110 in the TIMx\_SMCR register to configure the timer for trigger mode. Set TS=101 in TIMx\_SMCR register to select TI1 as 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.

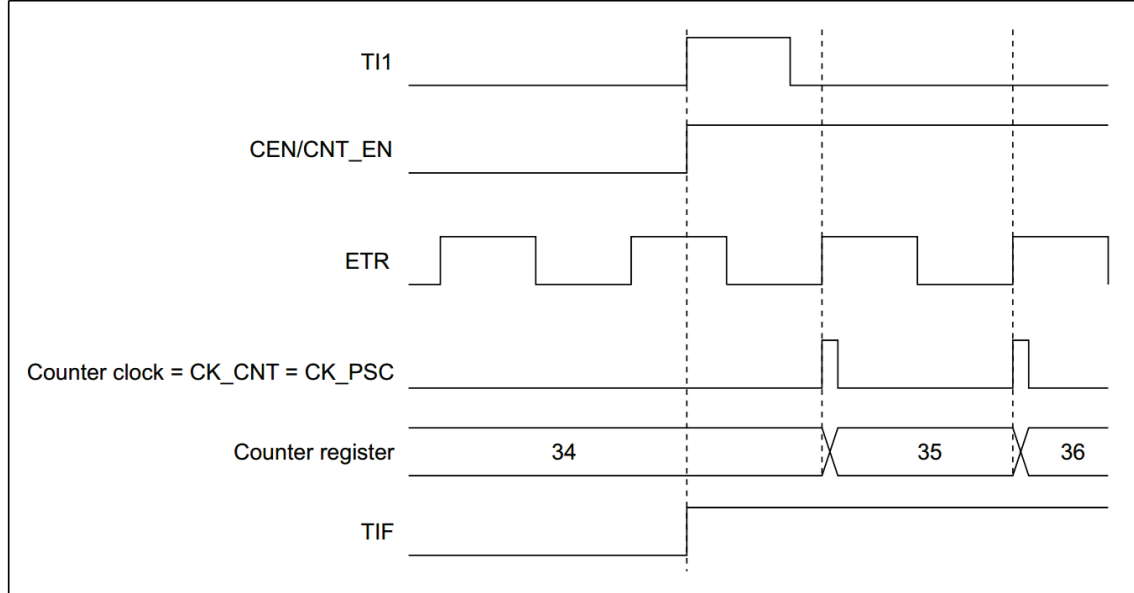


Figure135 Control Circuit in External Clock Mode 2 + Trigger Mode

### 14.3.15 Timer Synchronization

All TIMx timers are internally connected for timer synchronization or linking. When a timer is in master mode, it can reset, start, stop, or provide a clock to the counter of another timer in slave mode.

The following figure shows an overview of the Trigger Select and Master Mode Select modules.

#### Using a Timer as a Prescaler for Another Timer

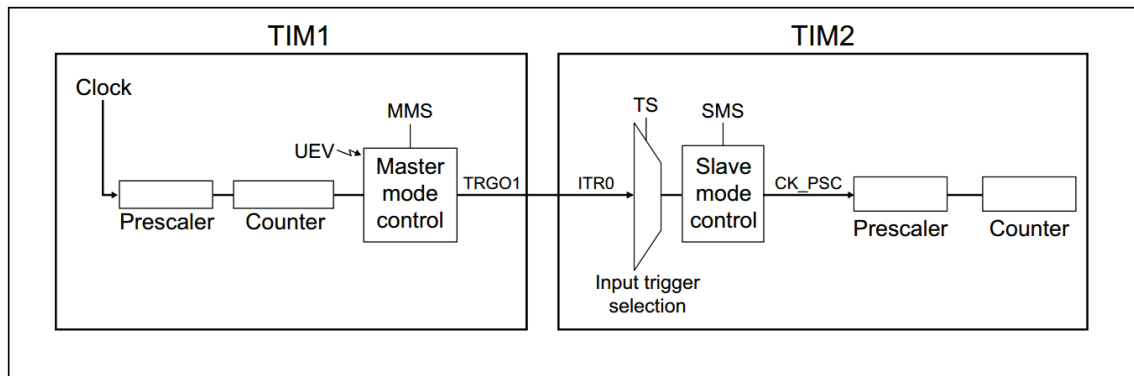


Figure136 Example of a Master/Slave Timer

For example, you can configure Timer 1 as a prescaler for Timer 2. Refer to Figure 136 for the following operation:

- Configure Timer1 as the master mode, which can output a periodic trigger signal at every update event UEV. With MMS='010' in the TIM1\_CR2 register, output a rising edge signal on TRGO1 whenever an update event is generated.
- Connect the TRGO1 output of Timer1 to Timer2, set TS='000' in the TIM2\_SMCR register, and configure Timer2 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 TIM2\_SMCR register); this allows Timer 2 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 (TIMx\_CR1 register) must be set to start each of the two timers.

Notes: 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 2.

### Using One Timer to Enable Another Timer

In this example, the enable of Timer 2 is controlled by the output comparison of Timer 1. Refer to Figure136 for connections. Timer 2 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 and send 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\_CCMR1 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 CEN=1 in TIM2\_CR1 register to enable Timer 2
- Set CEN=1 in the TIM1\_CR1 register to start Timer 1

Notes: Timer 2's clock is not synchronized with Timer 1's clock, and this mode only affects the enable signal of the Timer 2 counter.

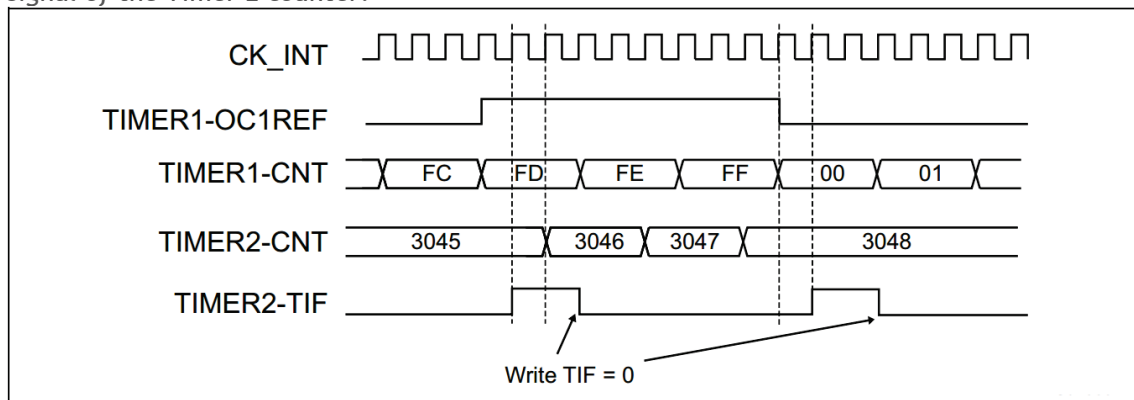


Figure137 OC1REF of Timer1 controls Timer2

In the example at Figure137, before Timer 2 is started, their counters and prescalers are not initialized, so they start counting from the current value. 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 timers can be reset by writing the UG bit of the TIMx\_EGR register.

In the next example, it is necessary to synchronize Timer 1 and Timer 2. 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 to be the master mode and send the Output Compare 1 reference signal (OC1REF) as the trigger output (MMS=100 in the TIM1\_CR2 register).
- Configure the OC1REF waveform for Timer 1 (TIM1\_CCMR1 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.

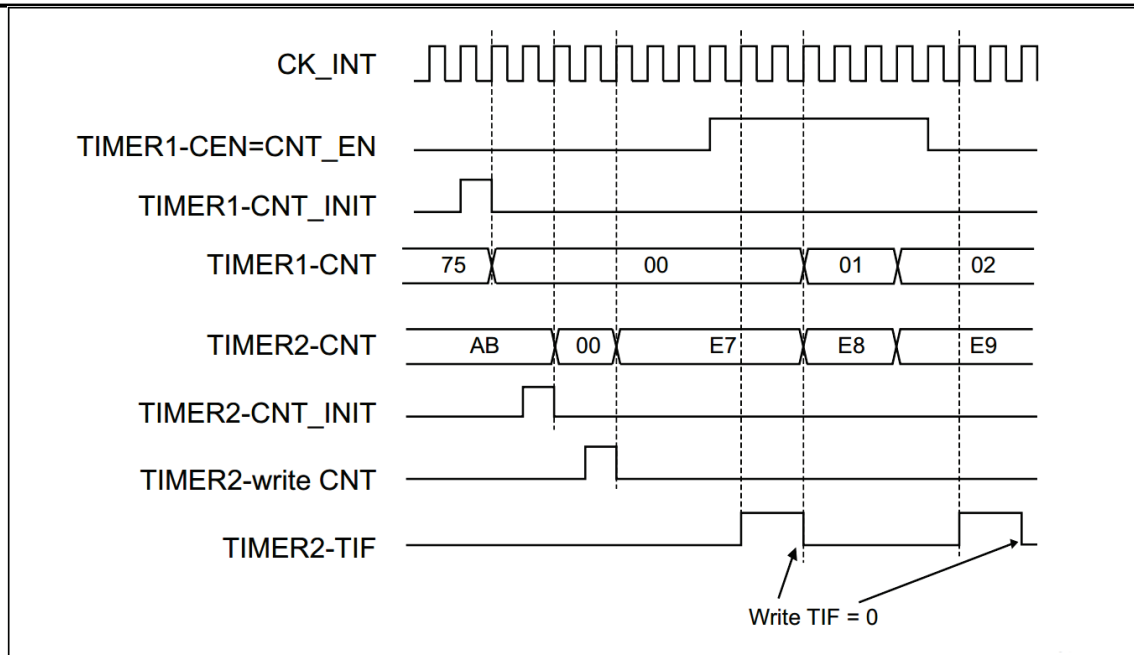


Figure138 Timer 2 can be controlled by enabling Timer 1

### Using a Timer to Start Another Timer

In this example, Timer 2 is enabled using an update event from Timer 1. Refer to Figure 136 for connections. Once Timer 1 generates an update event, Timer 2 starts counting from its current value (which can be non-zero) according to the divided internal clock. On receipt of a trigger signal, the CEN bit of Timer 2 is automatically set to '1' and the counter starts counting until a '0' is written to the CEN bit of the TIM2\_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 out its update event (UEV) as a trigger output (MMS=010 in the TIM1\_CR2 register).
- Configure the period of Timer 1 (TIM1\_ARR register).
- Configure Timer 2 to get an input trigger from Timer 1 (TS=000 in the TIM2\_SMCR register)
- Configure Timer 2 for trigger mode (SMS=110 in the TIM2\_SMCR register)
- Set CEN=1 in the TIM1\_CR1 register to start Timer 1.

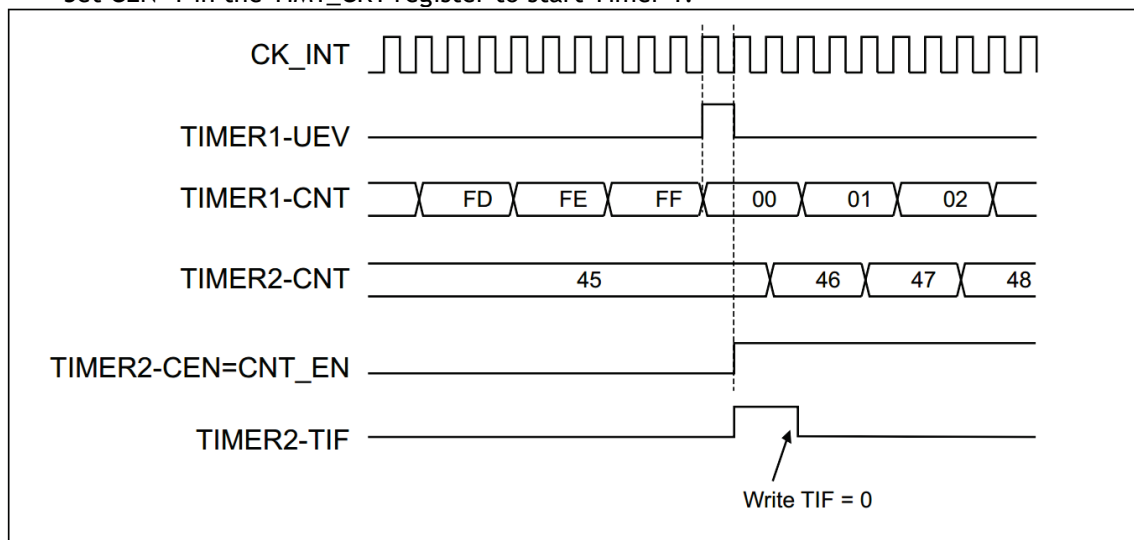


Figure139 Triggering Timer 2 using an update from Timer 1

In the previous example, it is possible to initialize both counters before starting the count. Figure 140 shows the action in the same configuration as 0, using trigger mode instead of gated mode (SMS=110 in the TIM2\_SMCR register).

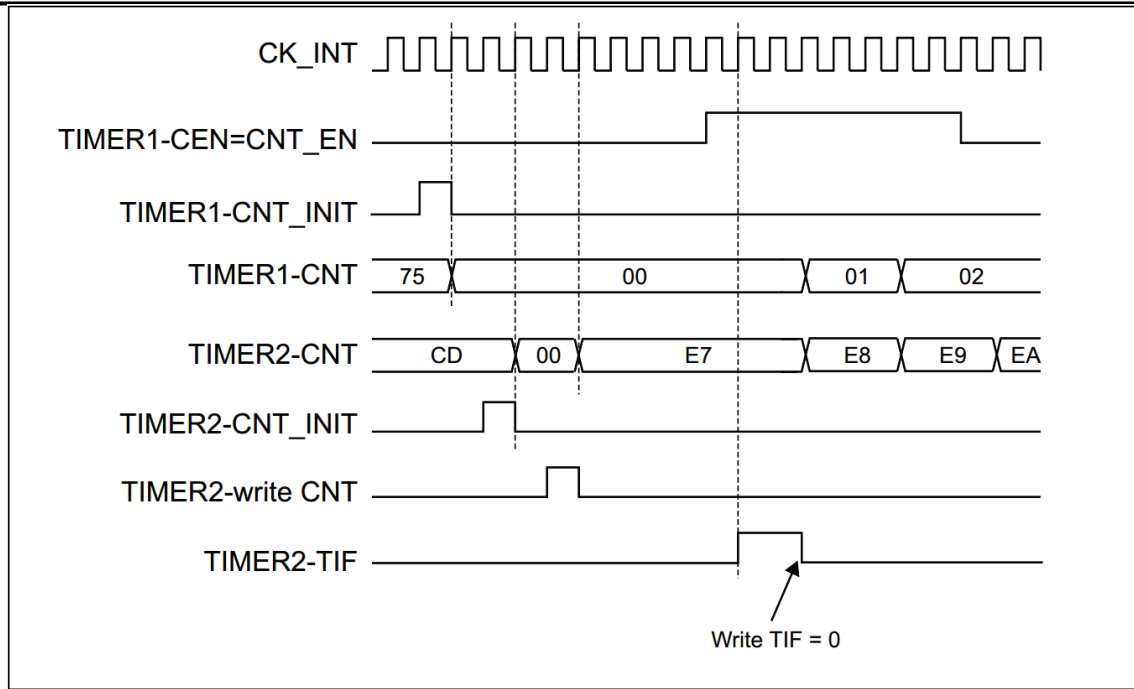


Figure140 Triggering Timer 2 using Timer 1's Enable

#### Using One Timer as a Prescaler for the Other

This example uses Timer 1 as a prescaler for Timer 2. Referring to Figure136 for the connections, the configuration is as follows:

- Configure Timer 1 to be the master mode, send its update event UEV as a trigger output (MMS='010' in the TIM1\_CR2 register). Then output a cycle signal each time the counter overflows.
- Configure the period of Timer 1 (TIM1\_ARR register).
- Configure Timer 2 to get an input trigger from Timer 1 (TS=000 in the TIM2\_SMCR register)
- Configure Timer 2 to use external clock mode (SMS=111 in the TIM2\_SMCR register)
- Set CEN=1 in the TIM1\_CR2 register to start Timer 2.
- Set CEN=1 in the TIM1\_CR1 register to start Timer 1.

#### Synchronized Start of 2 Timers Using an External Trigger

In this example Timer1 is enabled when the TI1 input of Timer1 rises, and Timer2 is enabled at the same time as Timer1, see Figure136. To ensure counter alignment, Timer 1 must be configured in master/slave mode (corresponding to TI1 as slave and Timer 2 as master):

- Configure Timer 1 to be the master mode and 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 2 to get an input trigger from Timer 1 (TS=000 in the TIM2\_SMCR register)
- Configure Timer 2 for trigger mode (SMS='110' for TIM2\_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.

**Notes:** In this example, both timers are initialized (set the appropriate UG bit) before startup, and both counters start at 0, but an offset can be inserted between the timers by writing to any of the counter registers (TIMx\_CNT). In the figure below you can see that in master/slave mode there is a delay between CNT\_EN and CK\_PSC in timer 1.

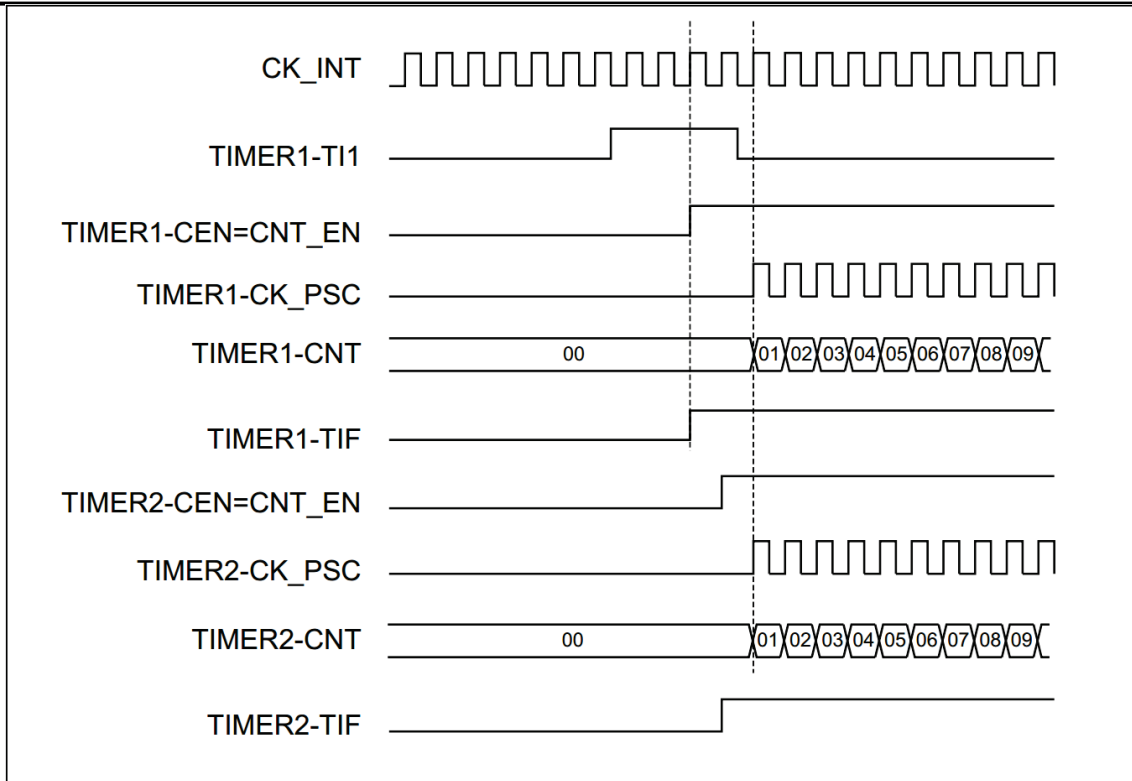


Figure141 Triggering Timer 1 and Timer 2 using the TI1 input of Timer 1

### 14.3.16 Debug Mode

When the microcontroller enters debug mode (the Cortex-M3 core is stopped), the TIMx counter either continues normal operation or stops, depending on the DBG\_TIMx\_STOP setting in the DBG module. For more details, see the subsequent section 31.16.2 : Debugging bxCAN and I2C support for timer, watchdog, .

## 14.4 TIM2 to TIM5 Register Descriptions

See Section 1 for details on the abbreviations used in the register descriptions. These peripheral registers can be operated in half-word (16-bit) or word (32-bit) format.

### 14.4.1 TIM2 to TIM5 Control Register 1 (TIMx\_CR1)

Offset address: 0x00

Reset value: 0x0000

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

Bit	notation	clarification
15:10	Reserved	Reserved, always reads 0.
9:8	CKD[1:0]	CKD[1:0]: Clock division factor (Clock division) Defines the dividing ratio between the timer clock (CK_INT) frequency and the sampling frequency used by the digital filter (ETR, TIx). 00: tDTS=tCK_INT 01: tDTS=2xtCK_INT 10: tDTS=4xtCK_INT 11: Reserved
7	ARPE	ARPE: Auto-reload preload enable bit 0: TIMx_ARR register is not buffered; 1: The TIMx_ARR register is loaded into the buffer.
6:5	CMS[1:0]	CMS[1:0]: Center-aligned mode selection 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. The output compare interrupt flag bit for the channel configured as an output (CCxS=00 in the TIMx_CMRx register) is set only when the counter is counting down.

		<p>10: Central Alignment Mode 2. the counter alternately counts up and down. The output compare interrupt flag bit for the channel configured as an output (CCxS=00 in the TIMx_CMRx register) is set only when the counter counts up.</p> <p>11: Central Alignment Mode 3. the counter alternately counts up and down. The output compare interrupt flag bit for the channel configured as an output (CCxS=00 in the TIMx_CMRx register) is set for both upward and downward counter counts.</p> <p>Note: With the counter on (CEN=1), switching from edge-aligned mode to center-aligned mode is not allowed.</p>
4	DIR	<p><b>DIR:</b> Direction</p> <p>0: Counter counts up;</p> <p>1: The counter counts down.</p> <p>Note: This bit is read-only when the counter is configured for Central Alignment Mode or Encoder Mode.</p>
3	OPM	<p><b>OPM:</b> Onepulse mode</p> <p>0: The 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	<p><b>URS:</b> Update request source The software selects the source of the UEV event with this bit.</p> <p>0: If an update interrupt or DMA request is enabled, either of the following events generates an update interrupt or DMA request:</p> <ul style="list-style-type: none"> <li>-Counter overflow/underflow</li> <li>-Setting the UG bit</li> <li>-Updates generated from the mode controller</li> </ul> <p>1: If the update interrupt or DMA request is enabled, the update interrupt or DMA request is generated only for counter overflow/underflow.</p>
1	UDIS	<p><b>UDIS:</b> Update disable</p> <p>Software allows/disallows the generation of UEV events with this bit</p> <p>0: UEV is allowed. the update (UEV) event is generated by any of the following events:</p> <ul style="list-style-type: none"> <li>-Counter overflow/underflow</li> <li>-Setting the UG bit</li> <li>-Updates generated from the mode controller</li> </ul> <p>Registers with caches are loaded with their preloaded values. (Translation: updating shadow registers)</p> <p>1: Disable UEV. no update event is generated and the shadow registers (ARR, PSC, CCRx) maintain their values. If the UG bit is set or a hardware reset is issued from the mode controller, the counters and prescalers are reinitialized.</p>
0	CEN	<p><b>CEN:</b> Enable Counter</p> <p>0: Disable the counter;</p> <p>1: Enable the counter.</p> <p>Note: External Clock, Gated Mode and Encoder Mode will not work until the CEN bit is set in software. Trigger mode can automatically set the CEN bit in hardware. In single pulse mode, CEN is automatically cleared when an update event occurs.</p>

## 14.4.2 TIM2 to TIM5 Control Register 2 (TIMx\_CR2)

Offset address: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TI1S	MMS[2:0]			CCDS	Reserved		
								rw	rw	rw	rw	rw			

Bit	notation	clarification
15:8	Reserved	Reserved, always reads 0.
7	TI1S	<p><b>TI1S:</b> TI1 Selection</p> <p>0: The TIMx_CH1 pin is connected to the TI1 input;</p> <p>1: The TIMx_CH1, TIMx_CH2, and TIMx_CH3 pins are connected to the TI1 inputs via heterodyne. See the section Interfacing with Hall Sensors in the previous chapter13.3.18 .</p>
6:4	MMS[2:0]	<p><b>MMS[2:0]:</b> Master mode selection</p> <p>These 3 bits are used to select the synchronization information (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 TIMx_EGR register is used as a trigger output (TRGO). If the reset is generated by a trigger input (from a mode controller in reset mode), the signal on TRGO is delayed relative to the actual reset.</p> <p>001: The enable-counter enable signal CNT_EN is used as a trigger output (TRGO). Sometimes it is necessary to start several 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-Update events are selected as trigger inputs (TRGO). For example, a master timer clock can be used as a prescaler for a slave timer.</p>



		011: Compare Pulse - When a capture or a successful comparison occurs, the trigger output sends a positive pulse (TRGO) when the CC1IF flag is to be set (even if it is already high). 100: The compare-OC1REF signal is used as a trigger output (TRGO).101: The compare-OC2REF signal is used as a trigger output (TRGO).110: The compare-OC3REF signal is used as a trigger output (TRGO). 111: The compare-OC4REF signal is used as a trigger output (TRGO).
3	CCDS	CCDS: 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.
2:0	Reserved	Reserved, always reads 0.

### 14.4.3 TIM2 to TIM5 Slave Mode Control Register (TIMx\_SMCR)

Offset address: 0x08

Reset value: 0x0000

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]		Reserved		SMS[2:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

Bit	notation	clarification
15	ETP	<b>ETP:</b> External trigger polarity This bit selects whether to use ETR or the inverse of ETR as the trigger operation 0: ETR is not inverted and is active high or on the rising edge; 1: ETR is inverted, active low or falling edge.
14	ECE	<b>ECE:</b> External clock enable bit This bit enables external clock mode 2 0: Disable external clock mode 2; 1: Enable external clock mode 2. the counter is driven by any valid edge on the ETRF signal. 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). Note 2: The following slave modes can be used in conjunction with External Clock Mode 2: Reset Mode, Gated Mode, and Trigger Mode; however, in this case TRGI cannot be connected to ETRF (TS bit cannot be '111'). Note 3: When External Clock Mode 1 and External Clock Mode 2 are both enabled, the external clock input is ETRF.
13:12	ETPS[1:0]	<b>ETPS[1:0]:</b> External trigger prescaler The frequency of the external trigger signal ETRP must be at most 1/4 of the frequency of CK_INT. when a faster external clock is input, the frequency of ETRP can be lowered using prescaling. 00: Turns off the prescaler; 01: ETRP frequency divided by 2; 10: ETRP frequency divided by 4; 11: ETRP frequency divided by 8.
11:8	ETF[3:0]	<b>ETF[3:0]:</b> External trigger filter 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 to produce a jump in the output. 0000: no filter, sampled at fDTS 0001: Sampling frequency fSAMPLING=fDTS/8, N=6 0010: Sampling frequency fSAMPLING=fDTS/8, N=8 0011: Sampling frequency fSAMPLING=fDTS/16, N=5 0100: Sampling frequency fSAMPLING=fDTS/16, N=6 0101: Sampling frequency fSAMPLING=fDTS/2, N=6 0110: Sampling frequency fSAMPLING=fDTS/2, N=8 0111: Sampling frequency fSAMPLING=fDTS/32, N=5 1000: Sampling frequency fSAMPLING=fDTS/32, N=6 1001: Sampling frequency fSAMPLING=fDTS/32, N=8 1010: Sampling frequency fSAMPLING=fDTS/4, N=6 1011: Sampling frequency fSAMPLING=fDTS/4, N=8 1100: Sampling frequency fSAMPLING=fDTS/4, N=6 1101: Sampling frequency fSAMPLING=fDTS/4, N=8 1110: Sampling frequency fSAMPLING=fDTS/4, N=6 1111: Sampling frequency fSAMPLING=fDTS/4, N=8
7	MSM	<b>MSM:</b> Master/slave mode 0: No effect; 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.
6:4	TS[2:0]	<b>TS[2:0]:</b> Trigger selection These 3-bit selections are used to synchronize the trigger input of the

		counter. 000: Internal trigger 0 (ITR0), TIM1 001: Internal Trigger 1 (ITR1), TIM2 010: Internal Trigger 2 (ITR2), TIM3 011: Internal Trigger 3 (ITR3), TIM4	100: Edge detector for TI1 (TI1F_ED) 101: Filtered Timer Input 1 (TI1FP1) 110: Filtered Timer Input 2 (TI2FP2) 111: External Trigger Input (ETRF)
		See  Table76 for details on ITRx in each timer. Note: These bits can only be changed when they are not used (e.g. SMS=000) to avoid false edge detection when changed.	
3	Reserved	Reserved, always reads 0.	
2:0	SMS[2:0]	<b>SMS[2:0]: Slave mode selection</b> 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 Input Control Register and Control Register) 000: Off Slave Mode - If CEN=1, the prescaler is driven directly from 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 - Depending on the level of TI2FP2, the counter counts up/down on the edge of TI1FP1. 011: Encoder Mode 3 - Depending on the input level of another signal, the counter counts up/down on the edges of TI1FP1 and TI2FP2. 100: Reset Mode-The rising edge of the selected trigger input (TRGI) reinitializes the counter and generates a signal to update the register. 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 are controlled. 110: Trigger Mode - 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. 111: External Clock Mode 1-The rising edge of the selected trigger input (TRGI) drives the counter. Note: Do not use the 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 gating mode is to check the level of the trigger input.	

Table76 TIMx Internal Trigger Connections

From Timer	ITR0(TS=000)	ITR1 (TS=001)	ITR2 (TS=010)	ITR3 (TS=011)
TIM2	TIM1	TIM8	TIM3	TIM4
TIM3	TIM1	TIM2	TIM5	TIM4
TIM4	TIM1	TIM2	TIM3	TIM8
TIM5	TIM2	TIM3	TIM4	TIM8

#### 14.4.4 TIM2 to TIM5 DMA/Interrupt Enable Register (TIMx\_DIER)

Offset address: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserv ed	TDE	Reserv ed	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Reserv ed	TIE	Reserv ed	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw		rw	rw	rw	rw	rw		rw		rw	rw	rw	rw	rw

Bit	notation	clarification
15	Reserved	Reserved, always reads 0.
14	TDE	<b>TDE:</b> Trigger DMA request enable 0: Disable triggering of DMA request; 1: Allow triggering of DMA requests.
13	Reserved	Reserved, always reads 0.
12	CC4DE	<b>CC4DE:</b> Capture/Compare4 DMA request enable 0: Capture/compare 4 DMA requests are disabled; 1: Allow capture/compare 4 DMA requests.
11	CC3DE	<b>CC3DE:</b> Capture/Compare3 DMA request enabled 0: Capture/compare 3 DMA requests are disabled; 1: Allows capture/comparison of DMA requests for 3.
10	CC2DE	<b>CC2DE:</b> Capture/Compare2 DMA request enabled 0: Capture/compare 2 DMA requests are disabled; 1: Allow capture/compare 2 DMA requests.

9	CC1DE	CC1DE: Capture/Compare1 DMA request enabled 0: Capture/compare 1 DMA requests are disabled; 1: Allow capture/compare 1 for DMA requests.
8	UDE	UDE: update DMA request enable 0: DMA requests for updates are disabled; 1: Allow updated DMA requests.
7	Reserved	Reserved, always reads 0.
6	TIE	TIE: Trigger interrupt enable 0: Disable triggering of interrupts; 1: Enable trigger interrupt.
5	Reserved	Reserved, always reads 0.
4	CC4IE	CC4IE: Allow Capture/Compare4 interrupt enable 0: Capture/compare 4 interrupt disabled; 1: Allow capture/compare 4 interrupts.
3	CC3IE	CC3IE: Allow Capture/Compare3 interrupt enable 0: Capture/Compare 3 interrupt disabled; 1: Capture/compare 3 interrupts are allowed.
2	CC2IE	CC2IE: Allow Capture/Compare2 interrupt enable 0: Capture/Compare 2 interrupt disabled; 1: Allow capture/compare 2 interrupts.
1	CC1IE	CC1IE: Allow Capture/Compare1 interrupt enable 0: Capture/Compare 1 interrupt disabled; 1: Allow capture/compare 1 interrupt.
0	UIE	UIE: update interrupt enable 0: Disable update interrupt; 1: Allow updates to be interrupted.

#### 14.4.5 TIM2 to TIM5 Status Register (TIMx\_SR)

Offset address: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	CC4OF	CC3OF	CC2OF	CC1OF	Reserved	TIF	Reserved	CC4IF	CC3IF	CC2IF	CC1IF	UIF			
	rc_w0	rc_w0	rc_w0	rc_w0		rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0			

Bit	notation	clarification
15:13	Reserved	Reserved, always reads 0.
12	CC4OF	CC4OF: Capture/Compare4 overcapture flag See CC1OF for description.
11	CC3OF	CC3OF: Capture/Compare3 overcapture flag See CC1OF for description.
10	CC2OF	CC2OF: Capture/Compare2 overcapture flag See CC1OF for description.
9	CC1OF	CC1OF: Capture/Compare1 overcapture flag This flag can be set '1' by hardware only if the corresponding channel is configured for input capture. Writing '0' clears this bit. 0: No duplicate captures are generated; 1: The state of CC1IF is already '1' when the counter value is captured in the TIMx_CCR1 register.
8:7	Reserved	Reserved, always reads 0.
6	TIF	TIF: Trigger interrupt flag (Triggerinterruptflag) It is '1' by hardware to this position 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 software. 0: No trigger event is generated; 1: Trigger interrupt waiting for response.
5	Reserved	Reserved, always reads 0.
4	CC4IF	CC4IF: Capture/Compare4 interrupt flag Refer to the CC1IF description.
3	CC3IF	CC3IF: Capture/Compare3 interrupt flag Refer to the CC1IF description.
2	CC2IF	CC2IF: Capture/Compare2 interrupt flag Refer to the CC1IF description.
1	CC1IF	CC1IF: Capture/Compare1 interrupt flag If channel CC1 is configured for output mode: This bit is set '1' by hardware when the counter value matches the comparison value, except in centrosymmetric mode (refer to the CMS bit in the TIMx_CR1 register). It is cleared '0' by software. 0: No match occurred;

		<p>1: The value of TIMx_CNT matches the value of TIMx_CCR1. If channel CC1 is configured for input mode: This bit is set '1' by hardware when a capture event occurs, and it is cleared '0' by software or by reading TIMx_CCR1. 0: No input capture is generated; 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	<p><b>UIF:</b> Update interrupt flag (Update interrupt flag) This bit is set '1' by hardware when an update event is generated. It is cleared '0' by software. 0: No update events are generated; 1: Update interrupt waiting for response. This bit is set '1' by hardware when the register is updated: -If UDIS=0 and URS=0 in the TIMx_CR1 register, an update event is generated when UG=1 in the TIMx_EGR register (software re-initializes the counter CNT); -If UDIS=0 and URS=0 in the TIMx_CR1 register, an update event is generated when the counter CNT is reinitialized by the trigger event. (Refer to the description of the synchronization control register)</p>

#### 14.4.6 TIM2 to TIM5 Event Generation Register (TIMx\_EGR)

Offset address:0x14

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									TG	Reserved	CC4G	CC3G	CC2G	CC1G	UG
									W		W	W	W	W	W

Bit	notation	clarification
15:7	Reserved	Reserved, always reads 0.
6	TG	<p><b>TG:</b> Trigger generation This bit is set '1' by software to generate a trigger event that is automatically cleared '0' by hardware. 0: No action; 1: TIF=1 in the TIMx_SR register generates the corresponding interrupt and DMA if they are turned on.</p>
5	Reserved	Reserved, always reads 0.
4	CC4G	<b>CC4G:</b> Capture/compare4 generation Refer to the CC1G description.
3	CC3G	<b>CC3G:</b> Capture/compare3 generation Refer to the CC1G description.
2	CC2G	<b>CC2G:</b> Capture/compare2 generation Refer to the CC1G description.
1	CC1G	<p><b>CC1G:</b> Generation of Capture/compare1 generation events This bit is set '1' by software to generate a capture/compare event and is automatically cleared '0' by hardware. 0: No action; 1: Generate a capture/compare event on channel CC1: If channel CC1 is configured as an output: Set CC1IF=1 to generate the corresponding interrupt and DMA if it is turned on. If channel CC1 is configured as an input: The current counter value is captured to the TIMx_CCR1 register; set CC1IF=1 to generate the corresponding interrupts and DMAs if they are turned on. if CC1IF is already 1, set CC1OF=1.</p>
0	UG	<p><b>UG:</b> Update generation This bit is set '1' by software and cleared '0' automatically by hardware. 0: No action; 1: Reinitialize the counter and generate an update event. Note that the prescaler counter is also cleared '0' (but the prescaler coefficients remain unchanged). The counter is cleared '0' if in centrosymmetric mode or DIR=0 (counting up), if DIR=1 (counting down) the counter takes the value of TIMx_ARR.</p>

## 14.4.7 TIM2 to TIM5 Capture/Compare Mode Register 1 (TIMx\_CMR1)

Offset address: 0x18

Reset value: 0x0000

The channel can be used as input (capture mode) or output (compare mode), and the direction of the channel is defined by the corresponding CCxS. The other bits of this register function differently in input and output modes. OCxS describes the function of the channel in output mode and ICxS describes the function of the channel in output mode. It is therefore important to note that the same bit functions differently in output mode and input mode.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2:0]			OC2PE	OC2FE	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	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW

### Output comparison mode:

Bit	notation	clarification
15	OC2CE	OC2CE: Output compare2 clear enable
14:12	OC2M[2:0]	OC2M[2:0]: Output compare2 mode
11	OC2PE	OC2PE: Output compare2 preload enable
10	OC2FE	OC2FE: Output compare2 fast enable
9:8	CC2S[1:0]	CC2S[1:0]: Capture/Compare2 selection This bit defines the direction of the channel (input/output), and the selection of the input pin: 00: The CC2 channel is configured as an output; 01: CC2 channel is configured as an input and IC2 is mapped on TI2; 10: The CC2 channel is configured as an input and IC2 is mapped on TI1; 11: The CC2 channel is configured as an input with IC2 mapped on TRC. This mode operates only when the internal trigger input is selected (by the TS bit of the TIMx_SMCR register). Note: CC2S is writable only when the channel is closed (CC2E='0' in the TIMx_CCER register).
7	OC1CE	OC1CE: Output compare1 clear enable 0: OC1REF is not affected by the ETRF input; 1: Clear OC1REF=0 as soon as the ETRF input is detected high.
6:4	OC1M[2:0]	OC1M[2:0]: Output compare1 mode (Output compare1 enable) These 3 bits define the action of the output reference signal OC1REF, which determines the value of OC1. OC1REF is active high, and the active level of OC1 depends on the CC1P bit. 000: Freeze. Comparison between output comparison register TIMx_CCR1 and counter TIMx_CNT does not work for OC1REF; 001: Set channel 1 to active level when matched. Forces OC1REF high when the value of counter TIMx_CNT is the same as capture/compare register 1 (TIMx_CCR1). 010: Set channel 1 to an invalid level when matched. Forces OC1REF low when the value of counter TIMx_CNT is the same as capture/compare register 1 (TIMx_CCR1). 011: Flip. Flips the level of OC1REF when TIMx_CCR1 = TIMx_CNT. 100: Forces an invalid level. Forces OC1REF low. 101: Force to active level. Forces OC1REF high. 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 active ( OC1REF=1). 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. 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. 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.
3	OC1PE	OC1PE: Output compare1 preload enable 0: Disables the preload function of the TIMx_CCR1 register, which can be written to the TIMx_CCR1 register at any time and the newly written value takes effect immediately. 1: Enable the preload function of TIMx_CCR1 register, read/write operation only operates on the preloaded register, the preloaded value of TIMx_CCR1 is transferred to the current register when the update event arrives. 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. Note 2: Only in single pulse mode (OPM='1' in TIMx_CR1 register), PWM mode can be

		used without checking the preload register, otherwise its action is uncertain.
2	OC1FE	<b>OC1FE:</b> Output compare1 fast enable This bit is used to speed up the response of the CC output to trigger input events. 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 a valid edge occurs at the input of the flip-flop. 1: The active edge of the input to the flip-flop acts as if a comparison match has occurred. Therefore, OC is set to the comparison level independent of the comparison result. The delay between the active edge of the sampling flip-flop and the CC1 output is reduced to 3 clock cycles. This bit only functions when the channel is configured for PWM1 or PWM2 mode.
1:0	CC1S[1:0]	<b>CC1S[1:0]:</b> Capture/Compare1 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: The CC1 channel is configured as an input with IC1 mapped on TRC. This mode operates only when the internal trigger input is 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).

### Input Capture Mode:

Bit	notation	clarification
15:12	IC2F[3:0]	<b>IC2F[3:0]:</b> Input capture2 filter
11:10	IC2PSC[1:0]	<b>IC2PSC[1:0]:</b> input/capture2 prescaler (input capture2 prescaler)
9:8	CC2S[1:0]	<b>CC2S[1:0]:</b> Capture/compare2 selection These 2 bits define the direction of the channel (input/output), and the selection of the input pin: 00: The CC2 channel is configured as an output; 01: CC2 channel is configured as an input and IC2 is mapped on TI2; 10: The CC2 channel is configured as an input and IC2 is mapped on TI1; 11: The CC2 channel is configured as an input with IC2 mapped on TRC. This mode operates only when the internal trigger input is selected (by the TS bit of the TIMx_SMCR register). Note: CC2S is writable only when the channel is closed (CC2E=0' in the TIMx_CCER register).
7:4	IC1F[3:0]	<b>IC1F[3:0]:</b> Input capture1 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 records N events and then generates a jump in the output: 0000: no filter, sampling at fDTS 1000: sampling frequency fSAMPLING = fDTS/8, N = 6 0001: Sampling frequency fSAMPLING=fCK_INT, N=21001: Sampling frequency fSAMPLING=fDTS/8, N=8 0010: Sampling frequency fSAMPLING=fCK_INT, N=41010: Sampling frequency fSAMPLING=fDTS/16, N=5 0011: Sampling frequency fSAMPLING=fCK_INT, N=81011: Sampling frequency fSAMPLING=fDTS/16, N=6 0100: Sampling frequency fSAMPLING=fDTS/2, N=61100: Sampling frequency fSAMPLING=fDTS/16, N=8 0101: Sampling frequency fSAMPLING=fDTS/2, N=81101: Sampling frequency fSAMPLING=fDTS/32, N=5 0110: Sampling frequency fSAMPLING=fDTS/4, N=61110: Sampling frequency fSAMPLING=fDTS/32, N=6 0111: Sampling frequency fSAMPLING=fDTS/4, N=81111: Sampling frequency fSAMPLING=fDTS/32, N=8 Note: When ICx[3:0] = 1, 2 or 3, fDTS in the formula is replaced by CK_INT.
3:2	IC1PSC[1:0]	<b>IC1PSC[1:0]:</b> input/capture1 prescaler (Input capture1 prescaler) These 2 bits define the prescaler coefficient for the CC1 input (IC1). Once CC1E = 0' (in the TIMx_CCER register), the prescaler is reset. 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]	<b>CC1S[1:0]:</b> Capture/Compare1 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: The CC1 channel is configured as an input with IC1 mapped on TRC. This mode operates only when the internal trigger input is 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).

## 14.4.8 TIM2 to TIM5 Capture/Compare Mode Register 2 (TIMx\_CMCR2)

Offset address: 0x1C

Reset value: 0x0000

Refer to the description of the CCMR1 register above

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4CE	OC4M [2:0]			OC4PE	OC4FE	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	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	notation	clarification
15	OC4CE	OC4CE: Output compare4 clear enable
14:12	OC4M [2:0]	OC4M[2:0]: Output compare4 mode
11	OC4PE	OC4PE: Output compare4 preload enable
10	OC4FE	OC4FE: Output compare4 fast enable
9:8	CC4S[1:0]	CC4S[1:0]: Capture/Compare4 selection These 2 bits define the direction of the channel (input/output), and the selection of the input pin: 00: The CC4 channel is configured as an output; 01: CC4 channel is configured as an input and IC4 is mapped on TI4; 10: CC4 channel is configured as an input and IC4 is mapped on TI3; 11: The CC4 channel is configured as an input with IC4 mapped on TRC. This mode operates only when the internal trigger input is selected (by the TS bit of the TIMx_SMCR register). Note: CC4S is writable only when the channel is closed (CC4E='0' in the TIMx_CCER register).
7	OC3CE	OC3CE: Output compare3 clear enable
6:4	OC3M [2:0]	OC3M[2:0]: Output compare3 mode
3	OC3PE	OC3PE: Output compare3 preload enable
2	OC3FE	OC3FE: Output compare3 fast enable
1:0	CC3S[1:0]	CC3S[1:0]: Capture/Compare3 selection These 2 bits define the direction of the channel (input/output), and the selection of the input pin: 00: The CC3 channel is configured as an output; 01: CC3 channel is configured as an input and IC3 is mapped on TI3; 10: CC3 channel is configured as an input and IC3 is mapped on TI4; 11: The CC3 channel is configured as an input with IC3 mapped on TRGI. This mode operates only when the internal trigger input is selected (by the TS bit of the TIMx_SMCR register). Note: CC3S is writable only when the channel is closed (CC3E='0' in the TIMx_CCER register).

### Input Capture Mode:

Bit	notation	clarification
15:12	IC4F[3:0]	IC4F[3:0]: Input capture4 filter
11:10	IC4PSC[1:0]	IC4PSC[1:0]: input/capture4 prescaler (input capture4 prescaler)
9:8	CC4S[1:0]	CC4S[1:0]: Capture/compare4selection These 2 bits define the direction of the channel (input/output), and the selection of the input pin: 00: The CC4 channel is configured as an output; 01: CC4 channel is configured as an input and IC4 is mapped on TI4; 10: The CC4 channel is configured as an input and IC4 is mapped on TI3; 11: The CC4 channel is configured as an input with IC4 mapped on TRC. This mode operates only when the internal trigger input is selected (by the TS bit of the TIMx_SMCR register). Note: CC4S is writable only when the channel is closed (CC4E='0' in the TIMx_CCER register).
7:4	IC3F[3:0]	IC3F[3:0]: Input capture3 filter
3:2	IC3PSC[1:0]	IC3PSC[1:0]: input/capture3 prescaler (Input capture3 prescaler)
1:0	CC3S[1:0]	CC3S[1:0]: Capture/Compare3 selection These 2 bits define the direction of the channel (input/output), and the selection of

		<p>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: The CC3 channel is configured as an input with IC3 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: CC3S is writable only when the channel is closed (CC3E=0' in the TIMx_CCER register).</p>
--	--	---

## 14.4.9 TIM2 to TIM5 Capture/Compare Enable Register (TIMx\_CCER)

Offset address: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	CC4P	CC4E	Reserved	CC3P	CC3E	Reserved	CC2P	CC2E	Reserved	CC1P	CC1E				
	rw	rw		rw	rw		rw	rw		rw	rw			rw	rw
Bit	notation	show													
15:14	Reserved	Reserved, always reads 0.													
13	CC4P	CC4P: Input/Capture4 output polarity (Capture/Compare4 output polarity) Refer to the description of CC1P.													
12	CC4E	CC4E: Input/Capture4 output enable (Capture/Compare4 output enable) Refer to the description of CC1E.													
11:10	Reserved	Reserved, always reads 0.													
9	CC3P	CC3P: Capture/Compare3 output polarity Refer to the description of CC1P.													
8	CC3E	CC3E: input/capture3 output enable (Capture/Compare3 output enable) Refer to the description of CC1E.													
7:6	Reserved	Reserved, always reads 0.													
5	CC2P	CC2P: Input/Capture2 output polarity (Capture/Compare2 output polarity) Refer to the description of CC1P.													
4	CC2E	CC2E: Capture/Compare2 output enable Refer to the description of CC1E.													
3:2	Reserved	Reserved, always reads 0.													
1	CC1P	<p>CC1P: Input/Capture1 output polarity (Capture/Compare1 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: This bit selects whether IC1 or the inverted signal of IC1 is used as the trigger or capture signal. 0: not inverted: capture occurs on the rising edge of IC1; when used as an external trigger, IC1 is not inverted. 1: Inverted: Capture occurs on the falling edge of IC1; when used as an external trigger, IC1 is inverted.</p>													
0	CC1E	<p>CC1E: Input/Capture1 output enable (Capture/Compare1 output enable) The CC1 channel is configured as an output: 0: Off - OC1 disables output. 1: Turn on -OC1 signal output to the corresponding output pin. The CC1 channel is configured as an input: This bit determines whether the counter value can be captured into the TIMx_CCR1 register. 0: Capture disabled; 0: Capture enable.</p>													

Table77 Output control bits for standard OCx channels

CCxE bit	OCx Output Status
0	Disable output (OCx=0, OCx_EN=0)
1	OCx = OCxREF + polarity, OCx_EN = 1

Notes: The status of the external I/O pins connected to the standard OCx channel depends on the OCx channel status and the GPIO and AFIO registers.



#### 14.4.10 TIM2 to TIM5 Counter (TIMx\_CNT)

Offset address: 0x24

Reset value: 0x0000

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

#### 14.4.11 TIM2 to TIM5 Prescaler (TIMx\_PSC)

Offset address: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC [15:0]															
rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw															
Bit	notation	clarification													
15:0	PSC [15:0]	<b>PSC[15:0]: Prescaler value (Prescaler value)</b> The clock frequency of the counter, CK_CNT, is equal to fCK_PSC/(PSC[15:0]+1). The PSC contains the value loaded into the current prescaler register when the update event is generated.													

#### 14.4.12 TIM2 to TIM5 Auto-Reload Register (TIMx\_ARR)

Offset address: 0x2C

Reset value: 0x0000

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	notation	clarification													
15:0	ARR[15:0]	<b>ARR[15:0]:Auto reload value</b> The ARR contains the values that will be transferred to the actual Auto-Reload Register. Refer to the 14.3.1 section for details: updates and actions related to the ARR. The counter does not work when the value of Auto-Reload is empty.													

#### 14.4.13 TIM2 to TIM5 Capture/Compare Register 1 (TIMx\_CCR1)

Offset address: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw/r0 rw/r0 rw/r0 rw/r0 rw/r0 rw/r0 rw/r0 rw/r0 rw/r0 rw/r0 rw/r0 rw/r0 rw/r0 rw/r0 rw/r0 rw/r0															
Bit	notation	clarification													
15:0	CCR1[15:0]	<b>CCR1[15:0]:Capture/Compare1 value (Capture/Compare1 value)</b> 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), 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. The current capture/comparison register participates in the comparison with counter TIMx_CNT and generates an output signal on port OC1. If the CC1 channel is configured as an input: CCR1 contains the counter value transmitted by the last input capture 1 event (IC1).													

#### 14.4.14 TIM2 to TIM5 Capture/Compare Register 2 (TIMx\_CCR2)

Offset address: 0x38

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw/r0	rw/r0	rw/r0	rw/r0	rw/r0	rw/r0	rw/r0	rw/r0	rw/r0	rw/r0	rw/r0	rw/r0	rw/r0	rw/r0	rw/r0	rw/r0
Bit	notation	clarification													
15:0	CCR2[15:0]	<b>CCR2[15:0]:Capture/Compare2 value</b> If the CC2 channel is configured as an output: CCR2 contains the value loaded into the current Capture/Compare 2 register (preloaded value). If the preload feature is not selected in the TIMx_CMR2 register (OC2PE bit), the written value is immediately transferred to the current register. Otherwise this preloaded value is transferred to the current Capture/Compare2 register only when an update event occurs. The current capture/comparison register participates in the comparison with counter TIMx_CNT and generates an output signal on port OC2. If the CC2 channel is configured as an input: CCR2 contains the counter value transmitted by the last input capture 2 event (IC2).													

#### 14.4.15 TIM2 to TIM5 Capture/Compare Register 3 (TIMx\_CCR3)

Offset address: 0x3C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit	notation	clarification													
15:0	CCR3[15:0]	<b>CCR3[15:0]:Capture/Compare3 value</b> If the CC3 channel is configured as an output: CCR3 contains the value loaded into the current Capture/Compare 3 register (preloaded value). If the preload feature is not selected in the TIMx_CMR3 register (OC3PE bit), the written value is immediately transferred to the current register. Otherwise this preloaded value is transferred to the current Capture/Compare 3 register only when an update event occurs. The current capture/comparison register participates in the comparison with counter TIMx_CNT and generates an output signal on port OC3. If the CC3 channel is configured as an input: CCR3 contains the counter value transmitted by the last input capture 3 event (IC3).													

#### 14.4.16 TIM2 to TIM5 Capture/Compare Register 4 (TIMx\_CCR4)

Offset address: 0x40

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw/r0	rw/r0	rw/r0	rw/r0	rw/r0	rw/r0	rw/r0	rw/r0	rw/r0	rw/r0	rw/r0	rw/r0	rw/r0	rw/r0	rw/r0	rw/r0
Bit	notation	clarification													
15:0	CCR4[15:0]	<b>CCR4[15:0]:Capture/Compare4 value</b> If the CC4 channel is configured as an output: CCR4 contains the value loaded into the current Capture/Compare 4 register (preloaded value). If the preload feature is not selected in the TIMx_CMR4 register (OC4PE bit), the written value is immediately transferred to the current register. Otherwise this preloaded value is transferred to the current capture/compare4 register only when an update event occurs. The current capture/comparison register participates in the comparison with counter TIMx_CNT and generates an output signal on port OC4. If the CC4 channel is configured as an input: CCR4 contains the counter value transmitted by the last input capture 4 event (IC4).													

## 14.4.17 TIM2 to TIM5 DMA Control Register (TIMx\_DCR)

Offset address: 0x48

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				DBL [4:0]				Reserved				DBA [4:0]			
rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW
Bit	notation	clarification													
15:13	Reserved	Reserved, always reads 0.													
12:8	DBL [4:0]	<b>DBL[4:0]:DMA burst length</b> These bits define the length of the DMA transfer in continuous mode (when a read or write is performed to the TIMx_DMAR register, the timer then performs one continuous transfer), i.e.: defines the number of bytes transferred: 00000: 1 byte 00001: 2 bytes 00010: 3 bytes ..... 10001: 18 bytes													
7:5	Reserved	Reserved, always reads 0.													
4:0	DBA [4:0]	<b>DBA[4:0]:DMA base address (DMA base address)</b> These bits define the base address of the DMA in continuous mode (when reading or writing to the TIMx_DMAR register), and the 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. .....													

## 14.4.18 TIM2 to TIM5 Continuous Mode DMA Address (TIMx\_DMAR)

Offset address: 0x4C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAB [15:0]															
rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW
Bit	notation	clarification													
15:0	DMAB [15:0]	<b>DMAB[15:0]:DMA register for burst accesses</b> A read or write to the TIMx_DMAR register results in an access operation to the register where the following address is located: the TIMx_CR1 address + DBA + DMA index, where: "TIMx_CR1 Address" is the address where control register 1 (TIMx_CR1) is located; "DBA" is the base address defined in the TIMx_DCR register; The "DMA Index" is an offset that is automatically controlled by the DMA and depends on the DBL defined in the TIMx_DCR register.													

### Example of how to use the DMA burst feature

In this example, the timer DMA burst feature is used to update the contents of the CCRx register (x = 2, 3, 4), the DMA

Transfers half a word into the CCRx register.

This is accomplished through the following steps.

1. Configure the corresponding DMA channels as follows.
  - The DMA channel peripheral address is the DMAR register address
  - The DMA channel memory address is the address of the buffer in RAM that contains the data to be transferred by DMA to the CCRx register.
  - Number of data to be transferred = 3 (see note below)
  - -Close round mode.
2. Configure the DCR registers by configuring the DBA and DBL bit fields as follows:DBL=3 transmissions, DBA=0xE.
3. enable TIMx update DMA request (set UDE bit in DIER register).
4. Enable TIMX
5. Enable DMA channel

**Notes:** This example applies to the case where each CCRx register needs to be updated once. If each CCRx register needs to be updated twice, the number of data to be transferred should be 6. Take the example of a buffer in RAM containing data 1, data 2, data 3, data 4, data 5 and data 6. Transfer data to the CCRx registers as follows:At the first update DMA request, data 1 is

transferred to CCR2, data 2 is transferred to CCR3, and data 3 is transferred to CCR4; at the second update DMA request, data 4 is transferred to CCR2, data 5 is transferred to CCR3, and data 6 is transferred to CCR4.

## 14.4.19 TIM2 to TIM5 Registers

The following table maps all registers of the TIMx into a 16-bit addressable (addressed) space.

Table78 TIM2 to TIM5 Registers and Reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
000h	TIMx_CR1	Reserved																						CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN	
	Reset value																							0	0	0	0	0	0	0	0			
004h	TIMx_CR2	Reserved																									TI1S	MMS [2:0]		CCDS	Reserved			
	Reset value																										0	0	0	0	0			
008h	TIMx_SMCR	Reserved																	ETP	ECE	ETPS [1:0]		EFT [3:0]			MSM	TS [2:0]		Reserved	SMS [2:0]				
	Reset value																		0	0	0	0	0	0	0	0	0	0		0	0			
00Ch	TIMx_DIER	Reserved																	TDE	Reserved	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Reserved	T1E	Reserved	CC4IE	CC3IE	CC2IE	CC1IE	UIE	
	Reset value																		0		0	0	0	0	0		0		0	0	0	0	0	
010h	TIMx_SR	Reserved																			CC4OF	CC3OF	CC2OF	CC1OF	Reserved	TI1	Reserved	CC4IF	CC3IF	CC2IF	CC1IF	UIF		
	Reset value																				0	0	0	0		0		0	0	0	0	0	0	0
014h	TIMx_EGR	Reserved																									Reserved	TG	Reserved	CC4G	CC3G	CC2G	CC1G	UG
	Reset value																											0		0	0	0	0	0
018h	TIMx_CMR1 Output Compare Mode	Reserved																	OC2CE	OC2M [2:0]		OC2PE	OC2FE	CC2S [1:0]		OC1CE	OC1M [2:0]		OC1PE	OC1FE	CC1S [1:0]			
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0			
	TIMx_CMR1 Input Capture Mode	Reserved																	IC2F [3:0]		IC2PSC [1:0]		CC2S [1:0]		IC1F [3:0]		IC1PSC [1:0]		CC1S [1:0]					
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0			
01Ch	TIMx_CMR2 Output Compare Mode	Reserved																	OC4CE	OC4M [2:0]		OC4PE	OC4FE	CC4S [1:0]		OC3CE	OC3M [2:0]		OC3PE	OC3FE	CC3S [1:0]			
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0			
	TIMx_CMR2 Input Capture Mode	Reserved																	IC4F [3:0]		IC4PSC [1:0]		CC4S [1:0]		IC3F [3:0]		IC3PSC [1:0]		CC3S [1:0]					
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0			
020h	TIMx_CCER	Reserved																			CC4P	CC4E	Reserved	CC3P	CC3E	Reserved	CC2P	CC2E	Reserved	CC1P	CC1E			
	Reset value																				0	0		0	0		0	0		0	0	0	0	
024h	TIMx_CNT	Reserved																	CNT [15:0]															
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
028h	TIMx_PSC	Reserved																	PSC [15:0]															
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0					
02Ch	TIMx_ARR	Reserved																ARR[15:0]																				
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
030h	Reserved																																					
034h	TIMx_CCR1	Reserved																CCR1[15:0]																				
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
038h	TIMx_CCR2	Reserved																CCR2[15:0]																				
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
03Ch	TIMx_CCR3	Reserved																CCR3[15:0]																				
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
040h	TIMx_CCR4	Reserved																CCR4[15:0]																				
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
044h	Reserved																																					
048h	TIMx_DCR	Reserved																		DBL [4:0]				Reserved		DBA [4:0]												
	Reset value																			0	0	0	0	0					0	0	0	0	0					
04Ch	TIMx_DMAR	Reserved																DMAB [15:0]																				
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				

SeeTable1 for register start addresses.

---

## 15 General Purpose Timers (TIM9 to TIM14)

### 15.1 TIM9 to TIM14 Introduction

The general-purpose timer is a 16-bit auto-load counter constructed by driving it through 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.

Each timer is completely independent and does not share any resources with each other. They can operate synchronously together, see section 15.3.12.

### 15.2 Main Functions of TIM9 to TIM14

#### 15.2.1 Main Functions of TIM9/TIM12

General purpose TIM9 through TIM14 timer functions include:

- 16-bit up, down, up/down auto-load counter
- 16-bit programmable (can be modified in real time) prescaler, counter clock frequency division factor is any value between 1 ~ 65536
- 4 independent channels:
  - Input Capture
  - Output Comparison
  - PWM generation (edge or center alignment mode)
  - Single pulse mode output
- Synchronization circuits using external signals to control timers and timer interconnections
- An interrupt is generated when the following event occurs:
  - Updates: Counter overflow up/down, counter initialization (via software or internal/external trigger)
  - Trigger event (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

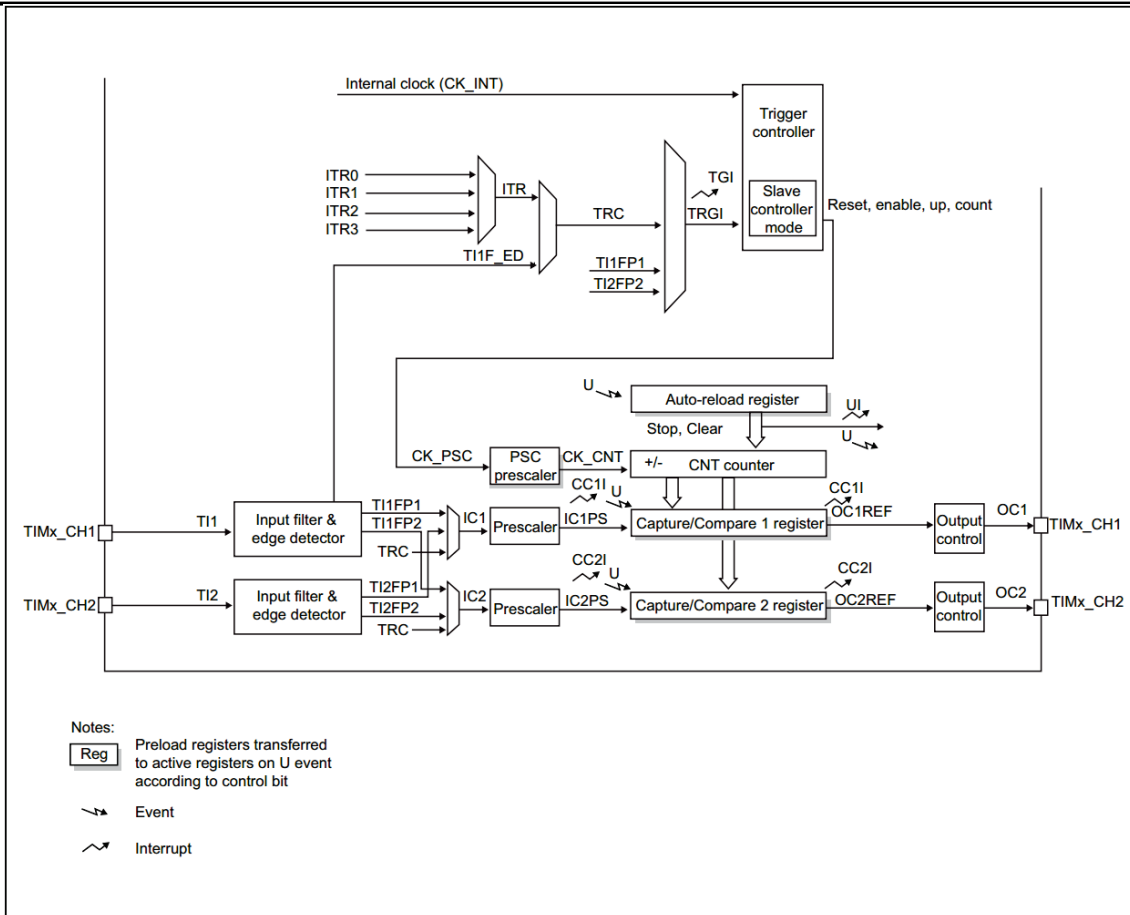


Figure 142 General Purpose Timer Block Diagram (TIM9/TIM12)

## 15.2.2 Main Functions of TIM10/TIM11 and TIM13/TIM14

General purpose TIM10/TIM11 and TIM13/TIM14 timer functions are included:

- 16-bit up, down, up/down auto-load counter
- 16-bit programmable (can be modified in real time) prescaler, counter clock frequency division factor is any value between 1 ~ 65536
- 4 independent channels:
  - Input Capture
  - Output Comparison
  - PWM generation (edge or center alignment mode)
  - Single pulse mode output
- An interrupt is generated when the following event occurs:
  - Updates: Counter overflow up/down, counter initialization (via software or internal/external trigger)
  - Input Capture
  - Output Comparison

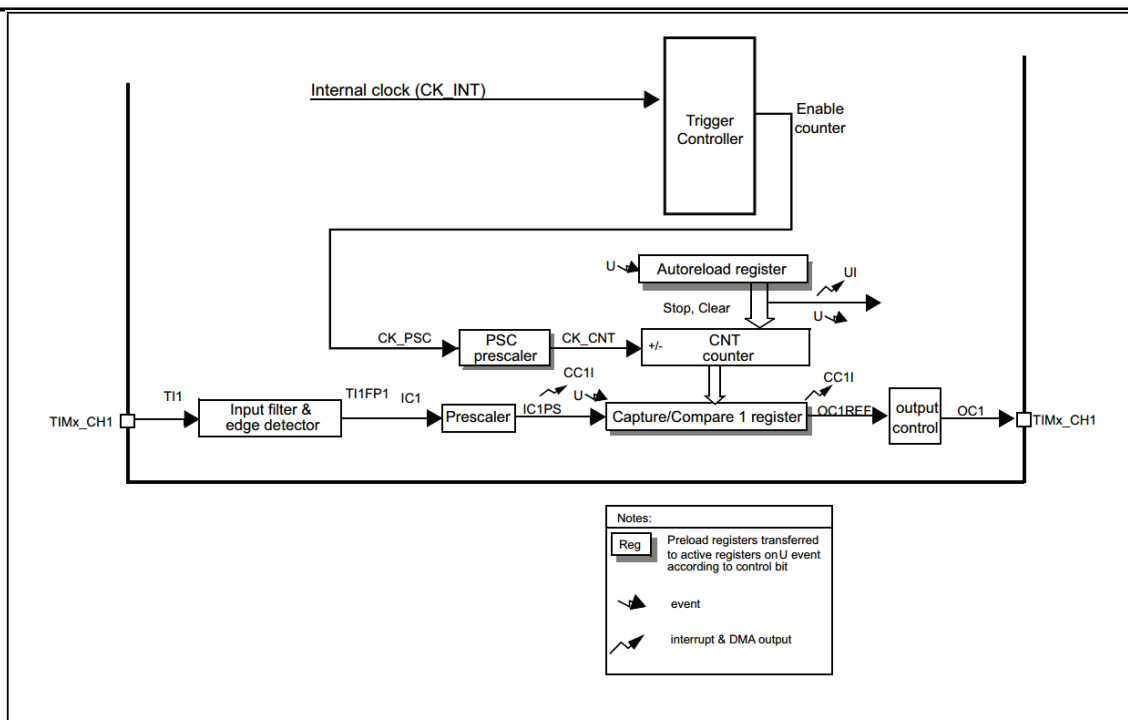


Figure143 General purpose timer block diagram (TIM10/11/12/13/14)

## 15.3 TIM9 to TIM14 Functional Description

### 15.3.1 Time Base Unit (in computing)

The main part of the programmable general purpose timer is a 16-bit counter and its associated auto-load register.

This counter clock is divided by a prescaler.

The counter, auto-load registers, and prescaler registers can be read and written by software and remain read and written while the counter is running.

The time base unit contains:

- Counter Register (TIMx\_CNT)
- Prescaler register (TIMx\_PSC)
- Autoload Register (TIMx\_ARR)

The autoloader registers are preloaded, and writing or reading the auto-reload registers will access the preload 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 immediately or at each update event UEV. An update event is generated when the counter reaches an overflow condition (underflow condition when counting down) and when the UDIS bit in the TIMx\_CR1 register is equal to '0'. The update event can also be generated by software. The generation of update events for each configuration is described in detail later. The counter is driven by the prescaler's clock output, CK\_CNT, which is valid only when the counter bit (CEN) in the counter's TIMx\_CR1 register is set. (See the Slave Mode description of the controller for details on counter enable).

*Notes: The true counter enable signal, CNT\_EN, is set after one clock cycle of CEN.*

#### Prescaler Description

The prescaler divides the counter clock frequency by any value between 1 and 65536. It is based on a 16-bit counter controlled by a 16-bit register (in the TIMx\_PSC register). This control register is buffered and can be during operation. The new prescaler parameter is used when the next update event arrives.

Figure144 and Figure145 give examples of changing counter parameters while the prescaler is running.



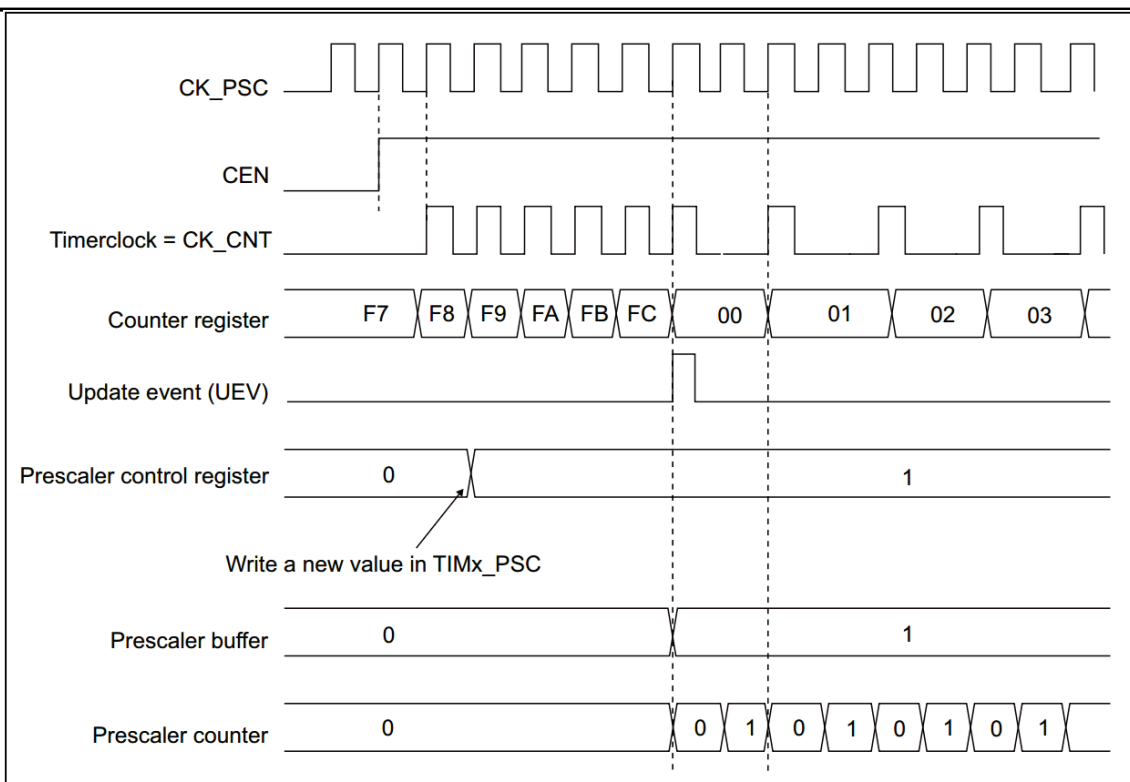


Figure144 Timing diagram of the counter when the parameter of the prescaler is changed from 1 to 2

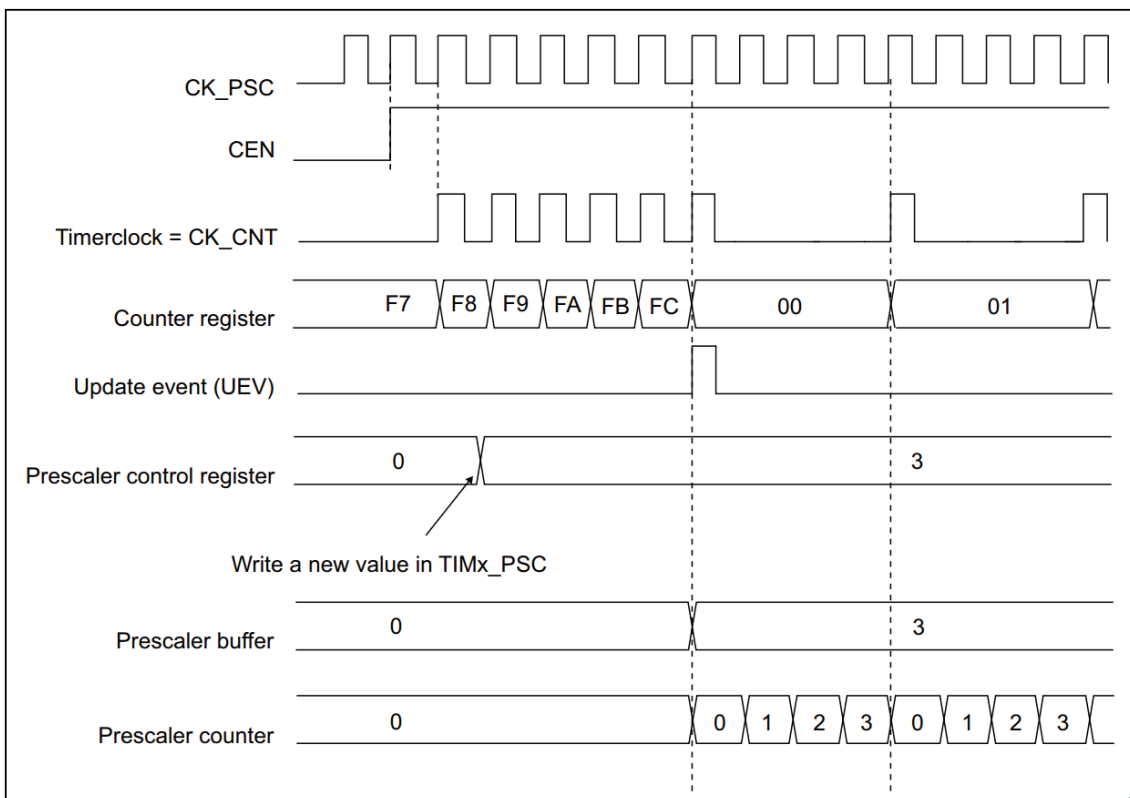


Figure145 Timing diagram of the counter when the parameter of the prescaler is changed from 1 to 4

## 15.3.2 Counter Mode

### Up Count Mode

In count-up mode, the counter counts from 0 to the auto-load value (the contents of the TIMx\_ARR counter), then starts counting from 0 again and generates a counter overflow event. An update event can be generated each time the counter overflows, and an update event can also be generated by setting the UG bit in the TIMx\_EGR register (either in software or using the slave mode controller).

Setting the UDIS bit in the TIMx\_CR1 register disables the update event; this prevents the shadow register from being updated when a new value is written to the preload register. No update event is generated until the UDIS bit is cleared '0'. However, when an update should be generated, the counter will still be cleared to '0' and the prescaler count will be reset to 0 (but the prescaler factor will remain unchanged). In addition, if the URS bit in the TIMx\_CR1 register is set (selecting update request), setting the UG bit will generate an update event UEV, but the hardware will not set the UIF flag (i.e., no interrupt request will be ); this is to avoid simultaneous update and capture interrupts when the counter is cleared 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 prescaler buffer is set to the value of the preload register (the contents of the TIMx\_PSC register).

- The autoloader shadow register is reset to the value of the preload register (TIMx\_ARR).

The following figure gives some examples of how the counter acts at different clock frequencies when TIMx\_ARR=0x36.

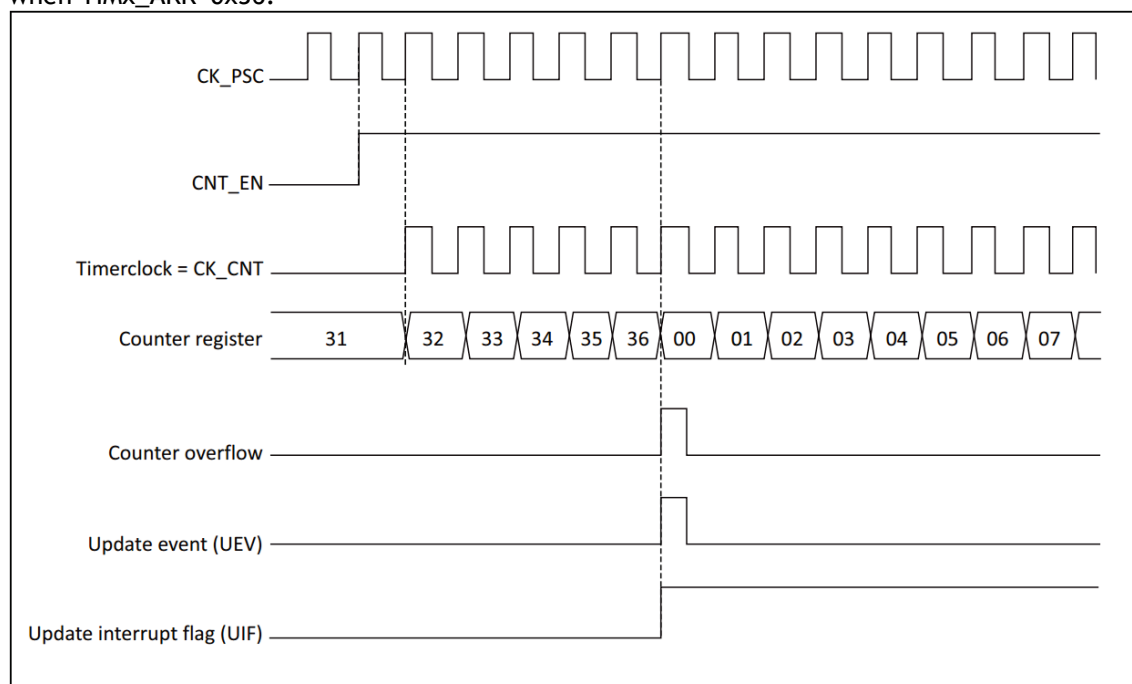


Figure146 Timing diagram of the counter with internal clock division factor of 1

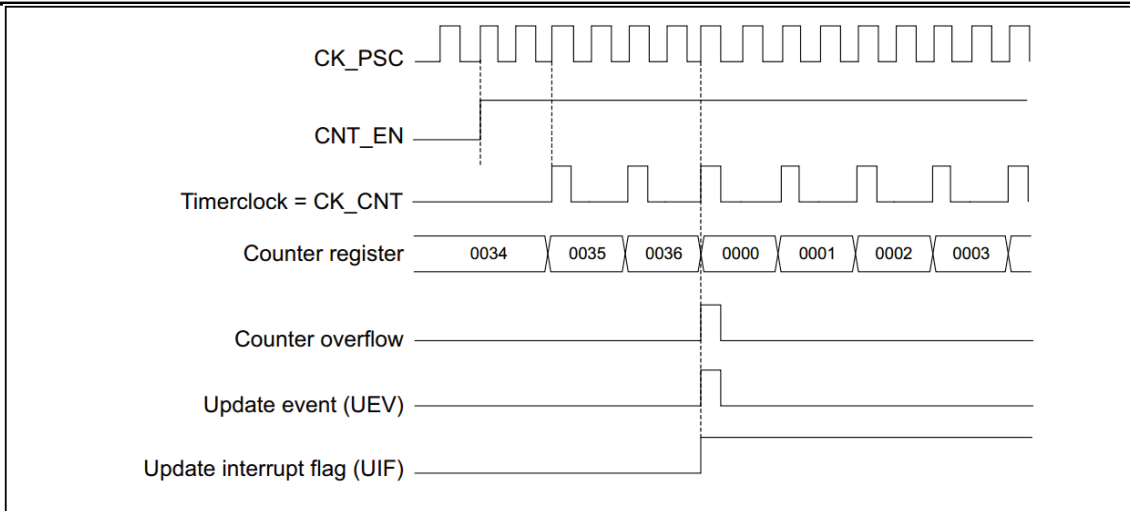


Figure147 Timing diagram of the counter with internal clock division factor of 2

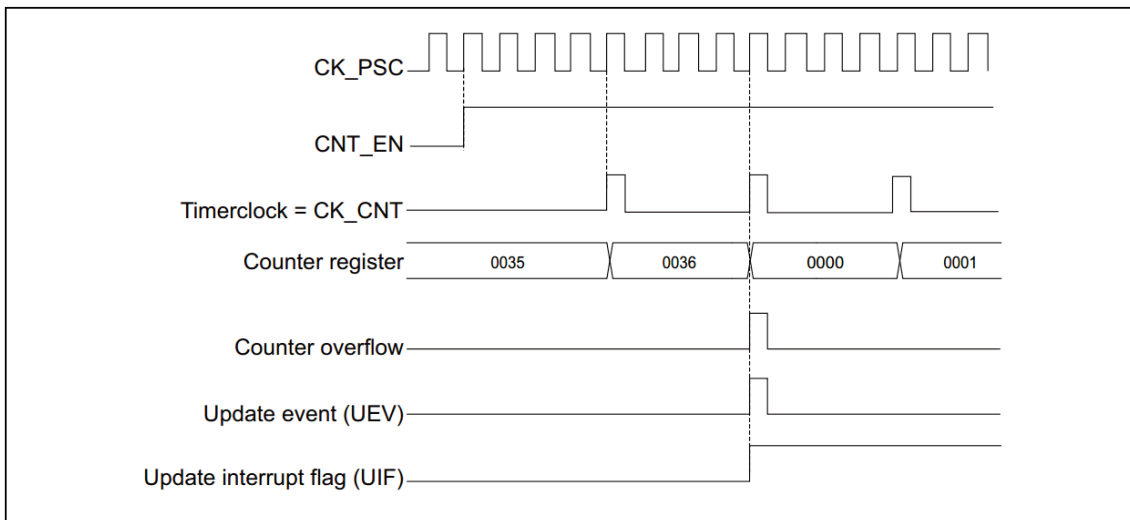


Figure148 Timing diagram of the counter with internal clock division factor of 4

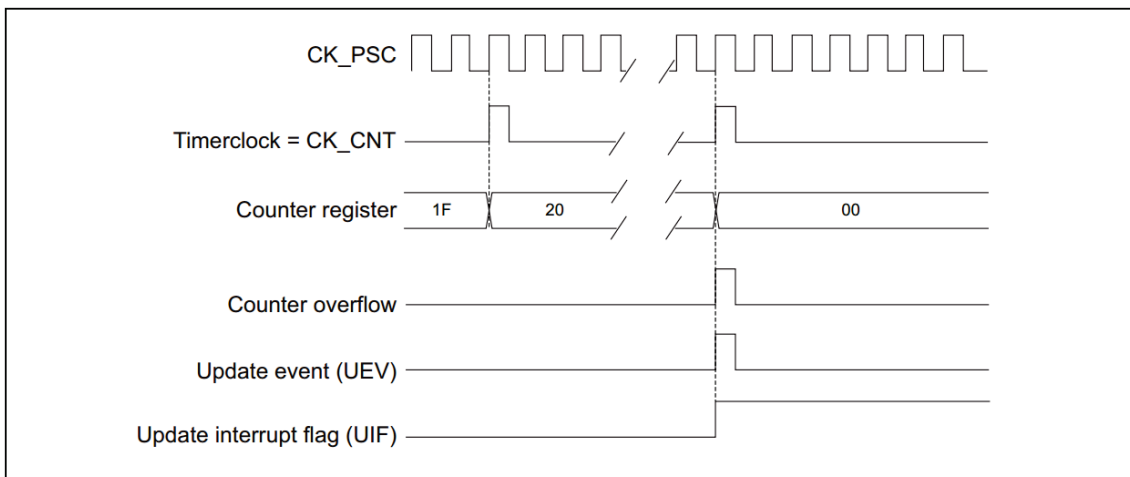


Figure149 Timing diagram of the counter with internal clock division factor N

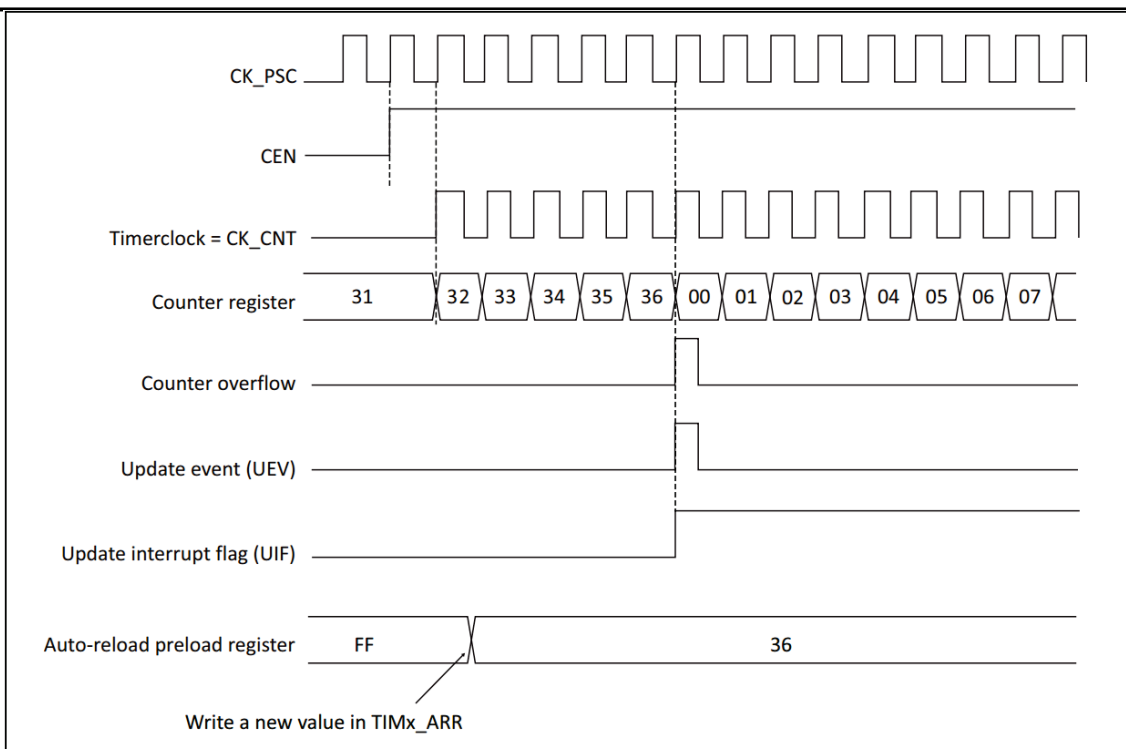


Figure150 Counter timing diagram, update event when ARPE=0 (TIMx\_ARR is not preloaded)

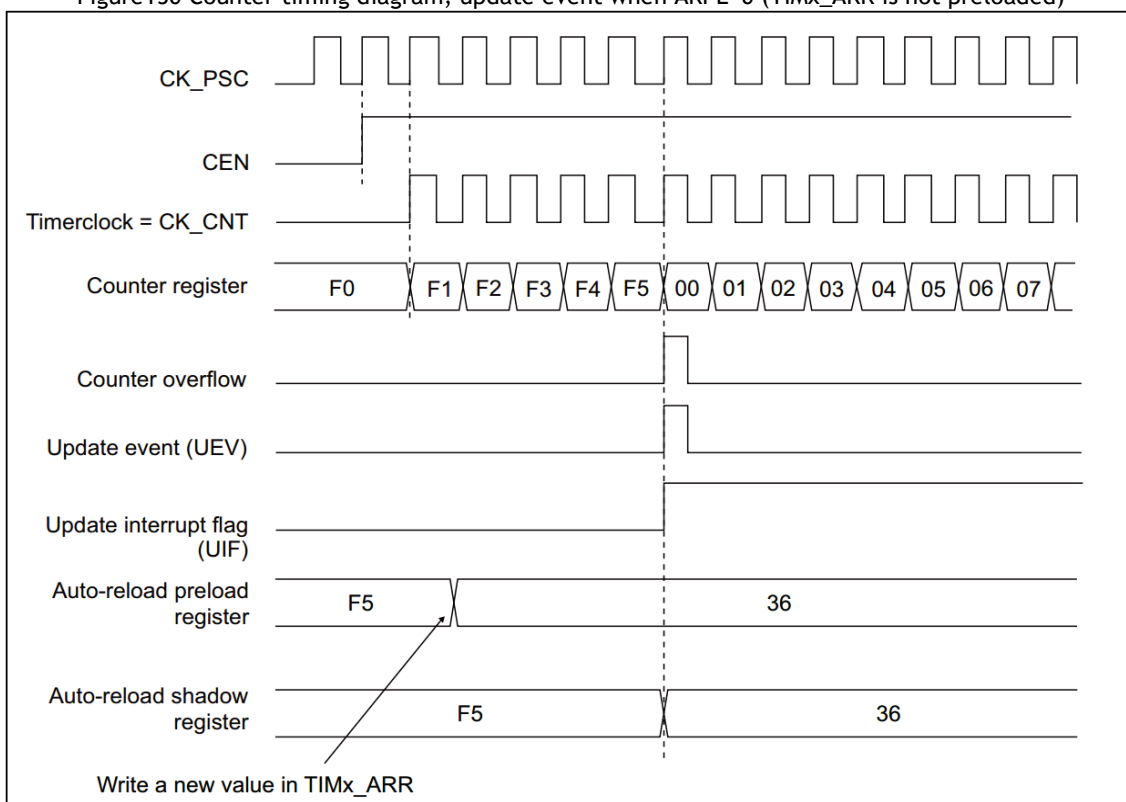


Figure151 Counter timing diagram, update event when ARPE=1 (preloaded with TIMx\_ARR)

### 15.3.3 Clock Selection

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

- Internal clock (CK\_INT)
- External clock mode 1 (for TIM9 and TIM12): External input pin (Tlx)
- Internal Trigger Input (ITRx) (for TIM9 and TIM12): Uses one timer as a prescaler for another timer, e.g., you can configure one timer, Timer1, as a prescaler for another timer, Timer2. See 13.3.15.

#### Internal Clock Source (CK\_INT)

If the slave mode controller is disabled (SMS=000 in the TIMx\_SMCR register), the CEN, DIR (TIMx\_CR1 register) and UG bits (TIMx\_EGR register) are de facto control bits and can only be modified by software (the UG bit is still cleared automatically). As long as the CEN bit is written to '1', the prescaler clock is provided by the internal clock CK\_INT.

The following figure shows the operation of the control circuit and up counter in the general mode without prescaler.

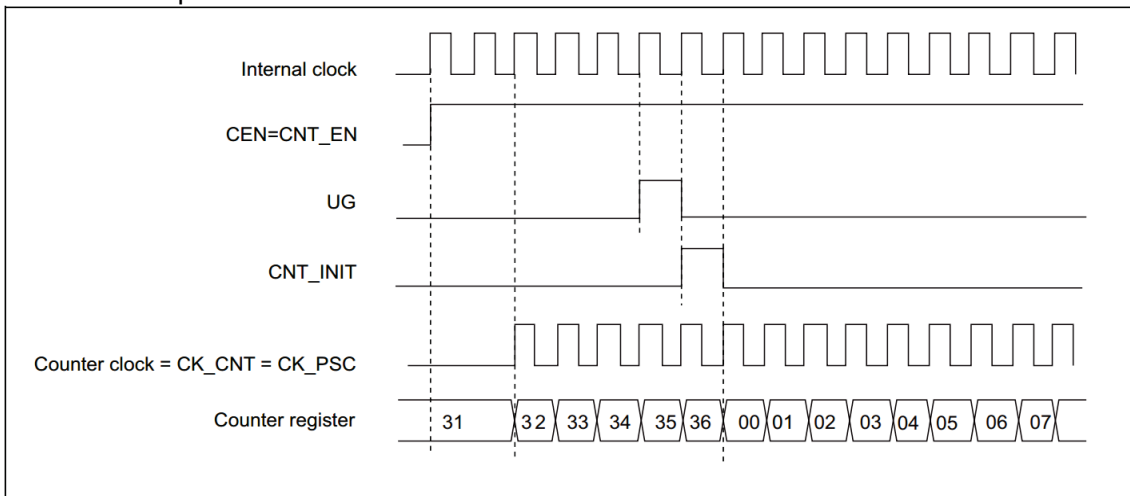


Figure152 Control circuit in general mode with internal clock division factor of 1

#### External Clock Source Mode 1 (TIM9 and TIM12)

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.

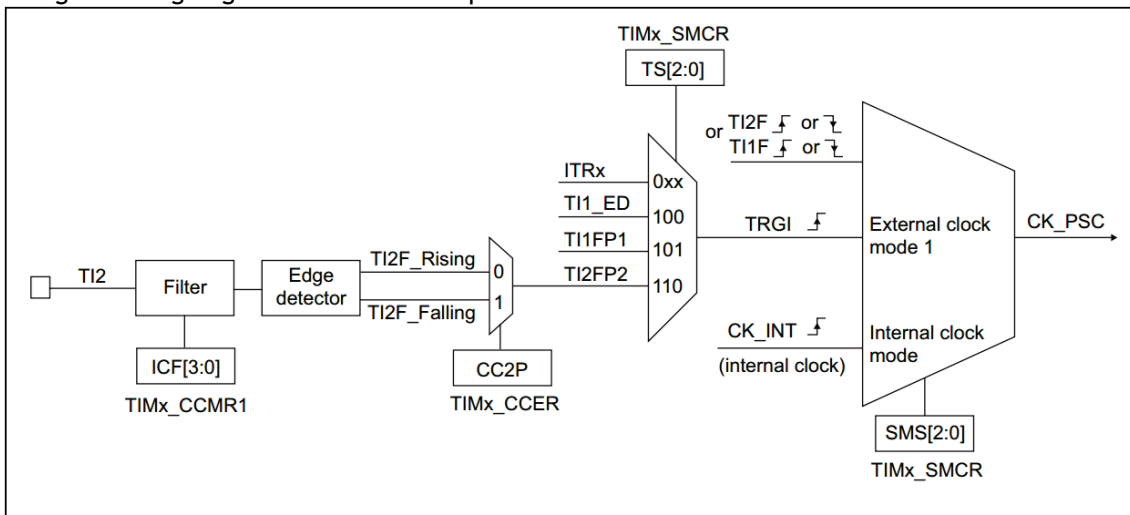


Figure153 TI2 External Clock Connection Example

For example, to configure the up counter to count on the rising edge of the TI2 input, use the following procedure:

1. Configure TIMx\_CMR1 register CC2S='01' to configure channel 2 to detect the rising edge of the TI2 input
2. Configure IC2F[3:0] of the TIMx\_CMR1 register to select the input filter bandwidth (if no filter is required, keep IC2F=0000)

Notes: The capture prescaler is not used as a trigger, so there is no need to configure it

3. Configure CC2P='0' in the TIMx\_CCER register to select the rising edge polarity
4. Configure the TIMx\_SMCR register with SMS='111' to select timer 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

When the rising edge occurs at TI2, the counter counts 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.

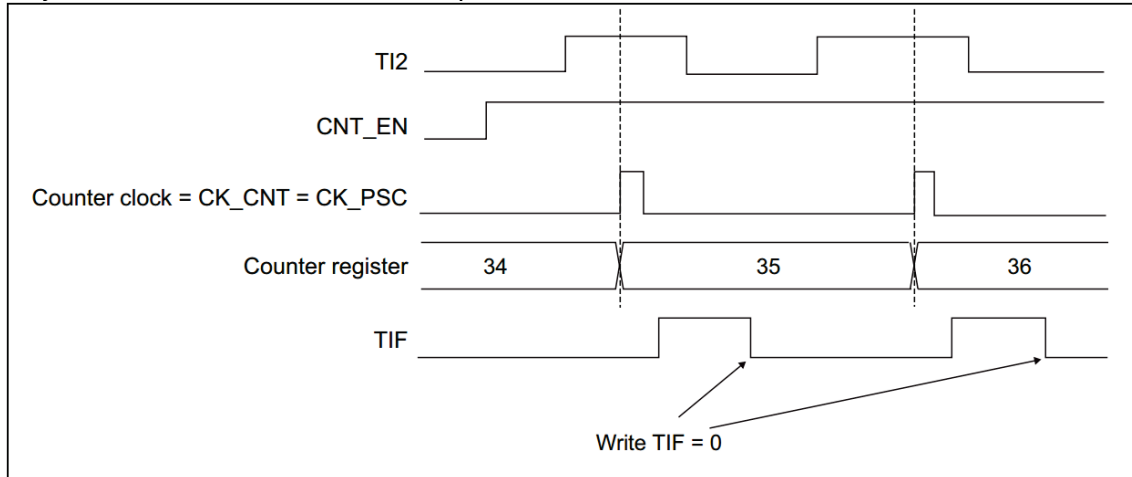


Figure154 Control circuit in external clock mode 1

### 15.3.4 Capture/Compare Channel

Each capture/compare channel is centered around a capture/compare register (containing shadow registers), including the input portion of the capture (digital filtering, multiplexing, and prescaler), and the output section (comparator and output control). The following diagrams give an overview of the capture/compare channels.

The input section samples the corresponding TIx input signal and generates a filtered signal TIxF. An edge detector with polarity selection then generates a signal (TIxFPx) that can be used as an input trigger from the mode controller or as a capture control. This signal is pre-divided into the capture register (ICxPS).

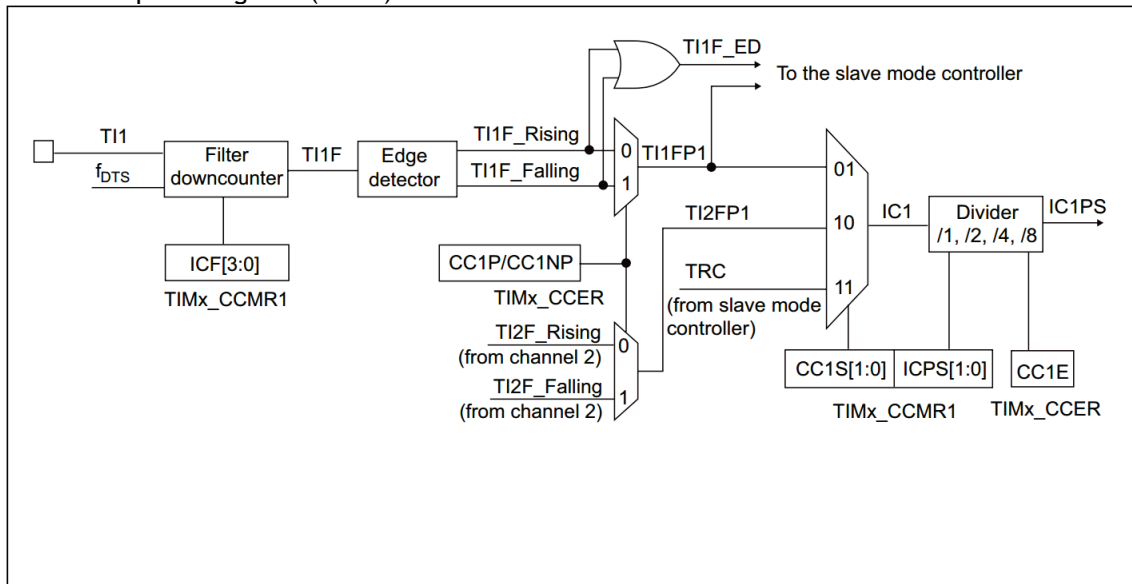


Figure155 Capture/compare channels (e.g. channel 1 input section)

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.

The diagram illustrates the OC1 output mode control logic. It features an 'Output mode controller' block that receives two inputs:  $CNT > CCR2$  and  $CNT = CCR2$ . This block is connected to a register labeled  $OC2M[2:0]$  under the identifier  $TIMx_CCMR1$ . The controller's output is a signal labeled  $OC1\_REF$ . This signal branches: one path goes to a multiplexer (MUX) with inputs 0 and 1, where input 1 is inverted and labeled  $CC1P$  (with  $TIMx\_CCER$  below it); the other path goes to an 'Output enable circuit' block, which also receives an input labeled  $CC1E$  (with  $TIMx\_CCER$  below it). The MUX output and the enable circuit's output are combined to produce the final  $OC1$  output signal.

can therefore sample the input signal 8 times (at fDTS frequency) to confirm a true edge shift on TI1, i.e. by writing IC1F=0011 in the TIMx\_CMR1 register.

3. To select the active conversion edge of the TI1 channel, write 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 the associated interrupt request by setting the CC1IE bit in the TIMx\_DIER register.

When an input capture occurs:

1. When a valid level transition is generated, the counter value is transferred to the TIMx\_CCR1 register.
2. The CC1IF flag is set (interrupt flag). When at least 2 consecutive captures occur and CC1IF has not been cleared, CC1OF is also set to '1'.
3. Interrupt generation depends on the CC1IE bit.

In order to handle capture overflows, it is recommended that data be read before the capture overflow flag is read in order to avoid losing capture overflow information that may be generated after the capture flag is read and before the data is read.

*Notes: Setting the corresponding CCxG bit in the TIMx\_EGR register allows you to generate input capture interrupts and / through software.*

### 15.3.6 PWM Input Mode (TIM9/12 only)

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, you need to measure the period (TIMx\_CCR1 register) and duty cycle (TIMx\_CCR2 register) of the PWM signal input to TI1 as follows (depending on the frequency of CK\_INT and the value of the prescaler)

1. Select the valid input of TIMx\_CCR1: Set CC1S=01 of TIMx\_CMR1 register (select TI1).
2. Select the active polarity of TI1FP1 (used to capture data into TIMx\_CCR1 and clear the counter): set CC1P=0 (active on rising edge).
3. To select a valid input for TIMx\_CCR2: Set CC2S=10 in the TIMx\_CMR1 register (selects TI1).
4. Select the active polarity of TI1FP2 (capture data to TIMx\_CCR2): set CC2P and CC2NP to 11 (active on falling edge).
5. To select a valid trigger input signal: set TS=101 in the TIMx\_SMCR register (select TI1FP1).
6. Configure the slave mode controller for reset mode: set SMS=100 in TIMx\_SMCR.
7. Enable Capture: Set CC1E=1 and CC2E=1 in TIMx\_CCER register.

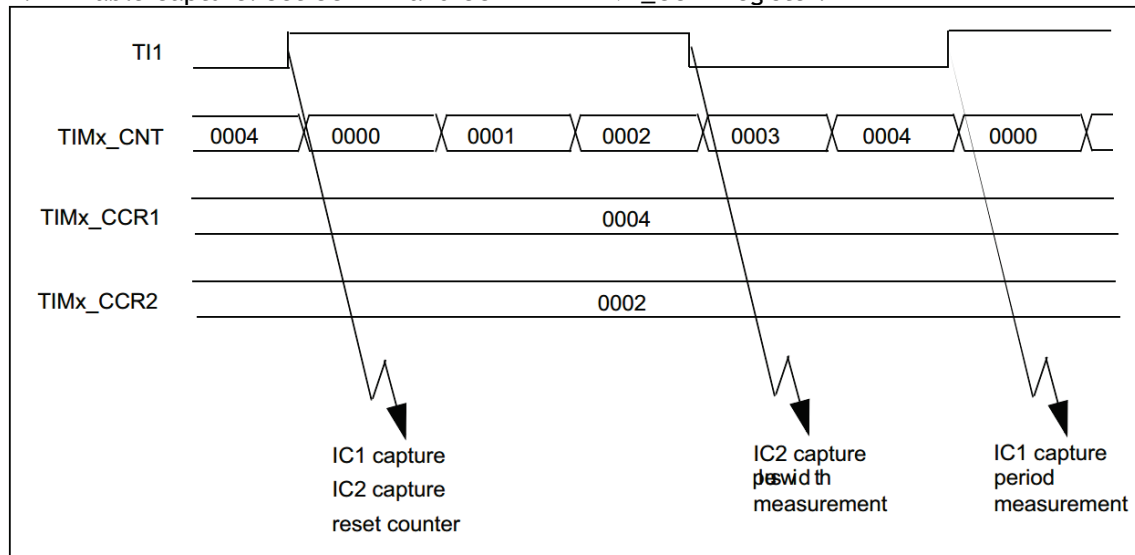


Figure 158 PWM Input Mode Timing



---

Since only TI1FP1 and TI2FP2 are connected to the Slave Mode Controller, only TIMx\_CH1/TIMx\_CH2 signals can be used for PWM input mode.

### 15.3.7 Forced Output Mode

In output mode (CCxS=00 in the TIMx\_CMRx register), the output compare signals (OCxREF and the corresponding OCx) can be forced to active or inactive state directly by the software, independently of the result of the comparison between the output compare register and the counter.

The output compare signal (OCxREF/OCx) is forced to active by setting the corresponding OCxM=101 in the TIMx\_CMRx register. In this way, OCxREF is forced high (OCxREF is always active high), and OCx gets the value of CCxP polarity bit reversed.

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 registers and counters is still performed and the corresponding flags are modified. Therefore the corresponding interrupt request is still generated. This will be described in the Output Compare Mode section below.

### 15.3.8 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:

1. 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. 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) when comparing matches.
2. Set the flag bit in the interrupt status register (CCxIF bit in the TIMx\_SR register).
3. An interrupt is generated if the corresponding interrupt mask (CCxIE bit in the TIMx\_DIER register) is set.

The OCxPE bit in TIMx\_CMRx selects whether or not the TIMx\_CCRx register requires the use of a preloaded register. In output compare mode, the update event UEV has no effect on the OCxREF and OCx outputs.

Synchronization can be done with an accuracy of up to one counting cycle of the counter. The 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. Selection of counter clock (internal, external, prescaler)
2. Write the corresponding data to the TIMx\_ARR and TIMx\_CCRx registers
3. To generate an interrupt request, set the CCxIE bit.
4. To select the output mode, e.g. to flip the OCx output pins when counter CNT matches CCRx, CCRx preload is not used, turn on the OCx outputs and high is active, you must set OCxM='011', OCxPE='0', CCxP='0' and CCxE='1'.
5. Setting the CEN bit of the TIMx\_CR1 register starts the counter

The TIMx\_CCRx register can be updated by software at any time to control the output waveform, provided that no preloaded registers are used (OCxPE='0', otherwise the TIMx\_CCRx shadow register can only be updated when the next update event occurs). An example is given in the following figure.

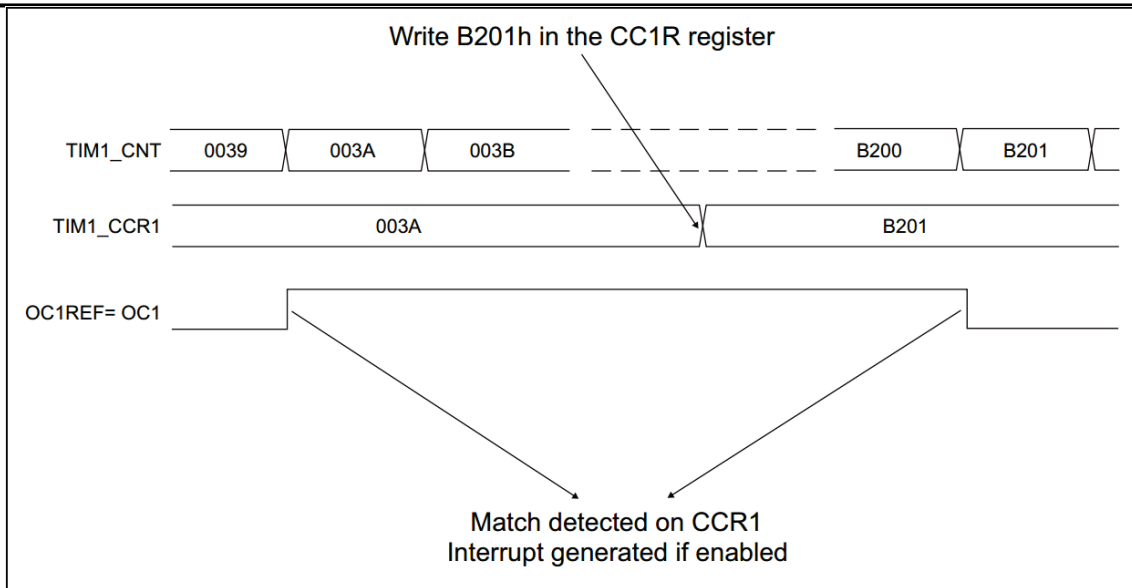


Figure159 Output Compare Mode, Flip OC1

### 15.3.9 PWM Mode

Pulse width modulation mode can generate 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 each of the OCx output channels to generate a single PWM. The TIMx\_CMRx register OCxPE bit must be set to enable the corresponding preloaded registers, and lastly the ARPE bit in the TIMx\_CR1 register must be set to enable the auto-reloaded preloaded registers. The TIMx\_CCMRx register OCxPE bit must be set to enable the corresponding preload registers, and finally the TIMx\_CR1 register ARPE bit must be set to enable the auto-reload preload register.

The 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.

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 CCxE bit in the TIMx\_CCER register controls the OCx output enable. The CCxE bit in the TIMx\_CCER register controls the OCx output enable. See the description of the TIMx\_CCERx 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  $TIMx\_CCRx \leq TIMx\_CNT$  or  $TIMx\_CNT \leq TIMx\_CCRx$  is met.

The timer can only generate PWM in edge-aligned mode when timed on the counter.

#### PWM Edge Alignment Mode

The following is an example of PWM mode 1. The PWM signal reference OCxREF is high when  $TIMx\_CNT < TIMx\_CCRx$  and low. If the comparison value in TIMx\_CCRx is greater than the auto-reload value (TIMx\_ARR), OCxREF remains '1'.

If the comparison value is 0, OCxREF is held at '0'. The following figure shows an example of an edge-aligned PWM waveform with TIMx\_ARR=8.

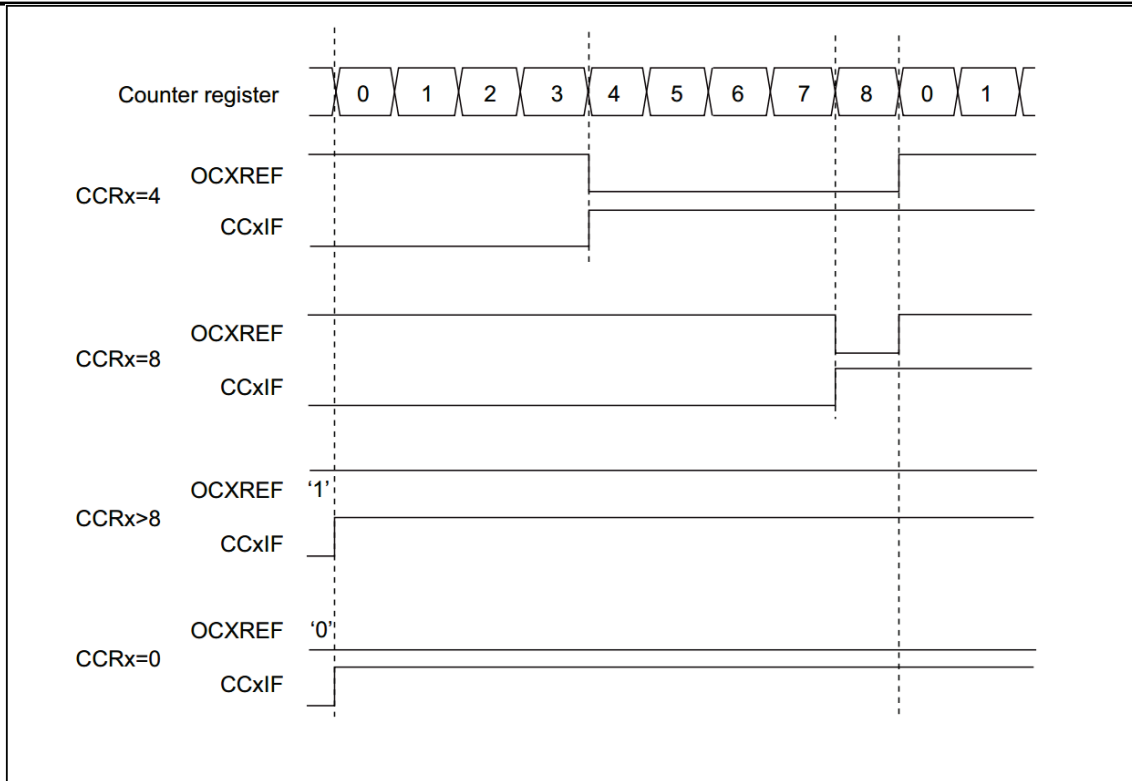


Figure160 Edge-aligned PWM waveform (ARR=8)

### 15.3.10 Single Pulse Mode

One-pulse mode (OPM) is a special case of one of the many modes described above. This mode allows the counter to respond to an excitation and, after a programmable delay, generate a pulse with a programmable pulse width.

The counter can be started from the mode controller to generate waveforms in output compare mode or PWM mode. Setting the OPM bit in the TIMx\_CR1 register selects 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:  $CNT < CCRx \leq ARR$  (specifically,  $0 < CCRx$ )

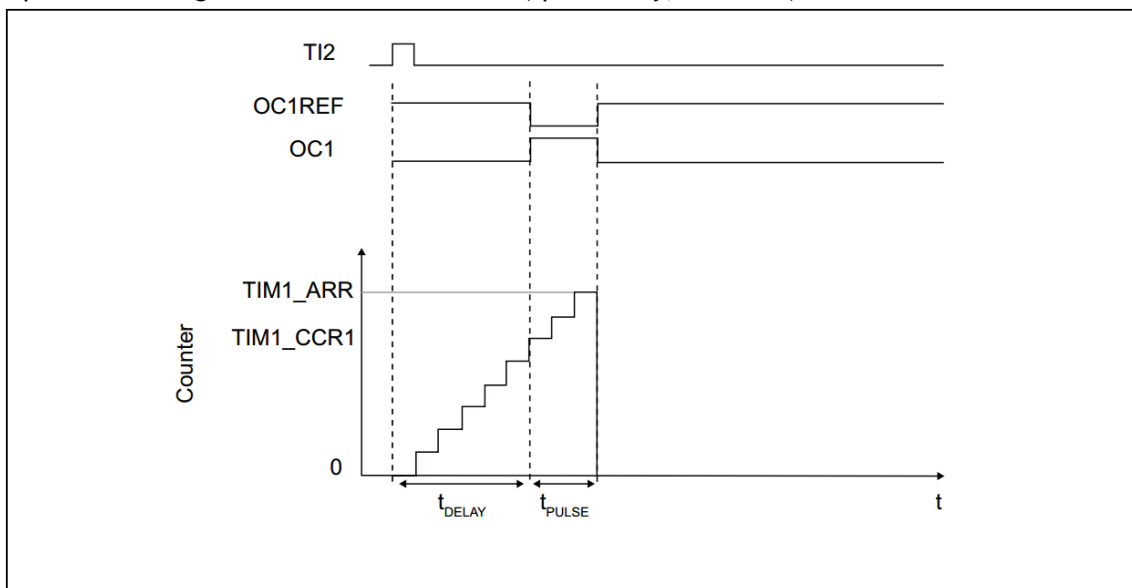


Figure161 Example of single pulse mode

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.

Assuming TI2FP2 as trigger 1.

1. Set CC2S='01' in the TIMx\_CMR1 register to map TI2FP2 to TI2.
2. Set CC2P='0' and CC2NP='0' in the TIMx\_CCER register to enable the TI2FP2 to detect the rising edge.
3. Set TS='110' in the TIMx\_SMCR register and the TI2FP2 triggers as a slave mode controller (TRGI).
4. 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)

- tDELAY is defined by the value written to the TIMx\_CCR1 register.
- tPULSE is defined by the difference between the autoloading 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 preload value; first, set OC1M="111" in the TIMx\_CCMR1 register to enter PWM mode 2; according to the need, we should set OC1M="111" in the TIMx\_CCMR1 register to enter PWM mode 2. When the counter reaches the preloaded value, a waveform from "1" to "0" should be generated; first, set OC1M in TIMx\_CCMR1 register to "111" to enter PWM mode 2; selectively enable the preload registers according to the needs: set OC1PE in TIMx\_CMR1 to "1" and set OC1PE in TIMx\_CCMR1 to "1" and set OC1PE in TIMx\_CCMR1 to "1" to "1". Set OC1PE="1" in TIMx\_CMR1 and ARPE in TIMx\_CR1 register; then fill in the comparison value in TIMx\_CCR1 register and the auto-load value in TIMx\_ARR register, modify the UG bit to generate an update event, and then wait for an external triggering 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 required, 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-load 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. Comparison operations between the counter and the compare value then generate the output transition. However, these operations require a certain number of clock cycles, which limits the minimum delay time tDELAY that can be obtained.

If you want to output waveforms with minimum delay, you can set the OCxFE bit in the TIMx\_CMRx register; at this point, OCxREF (and OCx) are forced to respond to the excitation without relying on the result of the comparison anymore, and the output waveform is the same as that when the comparison is matched. OCxFE only works when the channel is configured for PWM1 and PWM2 modes.

### 15.3.11 Synchronization of TIM9/12 Timers and External Triggers

The TIM9/12 timers can be synchronized with an external trigger in a variety of modes: reset mode, gated mode, and trigger 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 generated if the URS bit of the TIMx\_CR1 register is low; all preloaded 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:

1. 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 it does not need to be configured. the CC1S bit selects the input capture source only, i.e., CC1S=01 in the TIMx\_CMR1 register. set CC1P and CC1NP in the TIMx\_CCER register to 0 to determine the polarity (only the rising edge is detected).
2. 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.
3. 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 of TI1 occurs; at this point, the counter is cleared to zero counts again from zero. At the same time, the trigger flag (TIF bit in the TIMx\_SR register) is set, generating an interrupt request based on the setting of the TIE (interrupt enable) bit in the TIMx\_DIER. 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 circuitry at the TI1 input.

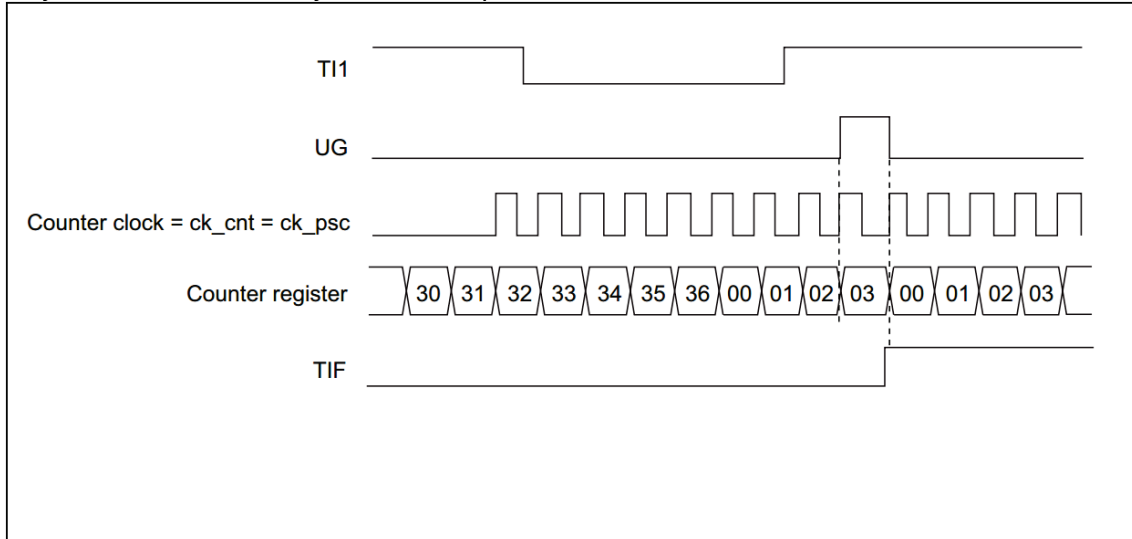


Figure162 Control circuit in reset mode

#### 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:

1. Configure channel 1 to detect a low level on TI1. Configure the input filter bandwidth (in this example, 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, set CC1S=01 in the TIMx\_CMR1 register. set CC1P=1 and CC1NP=0 in the TIMx\_CCER register to determine the polarity (low level detection only).
2. 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.
3. Setting CEN=1 in the TIMx\_CR1 register starts the counter. In gated mode, if CEN=0, the counter will not start, regardless of the trigger input level.

The counter starts counting according to the internal clock as long as TI1 is low and stops counting when TI1 goes high. The TIF flag in TIMx\_SR is set when 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.

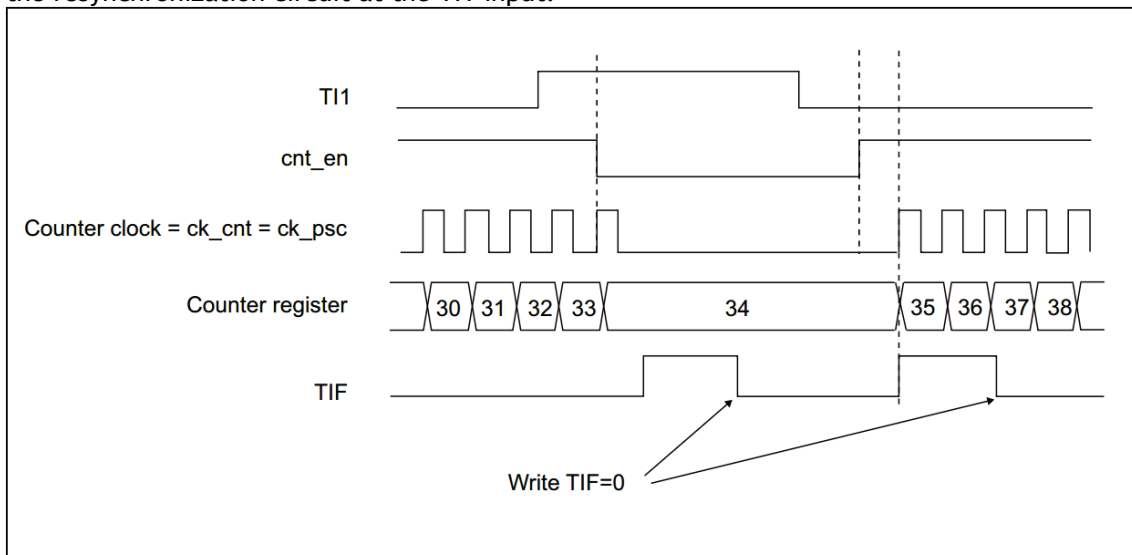


Figure163 Control Circuit in Gated Mode

### 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:

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 need to be configured. the CC2S bit is only used to select the input source, set CC2S=01 in the TIMx\_CMR1 register. set CC2P=1 and CC2NP=0 in the TIMx\_CCER register to determine the polarity (detects low levels only).
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 occurs in 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.

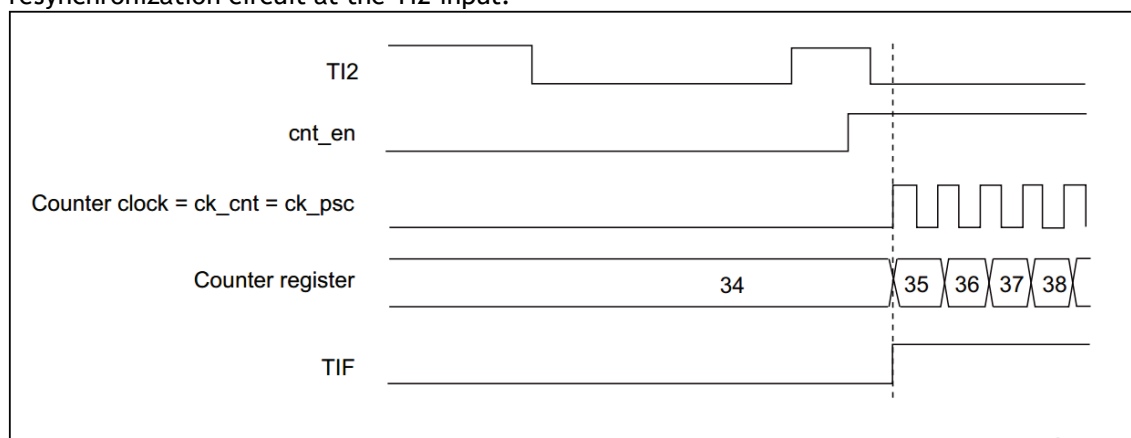


Figure164 Control Circuit in Trigger Mode

### 15.3.12 Timer Synchronization (TIM9/12)

TIM timers are internally linked together for timer synchronization or chaining. For more information, see Section14.3.15 :Timer Consistency.

*Notes: The slave timer's clock must be enabled before receiving events from the master timer and must not be changed at runtime while receiving triggers from the master timer.*

### 15.3.13 Debug Mode

When the microcontroller enters debug mode (the Cortex-M3 core is stopped), the TIMx counter either continues normal operation or stops, depending on the DBG\_TIMx\_STOP setting in the DBG module. For more details, see the subsequent section31.16.2 : Debugging bxCAN and I2Csupport for timer, watchdog, .

## 15.4 TIM9/TIM12 Registers

See Section1 for details on the abbreviations used in the register descriptions.

These peripheral registers can be operated in half-word (16-bit) or word (32-bit) format.

### 15.4.1 TIM9/12 Control Register 1 (TIMx\_CR1)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CKD[1:0]		ARPE	Reserved			OPM	URS	UDIS	CEN
						rw	rw	rw				rw	rw	rw	rw
Bit		notation		clarification											
15:10		Reserved		Reserved, always reads 0.											
9:8		CKD[1:0]		<b>CKD[1:0]:</b> Clock division factor (Clock division) Defines the dividing ratio between the timer clock (CK_INT) frequency and the sampling frequency used by the digital filter (ETR, Tlx). 00: tDTS=tCK_INT 01: tDTS=2xtCK_INT 10: tDTS=4xtCK_INT											

		11: Reserved
7	ARPE	<b>ARPE:</b> Auto-reload preload enable bit 0: TIMx_ARR register is not buffered; 1: The TIMx_ARR register is loaded into the buffer.
6:4	Reserved	Reserved, always reads 0.
3	OPM	<b>OPM:</b> 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	<b>URS:</b> Update request source The software selects the source of the UEV event with this bit. 0: If the update interrupt request is enabled, either of the following events generates an update interrupt request: -Counter overflow/underflow -Setting the UG bit 1: If the update interrupt request is enabled, the update interrupt request is generated only for counter overflow/underflow.
1	UDIS	<b>UDIS:</b> Update disable Software allows/disallows the generation of UEV events with this bit 0: UEV is allowed. the update (UEV) event is generated by either of the following events: -Counter overflow/underflow -Setting the UG bit -Updates generated from the mode controller Registers with caches are loaded with their preloaded values. (Translation: updating shadow registers) 1: Disable UEV. no update event is generated and the shadow registers (ARR, PSC, CCRx) maintain their values. If the UG bit is set or a hardware reset is issued from the mode controller, the counters and prescalers are reinitialized.
0	CEN	<b>CEN:</b> Enable Counter 0: Disable the counter; 1: Enable the counter. Note: In single pulse mode, CEN is automatically cleared when an update event occurs.

## 15.4.2 TIM9/12 Slave Mode Control Register (TIMx\_SMCR)

Offset address: 0x08

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									TS[2:0]			Reserv ed	SMS[2:0]		
									rW	rW	rW		rW	rW	rW

Bit	notation	clarification	
15:7	Reserved	Reserved	
6:4	TS[2:0]	<b>TS[2:0]:</b> Trigger selection These 3-bit selections are used to synchronize the trigger input of the counter. 000: Internal trigger 0 (ITR0), TIM1 001: Internal Trigger 1 (ITR1), TIM2 010: Internal Trigger 2 (ITR2), TIM3 011: Internal Trigger 3 (ITR3), TIM4 SeeTable79 for details on ITRx in each timer. Note: These bits can only be changed when they are not used (e.g. SMS=000) to avoid false edge detection when changed.	100: Edge detector for TI1 (TI1F_ED) 101: Filtered Timer Input 1 (TI1FP1) 110: Filtered Timer Input 2 (TI2FP2) 111: Reserved
3	Reserved	Reserved, always reads 0.	
2:0	SMS[2:0]	<b>SMS[2:0]:</b> Slave mode selection 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 Input Control Register and Control Register) 000: Off Slave Mode - If CEN=1, the prescaler is driven directly from the internal clock. 001: Reserved 010: Reserved 011: Reserved 100: Reset Mode-The rising edge of the selected trigger input (TRGI) reinitializes the counter and generates a signal to update the register. 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 are controlled. 110: Trigger Mode - The counter is started (but not reset) on the rising edge of the	



		<p>trigger input TRGI, and 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 the gated 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 gating mode is to check the level of the trigger input.</p> <p>Note:The slave timer's clock must be enabled before receiving events from the master timer and must not be changed at runtime while receiving triggers from the master timer.</p>
--	--	--

Table79 TIMx Internal Trigger Connections

From Timer	ITR0(TS=000)	ITR1 (TS=001)	ITR2 (TS=010)	ITR3 (TS=011)
TIM9	TIM2_TRG0	TIM3_TRG0	TIM10_OC	TIM11_OC
TIM12	TIM4_TRG0	TIM5_TRG0	TIM13_OC	TIM14_OC

### 15.4.3 TIM9/12 Interrupt Enable Register (TIMx\_DIER)

Offset address: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									TIE	Reserved			CC2IE	CC1IE	UIE
									rw				rw	rw	rw

Bit	notation	clarification
15:7	Reserved	Reserved, always reads 0.
6	TIE	TIE: Trigger interrupt tenable 0: Disable triggering of interrupts; 1: Enable trigger interrupt.
5:3	Reserved	Reserved, always reads 0.
2	CC2IE	CC2IE: Allow Capture/Compare2 interrupt enable 0: Capture/Compare 2 interrupt disabled; 1: Allow capture/compare 2 interrupts.
1	CC1IE	CC1IE: Allow Capture/Compare1 interrupt enable 0: Capture/Compare 1 interrupt disabled; 1: Capture/Compare 1 interrupt allowed.
0	UIE	UIE: Updateinterruptenable 0: Disable update interrupt; 1: Allow updates to be interrupted.

### 15.4.4 TIM9/12 Status Register (TIMx\_SR)

Offset address: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					CC2OF	CC1OF	Reserved		TIF	Reserved			CC2IF	CC1IF	UIF
					rcw0	rcw0			rcw0				rcw0	rcw0	rcw0

Bit	notation	clarification
15:11	Reserved	Reserved, always reads 0.
10	CC2OF	CC2OF: Capture/Compare2 overcapture flag See CC1OF for description.
9	CC1OF	CC1OF: Capture/Compare1 overcapture flag This flag can be set '1' by hardware only if the corresponding channel is configured for input capture. Writing '0' clears this bit. 0: No duplicate captures are generated; 1: The state of CC1IF is already '1' when the counter value is captured in the TIMx_CCR1 register.
8:7	Reserved	Reserved, always reads 0.
6	TIF	TIF: Trigger Interrupt Flag (TIF) It is '1' by hardware to this position 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 software. 0: No trigger event is generated; 1: Trigger interrupt waiting for response.
5:3	Reserved	Reserved, always reads 0.
2	CC2IF	CC2IF: Capture/Compare2 interrupt flag Refer to the CC1IF description.
1	CC1IF	CC1IF: Capture/Compare1 interrupt flag



		<p>If channel CC1 is configured for output mode: This bit is set '1' by hardware when the counter value matches the comparison value, except in centrosymmetric mode (refer to the CMS bit in the TIMx_CR1 register). It is cleared '0' by 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 '1' by hardware when a capture event occurs, and it is cleared '0' by software or by reading TIMx_CCR1. 0: No input capture is generated; 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	<p><b>UIF:</b> Update interrupt flag (Update interrupt flag) This bit is set '1' by hardware when an update event is generated. It is cleared '0' by software. 0: No update event is generated; 1: Update interrupt wait response. This bit is set '1' by hardware when the register is updated: -If UDIS=0 and URS=0 in the TIMx_CR1 register, an update event is generated when UG=1 in the TIMx_EGR register (software re-initializes the counter CNT); -If UDIS=0 and URS=0 in the TIMx_CR1 register, an update event is generated when the counter CNT is reinitialized by the trigger event. (Refer to the description of the synchronization control register)</p>

### 15.4.5 TIM9/12 Event Generation Register (TIMx\_EGR)

Offset address:0x14

Reset value:0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									TG	Reserved			CC2G	CC1G	UG
									W				W	W	W

Bit	notation	clarification
15:7	Reserved	Reserved, always reads 0.
6	TG	<p><b>TG:</b> Trigger generation This bit is set '1' by software to generate a trigger event that is automatically cleared '0' by hardware. 0: No action; 1: TIF=1 in the TIMx_SR register generates the corresponding interrupt if it is turned on.</p>
5:3	Reserved	Reserved, always reads 0.
2	CC2G	<p><b>CC2G:</b> Capture/compare2 generation Refer to the CC1G description.</p>
1	CC1G	<p><b>CC1G:</b> Capture/compare1 generation This bit is set '1' by software to generate a capture/compare event and is automatically cleared '0' by hardware. 0: No action; 1: Generate a capture/compare event on channel CC1: If channel CC1 is configured as an output: Set CC1IF=1 to generate the corresponding interrupt if it is turned on. If channel CC1 is configured as an input: The current counter value is captured to the TIMx_CCR1 register; set CC1IF=1 to generate the corresponding interrupt if it is turned on. If CC1IF is already 1, set CC1OF=1.</p>
0	UG	<p><b>UG:</b> Update generation This bit is set '1' by software and cleared '0' automatically by hardware. 0: No action; 1: Reinitialize the counter and generate an update event. Note that the prescaler counter is also cleared '0' (but the prescaler coefficients remain unchanged). The counter is cleared '0' if in centrosymmetric mode or DIR=0 (counting up), if DIR=1 (counting down) the counter takes the value of TIMx_ARR.</p>

## 15.4.6 TIM9/12 Capture/Compare Mode Register 1 (TIMx\_CMR1)

Offset address: 0x18

Reset value: 0x0000

The channel can be used as input (capture mode) or output (compare mode), and the direction of the channel is defined by the corresponding CCxS. The other bits of this register function differently in input and output modes. OCxx describes the function of the channel in output mode and ICxx describes the function of the channel in output mode. It is therefore important to note that the same bit functions differently in output mode and input mode.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2:0]			OC2PE	OC2FE	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	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW

### Output comparison mode:

Bit	notation	clarification
15	OC2CE	OC2CE: Output compare2 clear enable
14:12	OC2M[2:0]	OC2M[2:0]: Output compare2 mode
11	OC2PE	OC2PE: Output compare2 preload enable
10	OC2FE	OC2FE: Output compare2 fast enable
9:8	CC2S[1:0]	CC2S[1:0]: Capture/Compare2 selection This bit defines the direction of the channel (input/output), and the selection of the input pin: 00: The CC2 channel is configured as an output; 01: CC2 channel is configured as an input and IC2 is mapped on TI2; 10: The CC2 channel is configured as an input and IC2 is mapped on TI1; 11: The CC2 channel is configured as an input with IC2 mapped on TRC. This mode operates only when the internal trigger input is selected (by the TS bit of the TIMx_SMCR register). Note: CC2S is writable only when the channel is closed (CC2E='0' in the TIMx_CCER register).
7	OC1CE	OC1CE: Output compare1 clear enable 0: OC1REF is not affected by the ETRF input; 1: Clear OC1REF=0 as soon as the ETRF input is detected high.
6:4	OC1M[2:0]	OC1M[2:0]: Output compare1 mode (Output compare1 enable) These 3 bits define the action of the output reference signal OC1REF, which determines the value of OC1. OC1REF is active high, and the active level of OC1 depends on the CC1P bit. 000: Freeze. Comparison between output comparison register TIMx_CCR1 and counter TIMx_CNT does not work for OC1REF; 001: Set channel 1 to active level when matched. Forces OC1REF high when the value of counter TIMx_CNT is the same as capture/compare register 1 (TIMx_CCR1). 010: Set channel 1 to an invalid level when matched. Forces OC1REF low when the value of counter TIMx_CNT is the same as capture/compare register 1 (TIMx_CCR1). 011: Flip. Flips the level of OC1REF when TIMx_CCR1 = TIMx_CNT. 100: Forces an invalid level. Forces OC1REF low. 101: Force to active level. Forces OC1REF high. 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 active ( OC1REF=1). 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. 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	OC1PE: Output compare1 preload enable 0: Disables the preload function of the TIMx_CCR1 register, which can be written to the TIMx_CCR1 register at any time and the newly written value takes effect immediately. 1: Enable the preload function of TIMx_CCR1 register, read/write operation only operates on the preloaded register, the preloaded value of TIMx_CCR1 is transferred to the current register when the update event arrives. Note: Only in single pulse mode (OPM='1' in TIMx_CR1 register), PWM mode can be used without checking the preload register, otherwise its action is uncertain.
2	OC1FE	OC1FE: Output compare1 fast enable This bit is used to speed up the response of the CC output to trigger input events. 0: Depending on the value of the counter and CCR1, CC1 operates normally, even if

		<p>the trigger is open. The minimum delay to activate the CC1 output is 5 clock cycles when a valid edge occurs at the input of the flip-flop.</p> <p>1: The active edge of the input to the flip-flop acts as if a comparison match has occurred. Therefore, OC is set to the comparison level independent of the comparison result. The delay between the active edge of the sampling flip-flop and the CC1 output is reduced to 3 clock cycles.</p> <p>This bit only functions when the channel is configured for PWM1 or PWM2 mode.</p>
1:0	CC1S[1:0]	<p><b>CC1S[1:0]:</b> Capture/Compare1 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: CC1 channel is configured as an input and IC1 is mapped on TI2;</p> <p>11: The CC1 channel is configured as an input with IC1 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: CC1S is writable only when the channel is closed (CC1E='0' in the TIMx_CCER register).</p>

#### Input Capture Mode:

Bit	notation	clarification
15:12	IC2F[3:0]	<b>IC2F[3:0]:</b> Input capture2 filter
11:10	IC2PSC[1:0]	<b>IC2PSC[1:0]:</b> input/capture2 prescaler (input capture2 prescaler)
9:8	CC2S[1:0]	<p><b>CC2S[1:0]:</b> Capture/compare2selection</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 with IC2 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: CC2S is writable only when the channel is closed (CC2E='0' in the TIMx_CCER register).</p>
7:4	IC1F[3:0]	<p><b>IC1F[3:0]:</b> Input capture1 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 records N events and then generates a jump in the output:</p> <p>0000: no filter, sampling at fDTS 1000: sampling frequency fSAMPLING = fDTS/8, N = 6</p> <p>0001: Sampling frequency fSAMPLING=fCK_INT, N=2 1001: Sampling frequency fSAMPLING=fDTS/8, N=8</p> <p>0010: Sampling frequency fSAMPLING=fCK_INT, N=4 1010: Sampling frequency fSAMPLING=fDTS/16, N=5</p> <p>0011: Sampling frequency fSAMPLING=fCK_INT, N=8 1011: Sampling frequency fSAMPLING=fDTS/16, N=6</p> <p>0100: Sampling frequency fSAMPLING=fDTS/2, N=6 1100: Sampling frequency fSAMPLING=fDTS/16, N=8</p> <p>0101: Sampling frequency fSAMPLING=fDTS/2, N=8 1101: Sampling frequency fSAMPLING=fDTS/32, N=5</p> <p>0110: Sampling frequency fSAMPLING=fDTS/4, N=6 1110: Sampling frequency fSAMPLING=fDTS/32, N=8</p> <p>0111: Sampling frequency fSAMPLING=fDTS/4, N=8 1111: Sampling frequency fSAMPLING=fDTS/32, N=8</p> <p>Note: When ICxF[3:0] = 1, 2 or 3, fDTS in the formula is replaced by CK_INT.</p>
3:2	IC1PSC[1:0]	<p><b>IC1PSC[1:0]:</b> input/capture1 prescaler (Input capture1 prescaler)</p> <p>These 2 bits define the prescaling factor for the CC1 input (IC1).</p> <p>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]	<p><b>CC1S[1:0]:</b> Capture/Compare1 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: CC1 channel is configured as an input and IC1 is mapped on TI2;</p> <p>11: The CC1 channel is configured as an input with IC1 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: CC1S is writable only when the channel is closed (CC1E='0' in the TIMx_CCER register).</p>

## 15.4.7 TIM9/12 Capture/Compare Enable Register (TIMx\_CCER)

Offset address: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								CC2NP	Reserved	CC2P	CC2E	CC1NP	Reserved	CC1P	CC1E
								rw		rw	rw	rw		rw	rw

Bit	notation	clarification
15:8	Reserved	Reserved, always reads 0.
7	CC2NP	CC2NP: capture/compare 2 output polarity see CC1NP description
6	Reserved	Reserved, always reads 0.
5	CC2P	CC2P: Input/Capture2 output polarity (Capture/Compare2 output polarity) Refer to the description of CC1P.
4	CC2E	CC2E: Capture/Compare2 output enable Refer to the description of CC1E.
3	CC1NP	CC1NP: Capture/compare 1 complementary output polarity CC1 channel configured as output:CC1NP must remain clear The CC1 channel is configured as an input:CC1NP is used in conjunction with CC1P to define TI1FP1/TI2FP1 polarity (see CC1P description).
2	Reserved	Reserved, always reads 0.
1	CC1P	CC1P: Input/Capture1 output polarity (Capture/Compare1 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: This bit selects whether IC1 or the inverted signal of IC1 is used as the trigger or capture signal. 0: not inverted: capture occurs on the rising edge of IC1; when used as an external trigger, IC1 is not inverted. 1: inverted: capture occurs on the falling edge of IC1; when used as an external trigger, IC1 is inverted.
0	CC1E	CC1E: Input/Capture1 output enable (Capture/Compare1 output enable) The CC1 channel is configured as an output: 0: Off - OC1 disables output. 1: Turn on -OC1 signal output to the corresponding output pin. The CC1 channel is configured as an input: This bit determines whether the counter value can be captured into the TIMx_CCR1 register. 0: Capture disabled; 0: Capture enable.

Table80 Output control bits for standard OCx channels

CCxE bit	OCx Output Status
0	Disable output (OCx=0, OCx_EN=0)
1	OCx = OCxREF + polarity, OCx_EN = 1

Notes: The status of the external I/O pins connected to the standard OCx channel depends on the OCx channel status and the GPIO and AFIO registers.

## 15.4.8 TIM9/12 Counter (TIMx\_CNT)

Offset address: 0x24

Reset value: 0x0000

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

Bit	notation	clarification
15:0	CNT [15:0]	CNT[15:0]: Counter value

## 15.4.9 TIM9/12 Prescaler (TIMx\_PSC)

Offset address: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC [15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit	notation	clarification													
15:0	PSC [15:0]	PSC[15:0]: Prescaler value (Prescaler value) The clock frequency of the counter, CK_CNT, is equal to fCK_PSC/(PSC[15:0]+1). The PSC contains the value loaded into the current prescaler register when the update event is generated.													

## 15.4.10 TIM9/12 Automatic Reload Register (TIMx\_ARR)

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	notation	clarification													
15:0	ARR[15:0]	ARR[15:0]:Auto reload value The ARR contains the values that will be transferred to the actual auto-reload registers. Refer to the15.3.1 section for details: updates and actions related to the ARR. The counter does not work when the value of Auto-Reload is empty.													

## 15.4.11 TIM9/12 Capture/Compare Register 1 (TIMx\_CCR1)

Offset address: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro
Bit	notation	clarification													
15:0	CCR1[15:0]	<b>CCR1[15:0]:</b> Capture/Compare1 value (Capture/Compare1 value) 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), 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. The current capture/comparison register participates in the comparison with counter TIMx_CNT and generates an output signal on port OC1. If the CC1 channel is configured as an input: CCR1 contains the counter value transmitted by the last input capture 1 event (IC1).													

## 15.4.12 TIM9/12 Capture/Compare Register 2 (TIMx\_CCR2)

Offset address: 0x38

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro
Bit	notation	clarification													
15:0	CCR2[15:0]	CCR2[15:0]: Capture/Compare2 value													
		If the CC2 channel is configured as an output:													
		CCR2 contains the value loaded into the current Capture/Compare 2 register (preloaded value).													
		If the preload feature is not selected in the TIMx_CMR2 register (OC2PE bit), the written value is immediately transferred to the current register. Otherwise this preloaded value is transferred to the current Capture/Compare2 register only when an update event occurs.													
		The current capture/comparison register participates in the comparison with counter TIMx_CNT and generates an output signal on port OC2.													

The following table maps all registers of the TIM9/12 into a 16-bit addressable (addressed) space.

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h	TIMx_CR1	Reserved																						CKD [1:0]		ARPE	Reserved			OPM	URS	UDIS	CEN
	Reset value																							0	0								
008h	TIMx_SMCR	Reserved																						TS [2:0]			Reserved	SMS [2:0]					
	0																							0	0								
00Ch	TIMx_DIER	Reserved																						T1E	Reserved			CC2IE	CC1IE	UIE			
	0																																
010h	TIMx_SR	Reserved																				CC2OF	CC1OF	Reser ved	TI1	Reserved			CC2IF	CC1IF	UIF		
	0																															0	0
014h	TIMx_EGR	Reserved																						TG	Reserved			CC2G	CC1G	UG			
	0																																
018h	TIMx_CMR1 Output Compare Mode	Reserved																OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		Reserved	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]		
	0																	0	0	0			0	0		0	0						
	TIMx_CMR1 Input Capture Mode	Reserved																IC2F [3:0]			IC2 PSC [1:0]	CC2S [1:0]		IC1F [3:0]			IC1 PSC [1:0]	CC1S [1:0]					
	0																	0	0	0		0	0	0	0	0		0	0				
01Ch	Reserved																																
020h	TIMx_CCER	Reserved																						CC2NP	Reserved	CC2P	CC2E	CC1NP	Reserved	CC1P	CC1E		
	0																																
024h	TIMx_CNT	Reserved																CNT [15:0]															
	0																	0	0	0	0	0	0	0	0	0	0	0	0	0			
028h	TIMx_PSC	Reserved																PSC [15:0]															
	0																	0	0	0	0	0	0	0	0	0	0	0	0	0			
02Ch	TIMx_ARR	Reserved																ARR[15:0]															
	0																	0	0	0	0	0	0	0	0	0	0	0	0	0			
030h	Reserved																																
034h	TIMx_CCR1	Reserved																CCR1[15:0]															
	0																	0	0	0	0	0	0	0	0	0	0	0	0	0			
038h	TIMx_CCR2	Reserved																CCR2[15:0]															
	0																	0	0	0	0	0	0	0	0	0	0	0	0	0			
03Ch to	Reserved																																

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
04Ch																																	

See Table 1 for register start addresses.

## 15.5 TIM10/11/13/14 Registers

See Section 1 for details on the abbreviations used in the register descriptions.

These peripheral registers can be operated in half-word (16-bit) or word (32-bit) format.

### 15.5.1 TIM10/11/13/14 Control Register 1 (TIMx\_CR1)

Offset address: 0x00

Reset value: 0x0000

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

Bit	notation	clarification
15:10	Reserved	Reserved, always reads 0.
9:8	CKD[1:0]	<b>CKD[1:0]:</b> Clock division factor (Clock division) Defines the dividing ratio between the timer clock (CK_INT) frequency and the sampling frequency used by the digital filter (ETR, Tlx). 00: tDTS=tCK_INT 01: tDTS=2xtCK_INT 10: tDTS=4xtCK_INT 11: Reserved
7	ARPE	<b>ARPE:</b> Auto-reload preload enable bit 0: TIMx_ARR register is not buffered; 1: The TIMx_ARR register is loaded into the buffer.
6:4	Reserved	Reserved, always reads 0.
3	OPM	<b>OPM:</b> 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	<b>URS:</b> Update request source The software selects the source of the UEV event with this bit. 0: If the update interrupt request is enabled, either of the following events generates an update interrupt request: -Counter overflow/underflow -Setting the UG bit -Updates generated from the mode controller 1: If the update interrupt request is enabled, the update interrupt request is generated only for counter overflow/underflow.
1	UDIS	<b>UDIS:</b> Update disable Software allows/disallows the generation of UEV events with this bit 0: UEV is allowed. the update (UEV) event is generated by any of the following events: -Counter overflow/underflow -Setting the UG bit -Updates generated from the mode controller Registers with caches are loaded with their preloaded values. (Translation: updating shadow registers) 1: Disable UEV. no update event is generated and the shadow registers (ARR, PSC, CCRx) maintain their values. If the UG bit is set or a hardware reset is issued from the mode controller, the counters and prescalers are reinitialized.
0	CEN	<b>CEN:</b> Enable Counter 0: Disable the counter; 1: Enable the counter. Note: External Clock, Gated Mode and Encoder Mode will not work until the CEN bit is set in software. Trigger mode can automatically set the CEN bit in hardware. In single pulse mode, CEN is automatically cleared when an update event occurs.

### 15.5.2 TIM10/11/13/14 Interrupt Enable Register (TIMx\_DIER)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														CC1IE	UIE
														rw	rw
Bit	notation	clarification													
15:2	Reserved	Reserved, always reads 0.													
1	CC1IE	<b>CC1IE:</b> Allow Capture/Compare1 interrupt enable 0: Capture/Compare 1 interrupt disabled; 1: Capture/Compare 1 interrupt allowed.													
0	UIE	<b>UIE:</b> update interrupt enable 0: Disable update interrupt; 1: Allow updates to be interrupted.													

### 15.5.3 TIM10/11/13/14 Status Register (TIMx\_SR)

Offset address: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CC1OF	Reserved						CC1IF	UIF	
						rcw0							rcw0	rcw0	

Bit	notation	clarification
15:10	Reserved	Reserved, always reads 0.
9	CC1OF	<b>CC1OF:</b> Capture/Compare1 overcapture flag This flag can be set '1' by hardware only if the corresponding channel is configured for input capture. Writing '0' clears this bit. 0: No duplicate captures are generated; 1: The state of CC1IF is already '1' when the counter value is captured in the TIMx_CCR1 register.
8:2	Reserved	Reserved, always reads 0.
1	CC1IF	<b>CC1IF:</b> Capture/Compare1 interrupt flag If channel CC1 is configured for output mode: This bit is set '1' by hardware when the counter value matches the comparison value, except in centrosymmetric mode (refer to the CMS bit in the TIMx_CR1 register). It is cleared '0' by software. 0: No match occurred; 1: The value of TIMx_CNT matches the value of TIMx_CCR1. If channel CC1 is configured for input mode: This bit is set '1' by hardware when a capture event occurs, and it is cleared '0' by software or by reading TIMx_CCR1. 0: No input capture is generated; 1: The counter value has been captured (copied) to TIMx_CCR1 (an edge of the same polarity as selected is detected on IC1).
0	UIF	<b>UIF:</b> Update interrupt flag (Update interrupt flag) This bit is set '1' by hardware when an update event is generated. It is cleared '0' by software. 0: No update event is generated; 1: Update interrupt wait response. This bit is set '1' by hardware when the register is updated: -If UDIS=0 and URS=0 in the TIMx_CR1 register, an update event is generated when UG=1 in the TIMx_EGR register (software re-initializes the counter CNT); -If UDIS=0 and URS=0 in the TIMx_CR1 register, an update event is generated when the counter CNT is reinitialized by the trigger event. (Refer to the description of the synchronization control register)

### 15.5.4 TIM10/11/13/14 Event Generation Register (TIMx\_EGR)

Offset address: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														CC1G	UG
														w	w
Bit	notation	clarification													
15:2	Reserved	Reserved, always reads 0.													
1	CC1G	<b>CC1G:</b> Capture/compare1 generation This bit is set '1' by software to generate a capture/compare event and is automatically cleared '0' by hardware.													



		0: No action; 1: Generate a capture/compare event on channel CC1: If channel CC1 is configured as an output: Set CC1IF=1 to generate the corresponding interrupt if it is turned on. If channel CC1 is configured as an input: The current counter value is captured to the TIMx_CCR1 register; set CC1IF=1 to generate the corresponding interrupt if it is turned on. If CC1IF is already 1, set CC1OF=1.
0	UG	<b>UG:</b> Update generation This bit is set '1' by software and cleared '0' automatically by hardware. 0: No action; 1: Reinitialize the counter and generate an update event. Note that the prescaler counter is also cleared '0' (but the prescaler coefficients remain unchanged). The counter is cleared '0' if in centrosymmetric mode or DIR=0 (counting up), if DIR=1 (counting down) the counter takes the value of TIMx_ARR.

## 15.5.5 TIM10/11/13/14 Capture/Compare Mode Register 1 (TIMx\_CMR1)

Offset address: 0x18

Reset value: 0x0000

The channel can be used as input (capture mode) or output (compare mode), and the direction of the channel is defined by the corresponding CCxS. The other bits of this register function differently in input and output modes. OCxx describes the function of the channel in output mode and ICxx describes the function of the channel in output mode. It is therefore important to note that the same bit functions differently in output mode and input mode.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									OC1M[2:0]		OC1PE	OC1FE	CC1S[1:0]		
Reserved								IC1F[3:0]			IC1PSC[1:0]				
									rw	rw	rw	rw	rw	rw	rw

### Output comparison mode:

Bit	notation	clarification
15:7	Reserved	Reserved, always reads 0.
6:4	OC1M[2:0]	<b>OC1M[2:0]:</b> Output compare1 mode (Output compare1 enable) These 3 bits define the action of the output reference signal OC1REF, which determines the value of OC1. OC1REF is active high, and the active level of OC1 depends on the CC1P bit. 000: Freeze. Comparison between output comparison register TIMx_CCR1 and counter TIMx_CNT does not work for OC1REF; 001: Set channel 1 to active level when matched. Forces OC1REF high when the value of counter TIMx_CNT is the same as capture/compare register 1 (TIMx_CCR1). 010: Set channel 1 to an invalid level when matched. Forces OC1REF low when the value of counter TIMx_CNT is the same as capture/compare register 1 (TIMx_CCR1). 011: Flip. Flips the level of OC1REF when TIMx_CCR1 = TIMx_CNT. 100: Forces an invalid level. Forces OC1REF low. 101: Force to active level. Forces OC1REF high. 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 active ( OC1REF=1). 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. 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	<b>OC1PE:</b> Output compare1 preload enable 0: Disables the preload function of the TIMx_CCR1 register, which can be written to the TIMx_CCR1 register at any time and the newly written value takes effect immediately. 1: Enable the preload function of TIMx_CCR1 register, read/write operation only operates on the preloaded register, the preloaded value of TIMx_CCR1 is transferred to the current register when the update event arrives. Note: Only in single pulse mode (OPM='1' in TIMx_CR1 register), PWM mode can be used without checking the preload register, otherwise its action is uncertain.
2	OC1FE	<b>OC1FE:</b> Output compare1 fast enable This bit is used to speed up the response of the CC output to trigger input events. 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 a valid edge occurs at the input of the flip-flop. 1: The active edge of the input to the flip-flop acts as if a comparison match has

		<p>occurred. Therefore, OC is set to the comparison level independent of the comparison result. The delay between the active edge of the sampling flip-flop and the CC1 output is reduced to 3 clock cycles.</p> <p>This bit only functions when the channel is configured for PWM1 or PWM2 mode.</p>
1:0	CC1S[1:0]	<p><b>CC1S[1:0]:</b> Capture/Compare1 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: CC1 channel is configured as an input and IC1 is mapped on TI2;</p> <p>11: The CC1 channel is configured as an input with IC1 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: CC1S is writable only when the channel is closed (CC1E='0' in the TIMx_CCER register).</p>

#### Input Capture Mode:

Bit	notation	clarification
15:8	Reserved	Reserved, always reads 0.
7:4	IC1F[3:0]	<p><b>IC1F[3:0]:</b> Input capture1 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 records N events and then generates a jump in the output:</p> <p>0000: no filter, sampling at fDTS 1000: sampling frequency fSAMPLING=fDTS/8, N=6</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>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]	<p><b>IC1PSC[1:0]:</b> input/capture1 prescaler (Input capture1 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]	<p><b>CC1S[1:0]:</b> Capture/Compare1 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: CC1 channel is configured as an input and IC1 is mapped on TI2;</p> <p>11: The CC1 channel is configured as an input with IC1 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: CC1S is writable only when the channel is closed (CC1E='0' in the TIMx_CCER register).</p>

### 15.5.6 TIM10/11/13/14 Capture/Compare Enable Register (TIMx\_CCER)

Offset address: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												CC1NP	Reserved	CC1P	CC1E
												rw		rw	rw

Bit	notation	clarification
15:4	Reserved	Reserved, always reads 0.
3	CC1NP	<p><b>CC1NP:</b> Capture/compare 1 complementary output polarity.</p> <p>The CC1 channel is configured as an output:CC1NP must remain clear.</p> <p>The CC1 channel is configured as an input:CC1NP bit is used in conjunction with CC1P to define TI1FP1 polarity (see CC1P description).</p>

2	Reserved	Reserved, always reads 0.
1	CC1P	<b>CC1P:</b> Input/Capture1 output polarity (Capture/Compare1 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: This bit selects whether IC1 or the inverted signal of IC1 is used as the trigger or capture signal. 0: not inverted: capture occurs on the rising edge of IC1; when used as an external trigger, IC1 is not inverted. 1: Inverted: Capture occurs on the falling edge of IC1; when used as an external trigger, IC1 is inverted.
0	CC1E	<b>CC1E:</b> Input/Capture1 output enable (Capture/Compare1 output enable) The CC1 channel is configured as an output: 0: Off - OC1 disables output. 1: Turn on -OC1 signal output to the corresponding output pin. The CC1 channel is configured as an input: This bit determines whether the counter value can be captured into the TIMx_CCR1 register. 0: Capture disabled; 1: Capture enable.

Table82 Output control bits for standard OCx channels

CCxE bit	OCx Output Status
0	Disable output (OCx=0, OCx_EN=0)
1	OCx = OCxREF + polarity, OCx_EN = 1

Notes: The status of the external I/O pins connected to the standard OCx channel depends on the OCx channel status and the GPIO and AFIO registers.

### 15.5.7 TIM10/11/13/14 Counters (TIMx\_CNT)

Offset address: 0x24

Reset value: 0x0000

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

### 15.5.8 TIM10/11/13/14 Prescaler (TIMx\_PSC)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC [15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit	notation	clarification													
15:0	PSC [15:0]	<b>PSC[15:0]:</b> Prescaler value (Prescaler value) The clock frequency of the counter, CK_CNT, is equal to fCK_PSC/(PSC[15:0]+1). The PSC contains the value loaded into the current prescaler register when the update event is generated.													

### 15.5.9 TIM10/11/13/14 Automatic Reload Register (TIMx\_ARR)

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	notation	clarification													
15:0	ARR[15:0]	<b>ARR[15:0]:</b> Auto reload value The ARR contains the values that will be transferred to the actual Auto-Reload Register. Refer to for detailsthe15.3.1 section : updates and actions related to the ARR. The counter does not work when the value of Auto-Reload is empty.													

## 15.5.10 TIM10/11/13/14 Capture/Compare Register 1 (TIMx\_CCR1)

Offset address: 0x34

Reset value: 0x0000

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

Bit	notation	clarification
15:0	CCR1[15:0]	<p><b>CCR1[15:0]:</b>Capture/Compare1 value (Capture/Compare1 value)</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).</p> <p>If the preload feature 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: CCR1 contains the counter value transmitted by the last input capture 1 event (IC1).</p>

## 15.5.11 TIM10/11/13/14 Registers

The following table maps all registers of the TIM10/11/13/14 into a 16-bit addressable (addressed) space.

Table83 TIM10/11/13/14 Registers and Reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
000h	TIMx_CR1	Reserved																						CKD [1:0]		ARPE	Reserved			OPM	URS	UDIS	CEN			
	Reset value																							0	0									0		
008h	Reserved																																			
00Ch	TIMx_DIER	Reserved																														CC1IE	UIE			
	Reset value																																	0	0	
010h	TIMx_SR	Reserved																						CC10F	Reserved										CC1IF	UIF
	Reset value																																			
014h	TIMx_EGR	Reserved																														CC1G	UG			
	Reset value																																	0	0	
018h	TIMx_CMR1 Output Compare Mode	Reserved																							OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]						
	Reset value																															0	0	0		
	TIMx_CMR1 Input Capture Mode	Reserved																							IC1F [3:0]			IC1PSC [1:0]	CC1S [1:0]							
	Reset value																														0	0	0	0		
01Ch	Reserved																																			

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
020h	TIMx_CCER	Reserved																										CC1NP	Reserved	CC1P	CC1E		
	Reset value																											0	0	0			
024h	TIMx_CNT	Reserved																CNT [15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
028h	TIMx_PSC	Reserved																PSC [15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
02Ch	TIMx_ARR	Reserved																ARR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
030h	Reserved																																
034h	TIMx_CCR1	Reserved																CCR1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
038h~04Ch	Reserved																																

See Table 1 for register start addresses.

## 16 Basic Timer (TIM6 and TIM7)

### 16.1 Introduction to TIM6 and TIM7

The basic timers TIM6 and TIM7 each contain a 16-bit auto-load counter driven by their respective programmable prescalers. They can be used as general-purpose timers to provide time references and, in particular, to clock digital-to-analog converters (DACs). In fact, they are directly connected to the DAC inside the chip and directly drive the DAC via the trigger outputs. These two timers are independent of each other and do not share any resources.

### 16.2 Key Features of TIM6 and TIM7

The main functions of the TIM6 and TIM7 timers include:

- 16-bit Auto-Reload Accumulation Counter
- 16-bit programmable (real-time modifiable) prescaler for dividing the incoming clock by any value between 1 and ~ 65536
- Synchronization circuit for triggering DAC
- Generate interrupt request on update event (counter overflow)

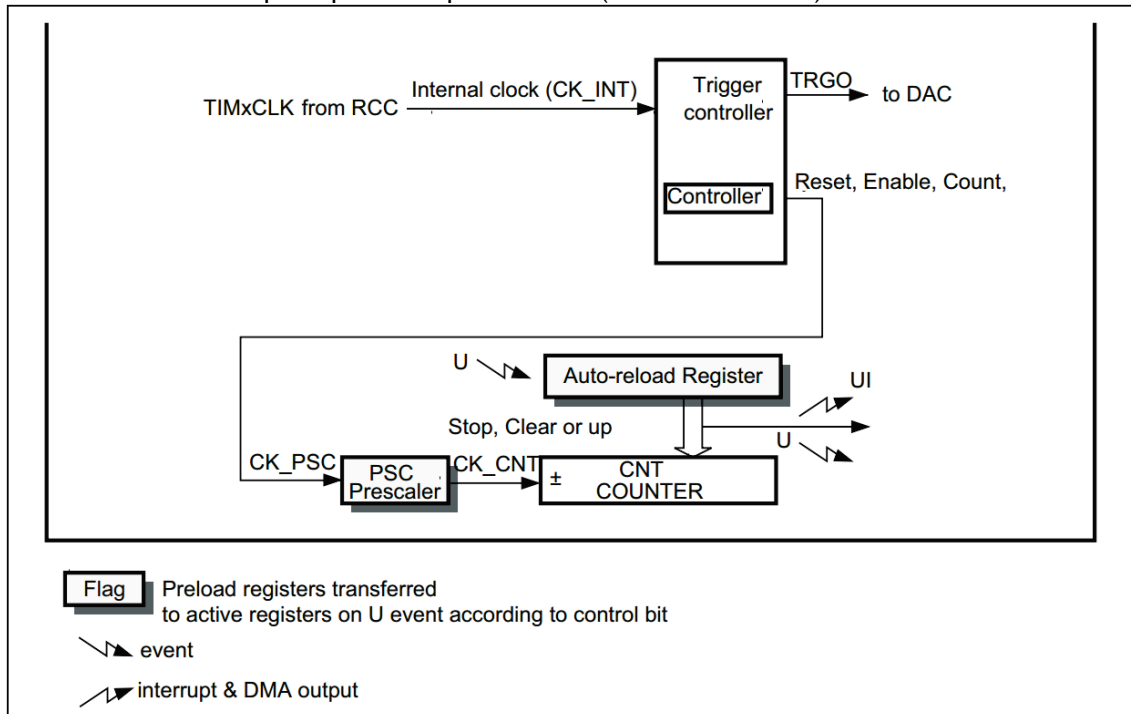


Figure 165 Basic Timer Block Diagram

### 16.3 TIM6 and TIM7 Functions

#### 16.3.1 Time Base Unit (In Computing)

The main part of this programmable timer is a 16-bit accumulator counter with auto-reload, and the clock for the counter is obtained through a prescaler.

The software can read and write counters, auto-reload registers, and prescaler registers, and can operate them even when the counters are running. The time base unit contains:

- Counter Register (TIMx\_CNT)
- Prescaler Register (TIMx\_PSC)
- Automatic Reload Register (TIMx\_ARR)

The auto-reload registers are preloaded, and each time an auto-reload register is read or written, it is actually realized by reading or writing the preload register. Depending on the auto-reload preload enable bit (ARPE) in the TIMx\_CR1 register, writes to the preload register can be transferred to its shadow register either immediately or at each update event. When the UDIS bit in the TIMx\_CR1 register is '0', an update event is issued by the hardware whenever the counter reaches an overflow value; software can also generate update events; details on the generation of update events follow.

The counter is driven by the prescaler output CK\_CNT, and setting the counter enable bit (CEN) in the TIMx\_CR1 register enables the counter to count.

*Notes: The actual setup counter enable signal CNT\_EN lags one clock cycle relative to CEN.*

### Prescaler

The prescaler can divide the counter clock by any value with a coefficient between 1 and 65536. It realizes the frequency division by counting in a 16-bit register (TIMx\_PSC). Because the TIMx\_PSC control register is buffered and its value can be changed during operation, the new prescaler value will take effect at the next update event.

The following two figures show examples of changing the prescaler coefficients during operation.

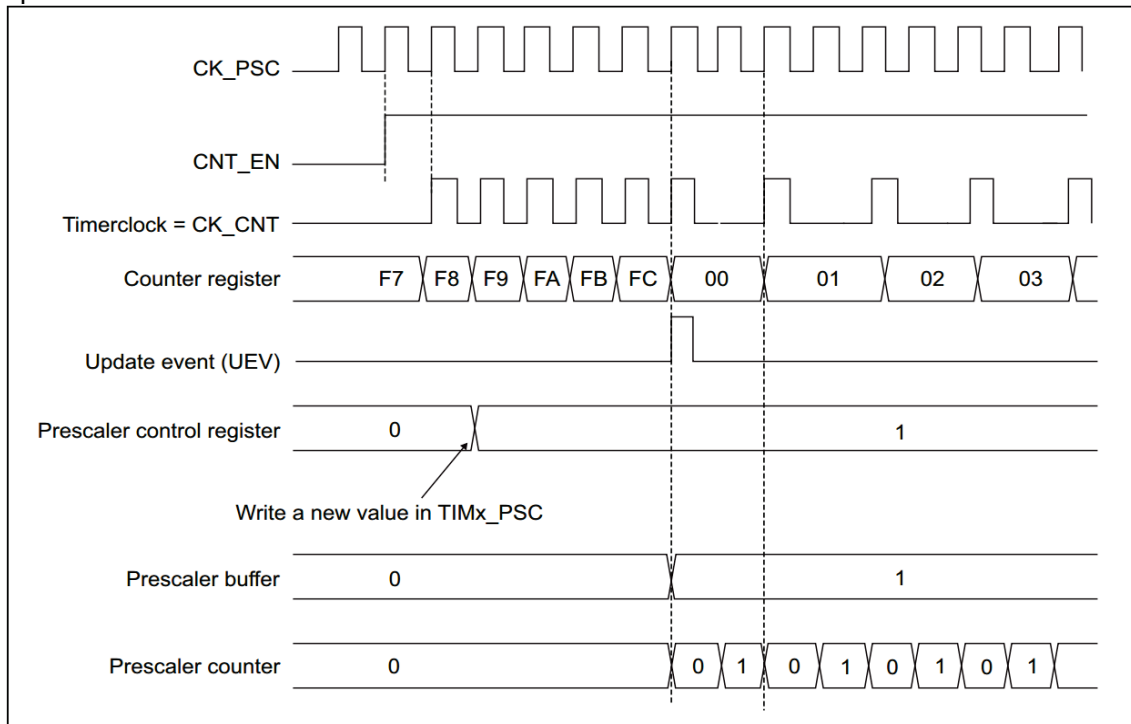


Figure166 Timing diagram of the counter with prescaler factor changed from 1 to 2

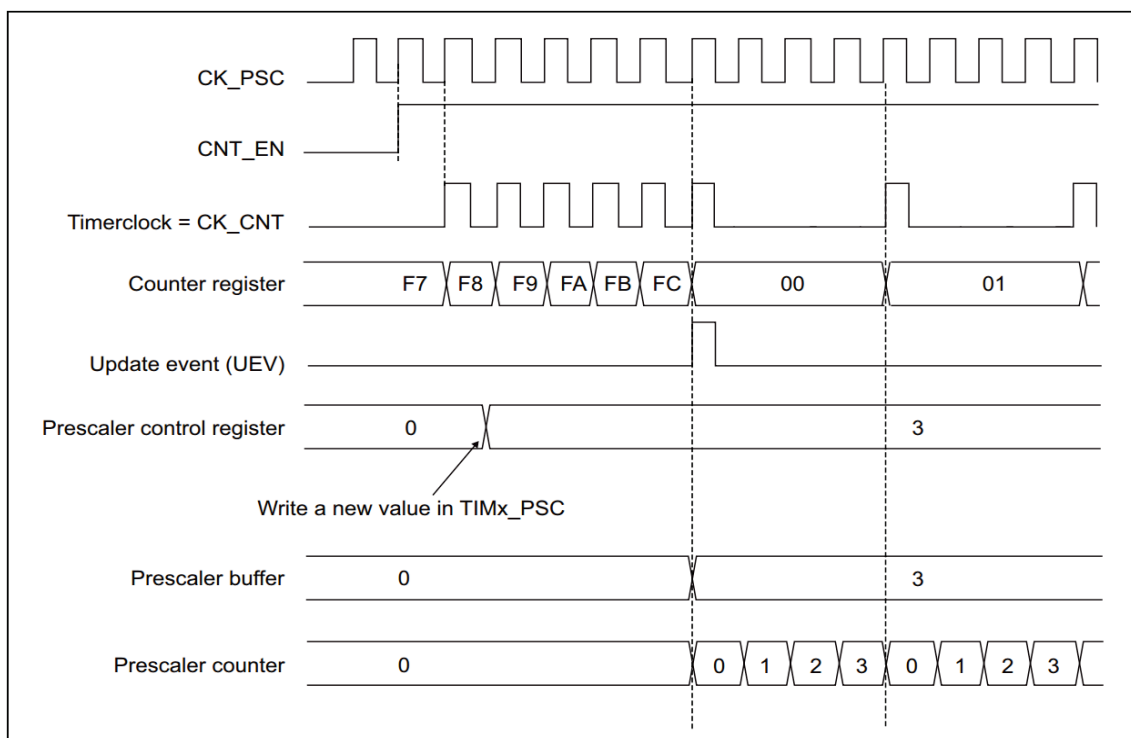


Figure167 Timing diagram of the counter with prescaler coefficient changed from 1 to 4

## 16.3.2 Counting Mode

The counter accumulates counts from 0 to the auto-reload value (TIMx\_ARR register), then starts counting again from 0 and generates a counter overflow event.

An update event can be generated each time the counter overflows; setting the UG bit of the TIMx\_EGR register (in software or using a slave mode controller) can also generate an update event.

Setting the UDIS bit in TIMx\_CR1 disables the generation of UEV events, which prevents the shadow registers from being changed when the preload registers are written. Until the UDIS bit is cleared to '0', no more update events will be generated, but the counter and prescaler will still start counting from 0 again when an update event should be generated (but the prescaler factor remains unchanged). Alternatively, if URS (select update request) in the TIMx\_CR1 register is set, setting the UG bit generates an update event UEV once, but does not set the UIF flag (i.e. no interrupt).

When an update event occurs, all registers are updated and (according to the URS bit) the update flag (UIF bit of the TIMx\_SR register) is set:

- Transmit the preload value (contents of the TIMx\_PSC register) to the prescaler's buffer.
- The auto-reload shadow register is updated to the preloaded value (TIMx\_ARR).

Below are some graphical examples of counter operation at different clock frequencies with TIMx\_ARR=0x36.

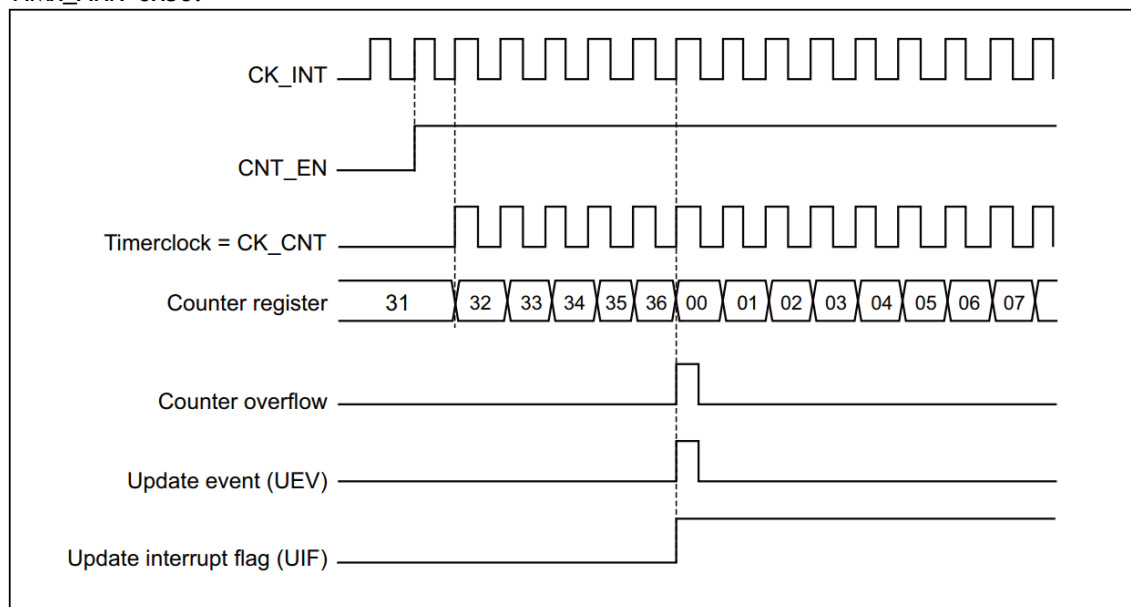


Figure168 Timing diagram of the counter with internal clock division factor of 1

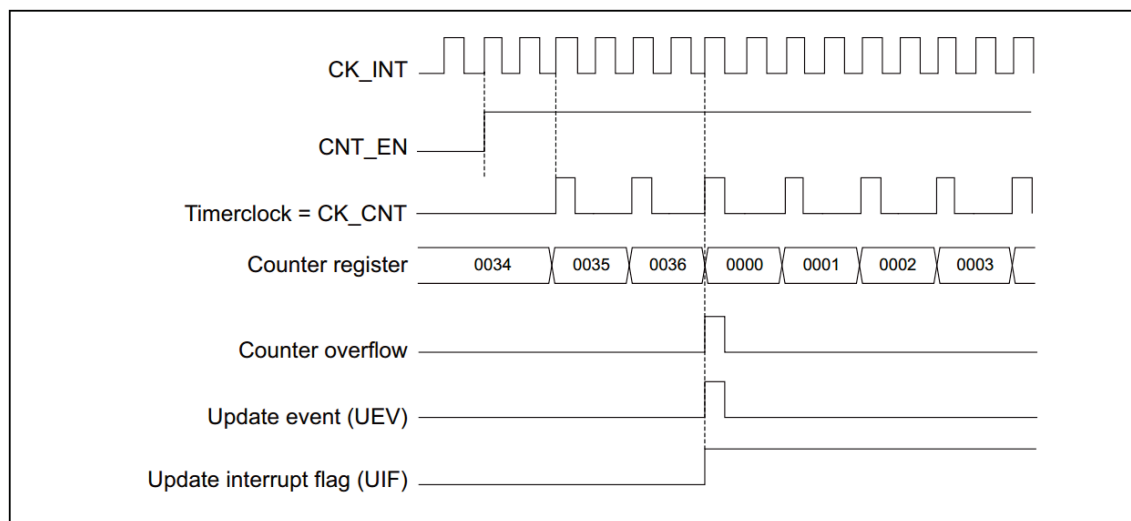


Figure169 Timing diagram of the counter with internal clock division factor of 2



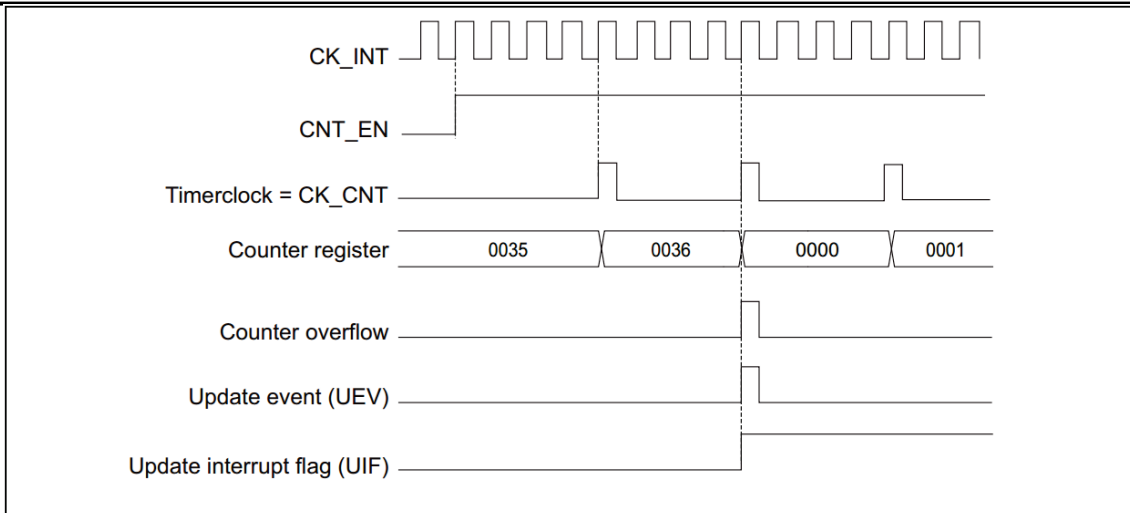


Figure170 Timing diagram of the counter with internal clock division factor of 4

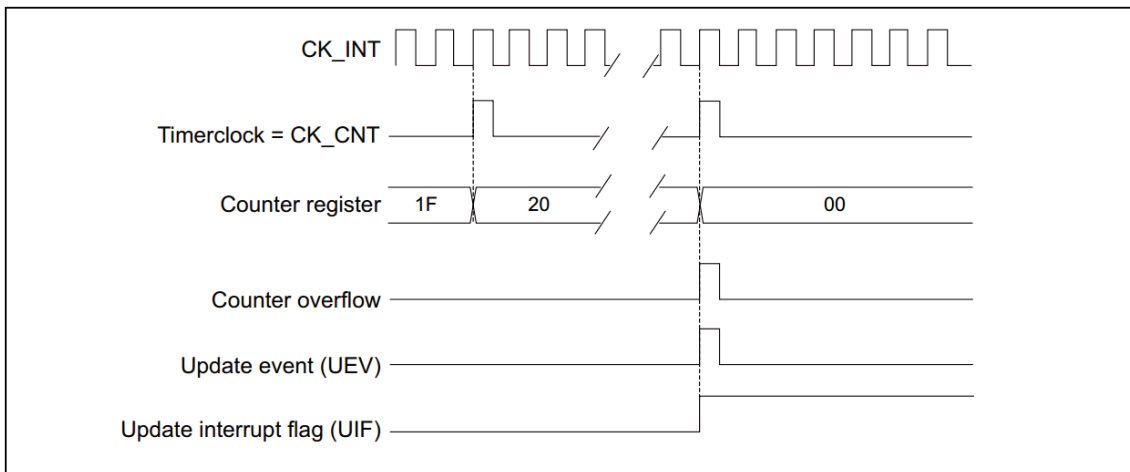


Figure171 Timing diagram of the counter with internal clock division factor N

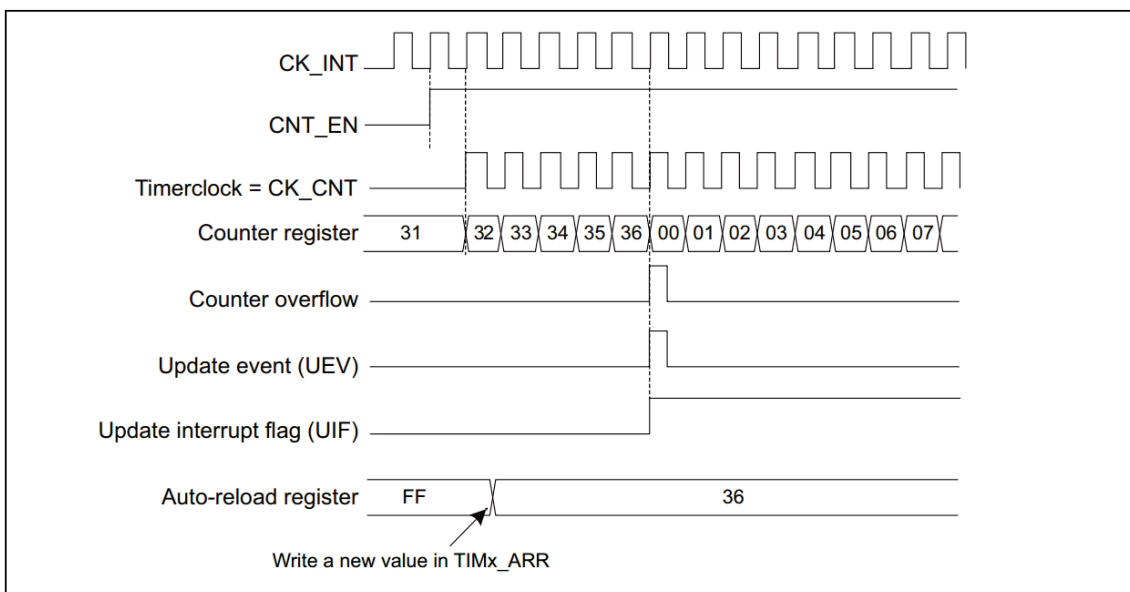


Figure172 Counter Timing Diagram, Update Event when ARPE=0 (TIMx\_ARR not preloaded)

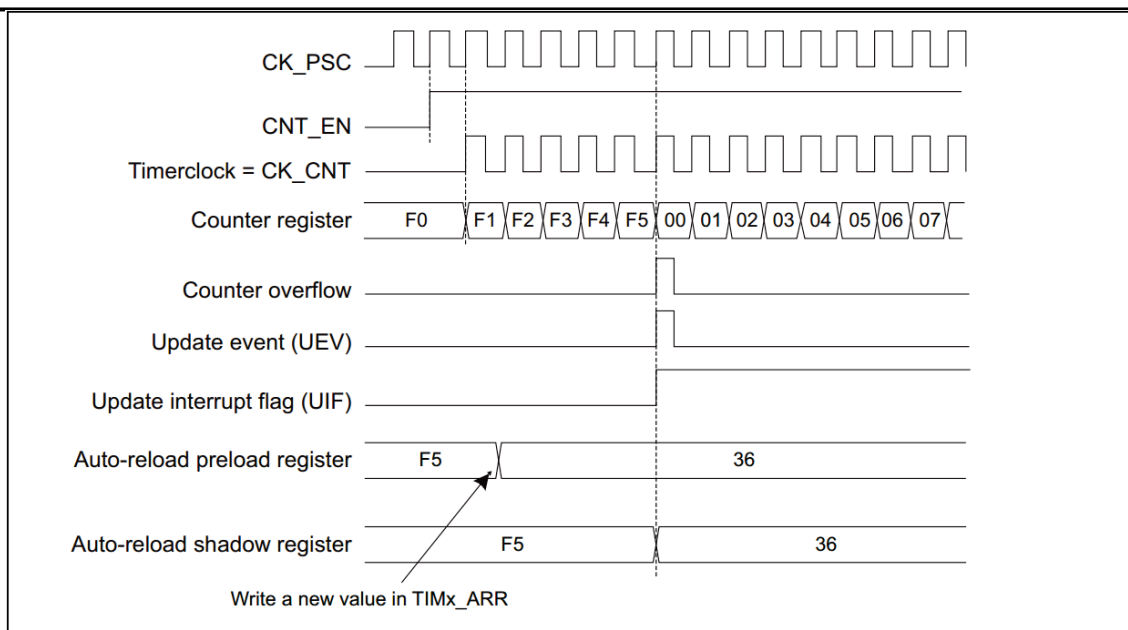


Figure173 Counter Timing Diagram, Update Event when ARPE=1 (Preloaded TIMx\_ARR)

### 16.3.3 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, and (except for the UG bit which is cleared automatically) they can only be changed by software. Once the CEN bit is set to '1', the internal clock supplies the clock to the prescaler.

The following figure illustrates the operation of the control circuit and the up counter in the normal mode without a prescaler.

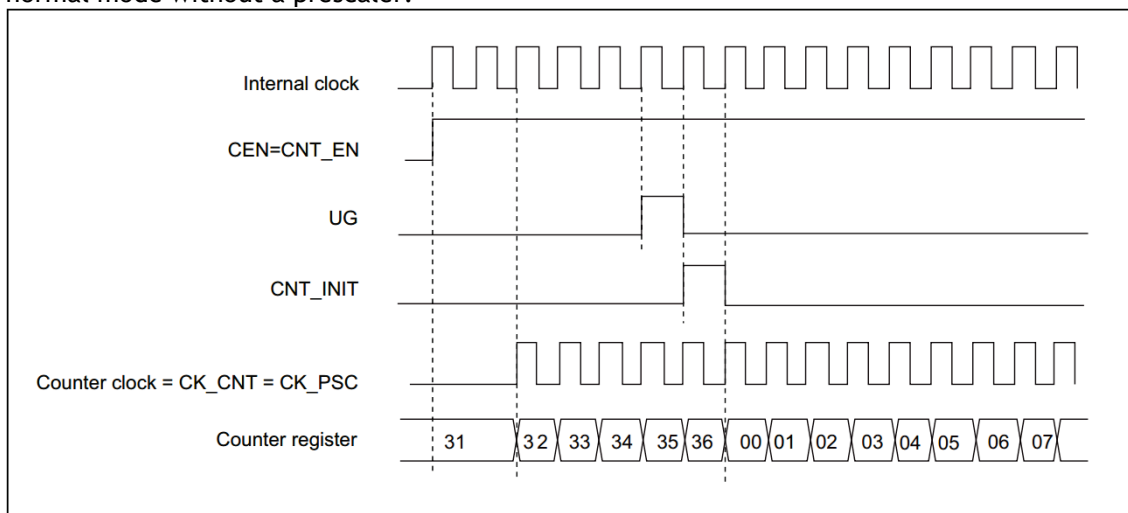


Figure174 Normal mode timing diagram with internal clock divider factor of 1

### 16.3.4 Debug Mode

When the microcontroller enters debug mode (Cortex-M3 core stop), the TIMx counter either continues counting or stops working depending on the setting of the configuration bit DBG\_TIMx\_STOP in the DBG module. See Section 31.16.2 for details.

## 16.4 TIM6 and TIM7 Registers

Refer to Section 1 for abbreviations used in register descriptions.

These peripheral registers can be operated in half-word (16-bit) or word (32-bit) format.

## 16.4.1 TIM6 and TIM7 Control Register 1 (TIMx\_CR1)

Offset address: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								ARPE	Reserved			OPM	URS	UDIS	CEN
								rw				rw	rw	rw	rw

Bit	notation	clarification
15:8	Reserved	Reserved, always reads 0.
7	ARPE	<b>ARPE:</b> auto-reload preload enable 0: TIMx_ARR register is not buffered 1: TIMx_ARR register has buffered
6:4	Reserved	Reserved, always reads 0.
3	OPM	<b>OPM:</b> One-pulse mode 0: counter does not stop when an update event occurs 1: The counter stops counting (clears the CEN bit) when the next update event occurs.
2	URS	<b>URS:</b> Update request source This bit is set and cleared by software to select the request source for UEV events. 0: If interrupts are enabled, either of the following events can generate an update interrupt request: -Counter overflow or underflow -Setting the UG bit -Through updates generated from the mode controller 1: If interrupts are enabled, only counter overflow or underflow can generate an update interrupt request.
1	UDIS	<b>UDIS:</b> Update disable This bit is set and cleared by software to enable or disable the generation of UEV events. 0: UEV enable. Update events (UEV) can be generated by the following events: -Counter overflow or underflow -Setting the UG bit -Through updates generated from the mode controller After generating an update event, the buffered registers are loaded with preloaded values. 1: Disable UEV. no update event (UEV) is generated and the shadow register maintains its contents (ARR, PSC). However, if the UG bit is set or a hardware reset is generated from the mode controller, the counters and prescalers are reinitialized.
0	CEN	<b>CEN:</b> Counter enable 0: Counter off 1: Enable counter Note: Gated mode is only valid when the CEN bit has been set by software, while triggered mode can automatically have the CEN bit set by hardware. In single pulse mode, CEN is automatically cleared when an update event is generated.

## 16.4.2 TIM6 and TIM7 Control Register 2 (TIMx\_CR2)

Offset address: 0x04

Reset value: 0x0000

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

Bit	notation	clarification
15:7	Reserved	Reserved, always reads 0.
6:4	MMS	<b>MMS:</b> Master mode selection These bits are used to select the synchronization message (TRGO) to be sent to the slave timer in master mode in the following combinations: 000: Reset-Uses the UG bit of the TIMx_EGR register as the trigger output (TRGO). If the trigger input generates a reset (from a mode controller configured for reset mode), the signal on TRGO is delayed relative to the actual reset signal. 001: The enable-counter enable signal CNT_EN is used as a trigger output (TRGO). It can be used to start multiple timers at the same moment, or to control the timing of enabling a slave timer. The counter enable signal is generated by a 'logical or' of the CEN control bit and the trigger input when configured for gated mode. When the counter enable signal is controlled by a trigger input, there will be some delay on the TRGO output unless master/slave mode is selected (see the MSM bit of the TIMx_SMCR register). 010: Update-Update events are used as trigger outputs (TRGO). For example a master timer can be used as a prescaler for a slave timer.

3:0	Reserved	Reserved, always reads 0.
-----	----------	---------------------------

### 16.4.3 TIM6 and TIM7 DMA/Interrupt Enable Register (TIMx\_DIER)

Offset address: 0x0C

Reset value: 0x0000

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

Bit	notation	clarification
15:9	Reserved	Reserved, always reads 0.
8	UDE	UDE: Update DMA request enable 0: Disable update of DMA requests 1: Enable update DMA request
7:1	Reserved	Reserved, always reads 0.
0	UIE	UIE: Update interrupt enable 0: Disable update interrupt 1: Enable update interrupt

### 16.4.4 TIM6 and TIM7 Status Registers (TIMx\_SR)

Offset address: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															UIF
															rc_w0

Bit	notation	clarification
15:1	Reserved	Reserved, always reads 0.
0	UIF	UIF: Update interrupt flag (Update interrupt flag) Hardware sets this bit when updating the interrupt and it is cleared by software. 0: No updates were generated. 1: An update interrupt is generated. This bit is set by hardware in the following cases: -The counter generates an overflow or underflow and UDIS = 0 in TIMx_CR1; -If URS=0 and UDIS=0 in TIMx_CR1, when the counter CNT is reinitialized using the UG bit of the TIMx_EGR register.

### 16.4.5 TIM6 and TIM7 Event Generation Registers (TIMx\_EGR)

Offset address: 0x14

Reset value: 0x0000

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

Bit	notation	clarification
15:1	Reserved	Reserved, always reads 0.
0	UG	UG: Update generation This bit is set by software and cleared automatically by hardware. 0: No effect 1: Reinitialize the timer's counter and generate an update to the registers. Note: The prescaler is also cleared (but the prescaler coefficients remain unchanged).

## 16.4.6 TIM6 and TIM7 Counters (TIMx\_CNT)

Offset address: 0x24

Reset value: 0x0000

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

## 16.4.7 TIM6 and TIM7 Prescalers (TIMx\_PSC)

Offset address: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC [15:0]															
rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW
Bit	notation	clarification													
15:0	PSC [15:0]	<b>PSC[15:0]: Prescaler value</b> The clock frequency of the counter, CK_CNT, is equal to $f_{CK\_PSC}/(PSC[15:0]+1)$ . At each update event, the value of the PSC is transferred to the actual prescaler register.													

## 16.4.8 TIM6 and TIM7 Auto-Reload Registers (TIMx\_ARR)

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	notation	clarification													
15:1	ARR[15:0]	<b>ARR[15:0]: Prescaler value</b> The value of the ARR will be transferred to the actual auto-reload register. See 14.3.1 for more details on the update and role of the ARR. If the auto-reload value is 0, the counter stops.													



## 17 Real Time Clock (RTC)

### 17.1 Introduction to RTC

The real-time clock is a stand-alone timer. The RTC module has a set of continuously counting counters that, when configured accordingly to the software, provide the functionality of a clock calendar. Modifying the counter values resets the current time and date of the system.

The RTC module and the clock configuration system (RCC\_BDCR register) are in the fallback area, i.e., the RTC settings and timing remain unchanged after a system reset or wakeup from standby mode.

After a system reset, access to the backup registers and RTC is disabled, this is to prevent accidental write operations to the backup area (BKP). Performing the following operation will enable access to the backup registers and RTC:

- Set the PWREN and BKPEN bits of register RCC\_APB1ENR to enable the power and backup interface clocks
- Set the DBP bit of register PWR\_CR to enable access to the back-up registers and RTC.

### 17.2 Main Characteristics

- Programmable prescaling factor: crossover factor up to 220.
- 32-bit programmable counter for longer time periods.
- 2 separated clocks: PCLK1 for the APB1 interface and the RTC clock (the RTC clock must be less than a quarter or more of the PCLK1 clock frequency).
- The following three clock sources for the RTC can be selected:
  - The HSE clock is divided by 128;
  - LSE oscillator clock;
  - LSI oscillator clock (see 6.2.8 section RTC Clock for details).
- 2 separate reset types:
  - The APB1 interface is reset by the system;
  - The RTC cores (prescalers, alarms, counters, and dividers) can only be reset by the fallback domain (see section 6.1.3 for details).
- 3 specialized maskable interrupts:
  - Alarm Clock Interrupt to generate a software programmable alarm clock interrupt.
  - Seconds interrupt, used to generate a programmable periodic interrupt signal (up to 1 second).
  - Overflow interrupt indicating that the internal programmable counter has overflowed and slewed to a 0 state.

### 17.3 Functional Description

#### 17.3.1 Summarize

The RTC consists of two main parts (see figure below). The first part (APB1 interface) is used to connect to the APB1 bus. This unit also contains a set of 16-bit registers that can be read and written to via the APB1 bus (see section 17.4). The APB1 interface is driven by the APB1 bus clock and is used to interface with the APB1 bus.

The other part (RTC core) consists of a set of programmable counters divided into two main modules. The first module is the RTC's prescaler module, which is programmable to generate the RTC time reference TR\_CLK of up to 1 s. The RTC's prescaler module contains a 20-bit programmable frequency divider (RTC prescaler). If the corresponding allowable bit is set in the RTC\_CR register, it is programmed in every

The RTC generates an interrupt (seconds interrupt) during the TR\_CLK cycle. The second module is a 32-bit programmable counter that can be initialized to the current system time. The system time is accumulated by TR\_CLK cycles and compared to the programmable time stored in the RTC\_ALR register, and an alarm interrupt is generated when the comparison matches if the appropriate allow bit is set in the RTC\_CR control register.

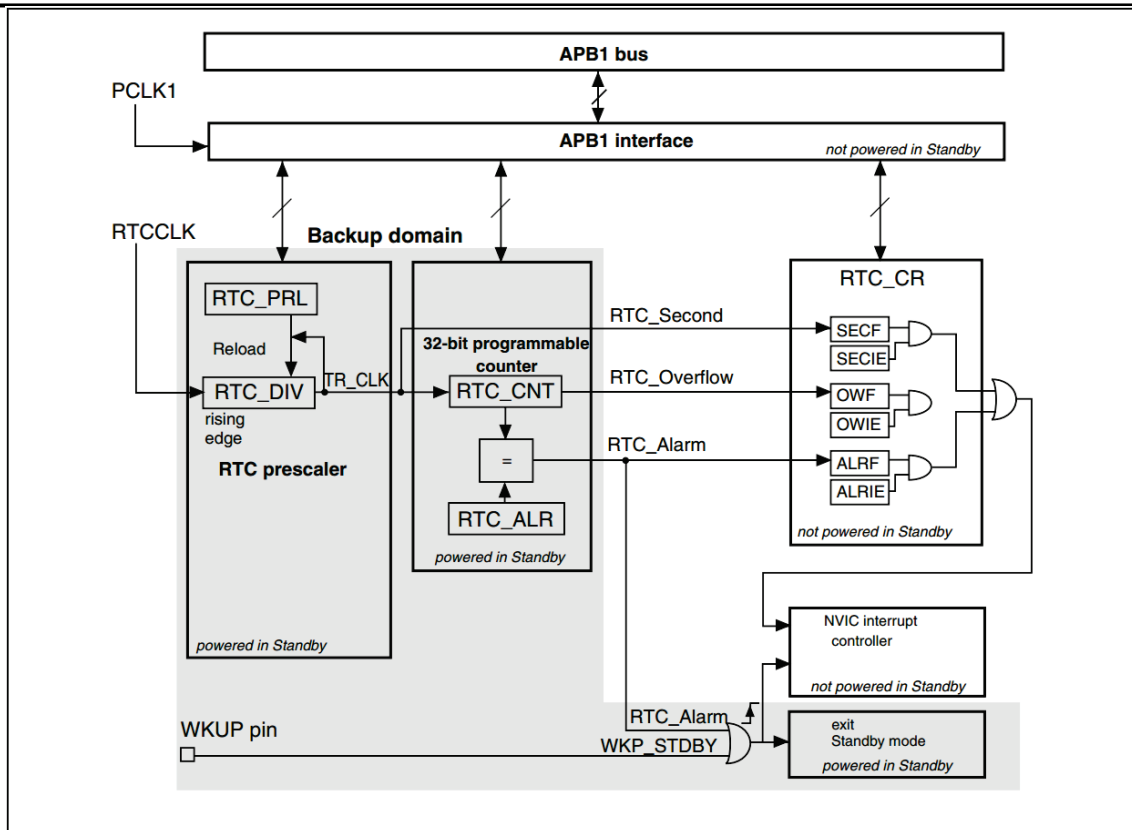


Figure175 Simplified RTC Block Diagram

### 17.3.2 Reset Process

All system registers are reset asynchronously by a system reset or power reset, except for the RTC\_PRL, RTC\_ALR, RTC\_CNT, and RTC\_DIV registers. The RTC\_PRL, RTC\_ALR, RTC\_CNT, and RTC\_DIV registers can only be reset by the backup domain reset signal, as described in Section 6.1.3.

### 17.3.3 Read RTC Registers

The RTC core is completely independent of the RTCAPB1 interface. Software accesses the RTC's prescaler values, counter values, and alarm values through the APB1 interface. However, the associated readable registers are updated only on the rising edge of the RTC clock that is resynchronized with the RTCAPB1 clock. the same is true for the RTC flags.

This means that if the APB1 interface is ever turned off and the read operation is just after the APB1 has been turned back on, the RTC register value read from the APB1 may be corrupted (usually read to 0) before the first internal register update. The following situations are capable of this scenario:

- A system reset or power reset occurs
- The system has just woken up from standby mode (see Section 4.3 : Low Power Mode).
- The system has just woken up from shutdown mode (see Section 4.3 : Low Power Mode).

In all of the above cases, the RTC core remains operational when the APB1 interface is disabled (reset, no clock, or power failure).

Therefore, if the RTC's APB1 interface is ever disabled while reading the RTC register, software must first wait for the RSF bit (Register Synchronization Flag) in the RTC\_CRL register to be set '1' by hardware.

*Notes: The APB1 interface of the RTC is not affected by low power modes such as WFI and WFE.*



### 17.3.4 Configure RTC Registers

The CNF bit in the RTC\_CRL register must be set to put the RTC into configuration mode before the RTC\_PRL, RTC\_CNT, and RTC\_ALR registers can be written.

In addition, write operations to any of the RTC registers must be performed after the previous write operation has been completed. You can determine whether an RTC register is in update by querying the RTOFF status bit in the RTC\_CR register. The RTC registers can be written only when the RTOFF status bit is '1'.

Configuration process:

1. Query the RTOFF bit until the value of RTOFF changes to '1'
2. Set CNF value to 1 to enter configuration mode
3. Write operations to one or more RTC registers
4. Clear the CNF flag bit to exit configuration mode
5. Query RTOFF until the RTOFF bit changes to '1' to confirm that the write operation is complete.

The write operation can be performed only when the CNF flag bit is cleared, a process that takes at least 3 RTCCLK cycles.

### 17.3.5 RTC Flag Setting

Set the RTC seconds flag (SECF) before changing the RTC counter on every RTC core clock cycle. Set the RTC Overflow Flag (OWF) on the last RTC clock cycle before the counter reaches 0x0000. Set RTC\_Alarm and the RTC Alarm Flag (ALRF) during the RTC clock cycle before the counter value reaches the value of the Alarm Register plus 1 (RTC\_ALR+1). Write operations to the RTC alarm clock must be synchronized with the RTC seconds flag using one of the following procedures:

- Use the RTC alarm interrupt and modify the RTC alarm and/or RTC counter in the interrupt handler.
- Wait for the SECF bit in the RTC control register to be set before changing the RTC alarm and/or RTC counter.

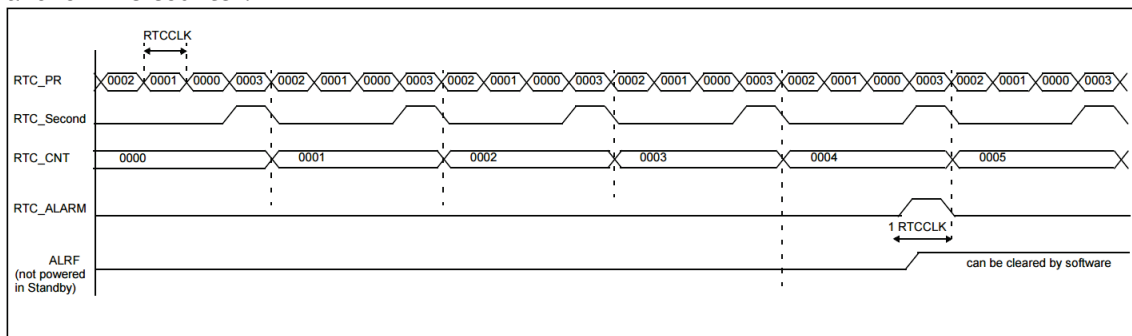


Figure 176 Example RTC Seconds and Alarm Clock Waveform Graph, PR=0003, ALARM=00004

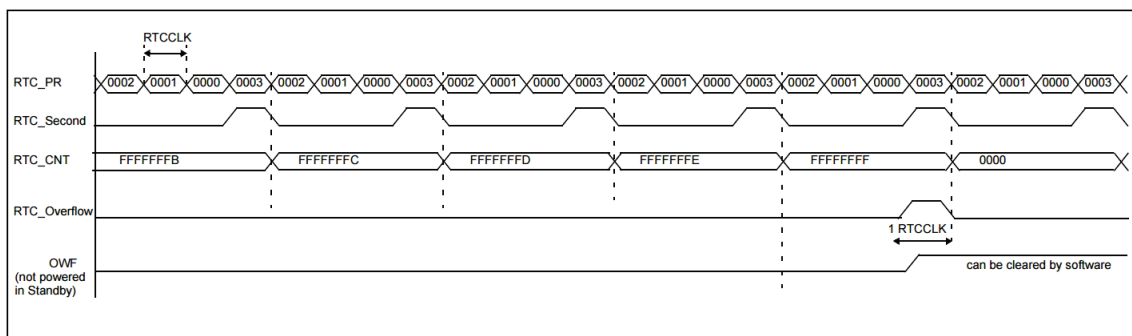


Figure 177 RTC Overflow Waveform Diagram Example, PR=0003

## 17.4 RTC Register Description

For abbreviations in register descriptions, refer to Section 1.

These peripheral registers can be operated in half-word (16-bit) or word (32-bit) format.

## 17.4.1 RTC Control Register High (RTC\_CRH)

Offset address: 0x00

Reset value: 0x0000

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

Bit	notation	clarification
15:3	Reserved	Reserved, forced to 0 by hardware.
2	OWIE	<b>OWIE:</b> Overflow interrupt enable bit 0: Mask (do not allow) overflow interrupts 1: Allow overflow interrupts
1	ALRIE	<b>ALRIE:</b> Allow alarm interrupt (Alarm interrupt enable) 0: Masked (not allowed) alarm interruption 1: Allow the alarm to be interrupted
0	SECIE	<b>SECIE:</b> second interrupt enable 0: Masked (not allowed) second interruption 1: Allow second interruptions

These bits are used to mask interrupt requests. Note: All interrupts are masked after a system reset, so a write to the RTC register can be used to ensure that there are no pending interrupt requests after initialization. The RTC\_CRH register cannot be written to while the peripheral is completing a previous write operation (flag bit RTOFF=0).

The RTC function is controlled by this control register. The write operation of some bits must be accomplished through a special configuration process (see section0 ).

## 17.4.2 RTC Control Register Low (RTC\_CRL)

Offset address: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										RTOFF	CNF	RSF	OWF	ALRF	SECF
										r	rw	rc_w0	rc_w0	rc_w0	rc_w0

Bit	notation	clarification
15:6	Reserved	Reserved, forced to 0 by hardware.
5	RTOFF	<b>RTOFF:</b> RTC operation OFF The RTC module uses this bit to indicate the status of the last operation performed on its registers, indicating whether the operation is complete or not. If this bit is '0', it indicates that no write operation can be performed on any of the RTC registers. This bit is read-only. 0: the last write operation to the RTC register is still in progress; the 1: The last write operation to the RTC register has been completed.
4	CNF	<b>CNF:</b> Configuration flag This bit must be set '1' by software to enter Configuration Mode to allow writing to the RTC_CNT, RTC_ALR, or RTC_PRL registers. A write operation will only be performed if this bit is cleared after being set '1' and re-cleared '0' by software. 0: Exit configuration mode (start updating RTC registers); 1: Enter configuration mode.
3	RSF	<b>RSF:</b> Registers synchronized flag (Registers synchronized flag) This bit is set '1' by hardware whenever the RTC_CNT register and RTC_DIV register are updated or cleared '0' by software. This bit must be cleared '0' by software after an APB1 reset, or after the APB1 clock is stopped. To perform any read operation, the user program must wait for this bit to be set '1' by hardware to ensure that RTC_CNT, RTC_ALR, or RTC_PRL has been synchronized. 0: The registers have not been synchronized; 1: The registers have been synchronized.
2	OWF	<b>OWF:</b> Overflow flag This bit is set '1' by hardware when the 32-bit programmable counter overflows. If OWIE=1 in the RTC_CRH register, an interrupt is generated. This bit can only be cleared '0' by software. Writing a '1' to this bit is not valid. 0: No overflow; 1: 32-bit programmable counter overflow.
1	ALRF	<b>ALRF:</b> Alarm flag When the 32-bit programmable counter reaches a predetermined value set in the RTC_ALR register, this bit is set '1' by hardware. If ALRIE=1 in the RTC_CRH register, an interrupt is generated. This bit can only be cleared '0' by software. Writing a '1' to this bit is not valid. 0: No alarm clock;

		1: There is an alarm clock.
0	SECF	<b>SECF: Second flag</b> When the 32-bit programmable prescaler overflows, this bit is set by hardware to '1' at the same time as the RTC counter is incremented by 1. Therefore, this flag provides a periodic signal (typically 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 a '1' to this bit is not valid. 0: The seconds flag condition does not hold; 1: The second sign condition holds.

The function of the RTC is controlled by this control register. The RTC\_CR register cannot be written while the previous write operation has not been completed (when RTOFF=0, see section0 for details).

*Notes:1 Any flag bits will remain pending until the appropriate RTC\_CR request bit is reset by software, indicating that the requested interrupt has been Accepted.*

- 2 All interrupts are disabled at reset, no pending interrupt requests, and write operations can be performed on the RTC registers.
- 3 When the APB1 clock is not running, the OWF, ALRF, SECF, and RSF bits are not updated.
- 4 The OWF, ALRF, SECF and RSF bits can only be set by hardware and cleared by software.
- 5 If ALRF=1 and ALRIE=1, generation of RTC global interrupts is allowed. If generation of EXTI line 17 interrupts is allowed in the EXTI controller, generation of RTC global interrupts and RTC alarm clock interrupts is allowed.
- 6 If ALRF=1, the generation of an RTC alarm interrupt is allowed if the interrupt mode of EXTI line 17 is set in the EXTI controller;  
If the event mode of EXTI line 17 is set in the EXTI controller, a pulse is generated on this line (no RTC alarm interrupt is generated).

### 17.4.3 RTC Prescaler Load Register (RTC\_PRLH/RTC\_PRL)

The prescaler loading registers are used to hold the cycle count values of the RTC prescaler. They are protected by the RTOFF bit of the RTC\_CR register and write operations are allowed only when the RTOFF value is '1'.

#### RTC Prescaler Load Register High (RTC\_PRLH)

Offset address: 0x08

Write-only (see section0 )

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												PRL [19:16]			
												W	W	W	W

Bit	notation	clarification
15:4	Reserved	Reserved, forced to 0 by hardware.
3:0	PRL [19:16]	<b>PRL[19:16]:</b> RTC prescaler reload value high These bits are used to define the clock frequency of the counter according to the following formula: $f_{TR\_CLK} = f_{RTCCLK} / (PRL[19:0] + 1)$ Note: A value of 0 is not recommended, otherwise the RTC interrupt and flag bits cannot be generated correctly.

#### RTC Prescaler Load Register Low (RTC\_PRL)

Offset address: 0x0C

Write-only (see section0 )

Reset value: 0x8000

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	notation	clarification
15:0	PRL[15:0]	<b>PRL[15:0]:</b> RTC prescaler reload value high These bits are used to define the clock frequency of the counter according to the following formula: $f_{TR\_CLK} = f_{RTCCLK} / (PRL[19:0] + 1)$ Note: A value of 0 is not recommended, otherwise the RTC interrupt and flag bits cannot be generated correctly.

Notes: If the input clock frequency (fRTCCLK) is 32.768 kHz, write 7FFFh to this register for a 1-second signal period

## 17.4.4 RTC Prescaler Remainder Register (RTC\_DIVH/RTC\_DIVL)

The value of the counter in the RTC prescaler is reset to the value of the RTC\_PRL register during each cycle of TR\_CLK. The user can read the RTC\_DIV register to obtain the current value of the prescaler counter without stopping the crossover counter, thus obtaining an accurate time measurement. This register is a read-only register whose value is reloaded by the hardware after the value in the RTC\_PRL or RTC\_CNT register is changed.

### RTC prescaler remainder register high (RTC\_DIVH)

Offset address: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												RTC_DIV[19:16]			
												r	r	r	r

Bit	notation	clarification
15:6	Reserved	Reserved, forced to 0 by hardware.
3:0	RTC_DIV [19:16]	RTC_DIV[19:16]:RTC clock divider high

### RTC prescaler remainder register low (RTC\_DIVL)

Offset address: 0x14

Reset value: 0x8000

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

Bit	notation	clarification
15:6	RTC_DIV [15:0]	RTC_DIV[15:0]:RTC clock divider low

## 17.4.5 RTC Counter Register (RTC\_CNTH/RTC\_CNTL)

The RTC core has a 32-bit programmable counter accessible through two 16-bit registers. The counter counts with reference to the TR\_CLK time base generated by the prescaler. The RTC\_CNT registers are used to hold the counter count values. They are write-protected by bit RTOFF of RTC\_CR, which allows write operations only when the RTOFF value is '1'. Write operations on the high or low registers (RTC\_CNTH or RTC\_CNTL) are able to load directly to the corresponding programmable counter and reload the RTC prescaler. When a read operation is performed, the count value (system time) within the counter is returned directly.

### RTC counter register high (RTC\_CNTH)

Offset address: 0x18

Reset value: 0x0000

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	notation	clarification
15:6	RTC_CNT [31:16]	RTC_CNT[31:16]: RTC counter high The high portion of the current value of the RTC counter can be obtained by reading the RTC_CNTH register. Configuration mode must be entered before (see Writing to this register section).

### RTC Counter Register Low (RTC\_CNTL)

Offset address: 0x1C

Reset value: 0x0000

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	notation	clarification													
15:6	RTC_CNT [15:0]	RTC_CNT[15:0]: RTC counter high The low part of the current value of the RTC counter can be obtained by reading the RTC_CNTH register. Configuration mode must be entered before (see 0writing to this register section ).													

## 17.4.6 RTC Alarm Clock Register (RTC\_ALRH/RTC\_ALRL)

An alarm event is triggered when the value of the programmable counter is equal to the 32-bit value in RTC\_ALR and an RTC alarm interrupt is generated. This register is write-protected by the RTOFF bit in the RTC\_CR register, which allows write operations only when the RTOFF value is '1'.

### RTC alarm clock register high (RTC\_ALRH)

Offset address: 0x20

Write-only (see section0 )

Reset value: 0xFFFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
rtc_alr[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
Bit	notation	clarification													
15:6	RTC_ALR [31:16]	RTC_ALR[31:16]: RTC alarm value high (RTC alarm high) This register is used to hold the high portion of the alarm time written by software. To write to this register, you must first enter configuration mode (see section0 ).													

### RTC alarm clock register low (RTC\_ALRL)

Offset address: 0x24

Write-only (see section0 )

Reset value: 0xFFFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTC_ALR[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
Bit	notation	clarification													
15:0	RTC_ALR [15:0]	RTC_ALR[15:0]: RTC alarm value low (RTC alarm low) This register is used to hold the low portion of the alarm time written by software. To write to this register, you must first enter configuration mode (see section0 ).													

## 17.4.7 RTC Register Image

The RTC registers are 16-bit addressable registers described as follows:

Table85 RTC-Register Image and Reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																						
000h	RTC_CRH	Reserved																																																					
	Reset value																																																						
004h	RTC_CRL	Reserved																																																					
	Reset value																																																						
008h	RTC_PRLH	Reserved																																																					
	Reset value																																																						
00Ch	RTC_PRLH	Reserved																PRL[15:0]																																					
	Reset value																	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0												
010h	RTC_DIVH	Reserved																DIV[31:16]																																					
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0											
014h	RTC_DIVL	Reserved																DIV[15:0]																																					
	Reset value																	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0											
018h	RTC_CNTH	Reserved																CNT [31:16]																																					
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0											
01Ch	RTC_CNTL	Reserved																CNT [15:0]																																					
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0										
020h	RTC_ALRH	Reserved																ALR[31:16]																																					
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1										
024h	RTC_ALRL	Reserved																ALR[15:0]																																					
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1										

SeeTable1 for register start addresses.

---

## 18 Independent Watchdog Dog (IWDG)

### 18.1 Introduction

The W55MH32 has two built-in watchdogs, providing increased security, time accuracy and flexibility of use. The two watchdog devices (Standalone Watchdog and Window Watchdog) can be used to detect and troubleshoot faults caused by software errors; triggering an interrupt (Window Watchdog only) or generating a system reset when the counter reaches a given timeout value.

The Independent Watchdog (IWDG) is driven by a dedicated Low Speed Clock (LSI) that remains active even if the master clock fails. The Window Watchdog is driven by a clock derived from the APB1 clock divider and detects abnormal late or early operation of the application program through a configurable time window.

The IWDG is best suited for applications that require the watchdog to function as a completely independent program outside of the main program and that require less timing accuracy. The WWDG is best suited for applications that require the watchdog to function in a precise timing window. The IWDG is best suited for applications that require the watchdog to function in a precise timing window.

See Chapter 17 for details on window watchdogs.

### 18.2 IWDG Main Properties

- Free-running decrement counter
- Clocking is provided by a separate RC oscillator (can operate in stop and standby modes)
- When the watchdog is activated, a reset is generated when the counter reaches 0x000.

### 18.3 IWDG Functional Description

Figure 178 shows the functional block diagram of the standalone watchdog module.

Writing 0xCCCC to the key register (IWDG\_KR) starts to enable the standalone watchdog; at this point the counter starts counting in decrements from its Reset value of 0xFFFF. When the counter counts to the end 0x000, a reset signal (IWDG\_RESET) is generated.

Whenever 0xAAAA is written in the key register IWDG\_KR, the value in IWDG\_RLR is reloaded into the

counter, thus avoiding the generation of a watchdog reset.

#### 18.3.1 Hardware Watchdog

If the user enables the "Hardware Watchdog" function in the selection byte, the watchdog will start to run automatically after the system power-on reset; if the software does not write the corresponding value to the key register before the end of the counter counting, the system will generate a reset.

#### 18.3.2 Register Access Protection

The IWDG\_PR and IWDG\_RLR registers are write-protected. To modify the values of these two registers, you must first write 0x5555 to the IWDG\_KR register. Writing to this register with a different value will disrupt the order of operations and the register will be re-protected. A reload operation (i.e., writing 0xAAAA) also activates write protection.

The status register indicates whether the prescaler value and decrement counter are being updated.

#### 18.3.3 Debug Mode

When the microcontroller enters debug mode (Cortex-M3 core stop), the IWDG's counters are able to continue working or stop depending on the state of the DBG\_IWDG\_STOP configuration bit in the debug module. See the section on the debug module for more details.

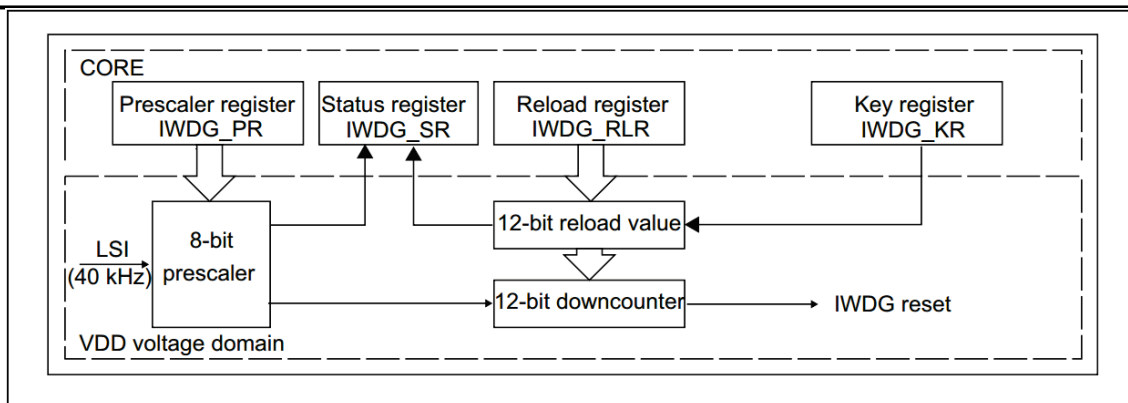


Figure178 Standalone Watchdog Block Diagram

Notes: The watchdog function is in the VDD power supply area, i.e., it still operates properly during shutdown and standby modes.

Table86 Watchdog Timeout (40kHz Input Clock (LSI))

presaring factor	PR[2:0] bits	Minimum time (ms) RL[11:0]=0x000	Maximum time (ms) RL[11:0]=0xFFF
/4	0	0.1	409.6
/8	1	0.2	819.2
/16	2	0.4	1638.4
/32	3	0.8	3276.8
/64	4	1.6	6553.6
/128	5	3.2	13107.2
/256	(6 or 7)	6.4	26214.4

Notes: These times are given in accordance with the 40kHz clock. In reality, the RC frequency inside the MCU will vary between 30kHz and 60kHz.

Furthermore, even if the RC oscillator frequency is exact, the exact timing still depends on the phase difference between the APB interface clock and the RC oscillator clock, so there will always be a complete RC cycle that is uncertain.

Relatively accurate watchdog timeouts can be obtained by calibrating the LSI. See section6.2.5 for more information on LSI calibration.

## 18.4 IWDG Register Description

See Section1 for details on the abbreviations used in the register descriptions.

These peripheral registers can be operated in half-word (16-bit) or word (32-bit) format.

### 18.4.1 Key Register (IWDG\_KR)

Address offset: 0x00

Reset value: 0x0000 0000 (reset in standby mode)

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

Bit	notation	clarification
31:16	Reserved	Reserved, always reads 0.
15:0	KEY[15:0]	KEY[15:0]:Key value (write register only, read value 0x0000)(Key value) Software must write 0xAAAA at regular intervals, otherwise the watchdog will generate a reset when the counter is 0. A write to 0x5555 indicates that access to the IWDG_PR and IWDG_RLR registers is allowed. (See section18.3.2 ) Write 0xCCCC to start the watchdog operation (not subject to this command word if hardware watchdog is selected).





## 18.4.4 Status Register (IWDG\_SR)

Address offset: 0x0C

Reset value: 0x0000 0000 (not reset in standby mode)

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

Bit	notation	clarification
31:2	Reserved	Reserved, always reads 0.
1	RVU	RVU: Watchdog counter reload value update This bit is set '1' by hardware to indicate that a reload value update is in progress. This bit is cleared '0' by hardware when the reload update in the VDD domain is complete (up to 5 40kHz RC cycles). The reload value can only be updated after the RVU bit is cleared '0'.
0	PVU	PVU: Watchdog prescaler value update This bit is set '1' by hardware to indicate that a prescaler update is in progress. This bit is cleared '0' by hardware when the prescaler update in the VDD domain is complete (up to 5 40kHz RC cycles). The prescaler value can only be updated after the PVU bit is cleared '0'.

**Notes:** If more than one reload value or prescaler value is used in an application program, the RVU bit must be cleared before the preload value can be changed again, and the PVU bit must be cleared before the prescaler value can be changed again. However, it is not necessary to wait for the RVU or PVU to reset after the prescaler and/or reload values have been updated, and execution of the following code can continue. (That is, this write operation will continue to be executed to completion even in low power mode.)

## 18.4.5 IWDG Register Image

Table87 IWDG Register Image and Reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h	IWDG_KR	Reserved																KEY[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
004h	IWDG_PR	Reserved																												PR[2:0]			
	Reset value																													0	0	0	
008h	IWDG_RLR	Reserved																RL[11:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
00Ch	IWDG_SR	Reserved																												RVU	PVU		
	Reset value																													0	0		

SeeTable1 for register start addresses.

## 19 Window Watchdog (WWDG)

### 19.1 WWDG Profile

Window watchdogs are commonly used to monitor for software failures that occur when an application program deviates from its normal sequence of operation due to external disturbances or unforeseen logic conditions. Unless the decrement counter value is flushed before the T6 bit becomes 0, the watchdog circuit generates an MCU reset when a preset time period is reached. An MCU reset will also be generated if the 7-bit decrement counter value (in the control register) is flushed before the decrement counter reaches the window register value. This indicates that the decrement counter needs to be refreshed within a limited time window.

### 19.2 WWDG Key Features

- Programmable free-running decrement counter
- conditional reset (computing)
  - When the value of the decrement counter is less than 0x40, (if the watchdog is started) a reset is generated.
  - When the decrementing counter is reloaded outside the window, (if the watchdog is started) a reset is generated. See 0.
- If the watchdog is started and interrupts are allowed, an Early Wakeup Interrupt (EWI) is generated when the decrementing counter equals 0x40, which can be used to reload the counter to avoid a WWDG reset.

### 19.3 WWDG Functional Description

A reset is generated if the watchdog is activated (the WDGA bit in the WWDG\_CR register is set to '1') and when the 7-bit (T[6:0]) decrementing counter is flipped from 0x40 to 0x3F (the T6 bit is cleared). A reset is generated if software reloads the counter when the counter value is greater than the value in the window register.

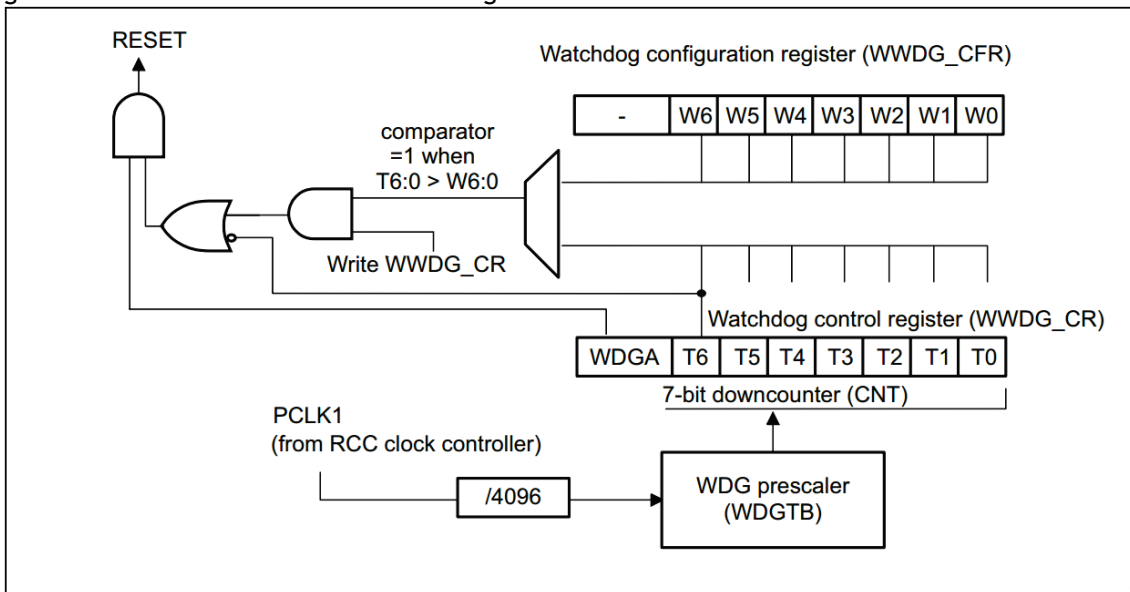


Figure 179 Watchdog Block Diagram

The application program must periodically write to the WWDG\_CR register during normal operation to prevent a reset of the MCU from occurring. A write operation can only be performed if the counter value is less than the value of the window register. The value stored in the WWDG\_CR register must be between 0xFF and 0xC0:

- start the watchdog
- After a system reset, the watchdog is always off, and setting the WDGA bit of the WWDG\_CR register enables the watchdog to be turned on. watchdog, after which it cannot be turned off again unless a reset occurs.
- Control Decrement Counter

The decrement counter is in a free-running state and continues to count down even if the watchdog is disabled. When the watchdog is enabled, the T6 bit must be set to prevent a reset from being generated immediately.

The T[5:0] bits contain the number of timings before the watchdog generates a reset; the delay time before reset varies between a minimum and a maximum value. maximum value; this is because the preshunt value is unknown when the WWDG\_CR register is written.

The configuration register (WWDG\_CFR) contains the upper limit value of the window: to avoid generating a reset, the decrement counter must be reloaded when its value is less than the value of the window register and greater than 0x3F is reloaded. 0 describes the operation of the window register.

Another way to reload the counter is to utilize the Early Wakeup Interrupt (EWI). Setting the WEI bit in the WWDG\_CFR register turns on this interrupt. This interrupt is generated when the decrementing counter reaches 0x40, and the corresponding interrupt service program (ISR) can be used to load the counter to prevent a WWDG reset. This interrupt can be cleared by writing '0' in the WWDG\_SR register.

*Notes: A software reset can be generated with the T6 bit (set the WDGA bit to '1' and the T6 bit to '0').*

## 19.4 How to Write a Watchdog Timeout Program

The window watchdog timeout can be calculated using the formula provided by 0.

**WARNING:** When writing to the WWDG\_CR register, always set the T6 bit to '1' to avoid immediately generating a reset.

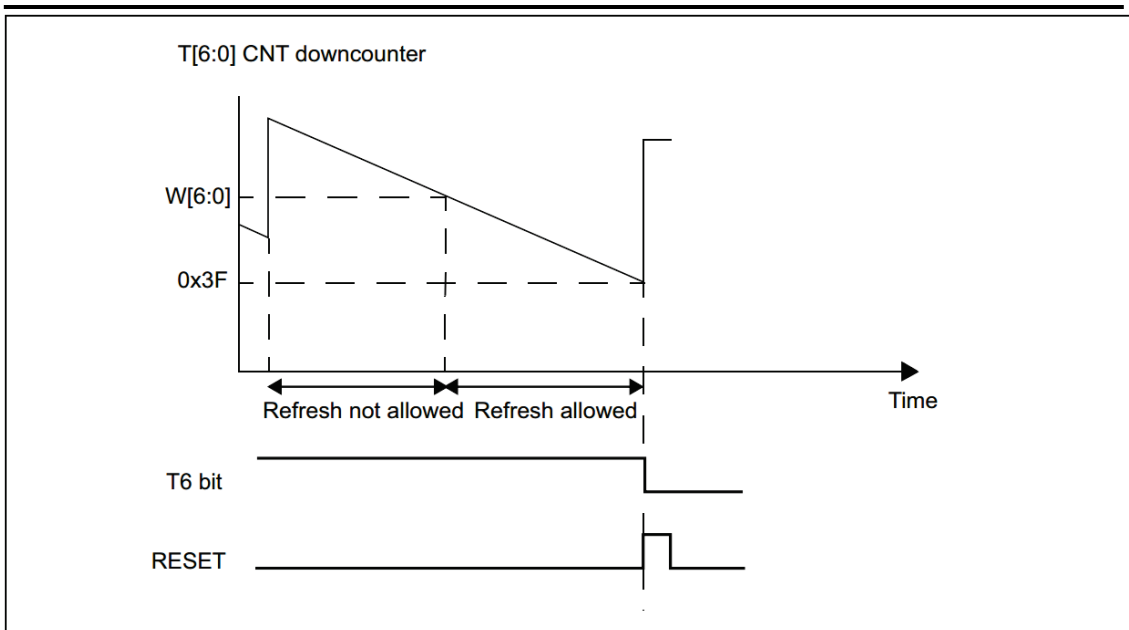


Figure180 Window Watchdog Timing Chart

The formula to calculate the WWDG timeout value is given by:

$$t_{WWDG} = t_{PCLK1} \times 4096 \times 2^{WDGTB[1:0]} \times (T[5:0] + 1) \quad (ms)$$

where:

$t_{WWDG}$ : WWDG timeout

$t_{PCLK1}$ : APB1 clock period measured in ms

4096: value corresponding to internal divider

As an example, let us assume APB1 frequency is equal to 24 MHz, WDGTB[1:0] is set to 3 and T[5:0] is set to 63:

$$t_{WWDG} = \frac{1}{24000} \times 4096 \times 2^3 \times (63 + 1) = 21.85ms$$

For the minimum and maximum values of  $t_{WWDG}$ , refer to the following table:

Table88 Minimum and maximum timeout values @36 MHz ( $f_{PCLK1}$ )

Prescaler	WDGTB	Min timeout value	Max timeout value
1	0	113 $\mu$ s	7.28 ms
2	1	227 $\mu$ s	14.56 ms
4	2	455 $\mu$ s	29.12 ms
8	3	910 $\mu$ s	58.25 ms

## 19.5 Debug Mode

When the microcontroller enters debug mode (Cortex-M3 core stop), the WWDG counters are able to continue working or stop depending on the state of the DBG\_WWDG\_STOP configuration bit in the debug module. See Section31.16.2 for details.

## 19.6 Register Description

See Section1 for details on the abbreviations used in the register descriptions.

These peripheral registers can be operated in half-word (16-bit) or word (32-bit) format.

### 19.6.1 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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								WDGA	T6	T5	T4	T3	T2	T1	T0
								rs	rw	rw	rw	rw	rw	rw	rw

Bit	notation	clarification
31:8	Reserved	Reserved, always reads 0.
7	WDGA	<b>WDGA:</b> Activation bit This bit is set '1' by software, but can only be cleared '0' by hardware after a reset. The watchdog can generate a reset when WDGA=1. 0: Prohibition of watchdogs 1: Enable Watchdog
6:0	T[6:0]	<b>T[6:0]:</b> 7-bit counter (MSB to LSB) (7-bit counter) These bits are used to store the watchdog counter value. Every (4096x2WDGTB) PCLK1 cycles is decremented by 1. A watchdog reset is generated when the counter value changes from 40h to 3Fh (T6 becomes 0).

### 19.6.2 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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						EWI	WDGTB 1	WDGTB 0	W6	W5	W4	W3	W2	W1	W0
						rs	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	notation	clarification
31:8	Reserved	Reserved, always reads 0.
9	EWI	<b>EWI:</b> Early wakeup interrupt (Early wakeup interrupt) If this bit is set to '1', an interrupt is generated when the counter value reaches 40h. This interrupt can only be cleared by hardware after a reset.
8:7	WDGTB[1:0]	<b>WDGTB[1:0]:</b> Time base (Timer base) The time base of the prescaler can be set as follows: 00:CK timer clock (PCLK1 divided by 4096) divided by 1 01:CK timer clock (PCLK1 divided by 4096) divided by 2 10:CK timer clock (PCLK1 divided by 4096) divided by 4 11:CK timer clock (PCLK1 divided by 4096) divided by 8
6:0	W[6:0]	<b>W[6:0]:</b> 7-bit window value (7-bit window value) These bits contain the window value used for comparison with the decrementing counter.

### 19.6.3 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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															EWIF
rc_w0															

Bit	notation	clarification
31:1	Reserved	Reserved, always reads 0.
0	EWIF	<b>EWIF:</b> Early wakeup interrupt flag (Early wakeup interrupt flag) This bit is set '1' by hardware when the counter value reaches 40h. It must be cleared by a software write '0'. Writing a '1' to this bit is not valid. This bit is also set '1' if interrupts are not enabled.

### 19.6.4 WWDG Register Image

Table89 WWDG Register Image and Reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h	WWDG_CR	Reserved																							WDGA	T[6:0]							
	Reset value																								0	1	1	1	1	1	1	1	
004h	WWDG_CFR	Reserved																					EWI	WDGTB1	WDGTB0	W[6:0]							
	Reset value																						0	0	0	1	1	1	1	1	1	1	
008h	WWDG_SR	Reserved																															FWIF
	Reset value																																0

SeeTable1 for register start addresses.

## 20 True Random Number Generator (TRNG)

### 20.1 TRNG Profile

The TRNG unit is used to generate true random number sequences. Each operation generates a 128-bit true random number sequence. It can be configured to generate a CPU interrupt request after random number generation.

### 20.2 Register Description

See Section1 for details on the abbreviations used in the register descriptions.  
These peripheral registers can be operated in half-word (16-bit) or word (32-bit) format.

#### 20.2.1 Control Status Register (RNG\_CSR)

Address offset: 0x00

Reset value: 0x0000 007F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										EDGE_SEL	INT	Reserved	ATT ACK	Reserved	S128
										rw	rw		rw		rw

Bit	notation	clarification
31:6	Reserved	Reserved, always reads 0.
5	EDGE_SEL	EDGE_SEL: RNG data sampling edge selection 0: Rising edge sampling 1: Falling edge sampling
4	INT	INT: Interrupt enable 0: No interrupt is generated 1: An interrupt is generated
3	Reserved	Reserved, always reads 0.
2	ATTACK	ATTACK: Detected 47 consecutive cells with 0 or 1 attack status bits 0: No attack 1: Attack detected
1	Reserved	Reserved, always reads 0.
0	S128	S128: 128-bit random number status bit. Set to '1' by hardware, cleared to '0' by software. After being cleared to '0', it automatically starts generating a new 128-bit random number. 0: 128-bit random number not generated 1: 128-bit random number generated

#### 20.2.2 Data Register (RNG\_DR)

Address offset: 0x04

Reset value: 0x0000 0000

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

Bit	notation	clarification
31:0	DATA	DATA: 32-bit random number data. It is necessary to wait until the S128 bit of the RNG_CSR register is set to '1' before reading. Read this register 4 consecutive times to obtain a 128-bit random number.

## 20.2.3 Analog Control Register (RNG\_AMA)

Address offset: 0x0C

Reset value: 0x000F F486

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved			ANA_OUT	Reserved											
			rw												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PD_TRNG				Reserved											
rw															
Bit	notation			clarification											
31:29	Reserved			Reserved, always reads 0.											
28	ANA_OUT			ANA_OUT: Output selection 0: Output after data processing 1: Direct simulation output											
27:16	Reserved			Reserved, always reads 0.											
15:12	PD_TRNG			PD_TRNG: Selection of PD signals from 4 analog random sources bit12: corresponds to random source 0 bit13: corresponds to random source 1 bit14: corresponds to random source 2 bit15: corresponds to random source 3											
11:0	Reserved			Reserved, always reads 0.											

## 20.2.4 Pseudo-Random Sequence Register (RNG\_PN)

Address offset: 0x10

Reset value: 0x69D8 4C18

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PN[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
PN[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit	notation			clarification											
31:0	PN			PN: 32-bit pseudo-random sequence, updated once per system cycle. The initial value is configurable.											

## 20.2.5 FIFO Status Register (RNG\_INDEX)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIFO_RD_OV	Reserved														
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														RNG_INDEX	
														r	r
Bit	notation			clarification											
31	FIFO_RD_OV			FIFO_RD_OV: FIFO read overflow flag. When a new 128-bit random number is read completely, and then read or write operations are performed on the FIFO again, this bit is set to '1' and cleared to '0' by software.											
30:2	Reserved			Reserved.											
1:0	RNG_INDEX			RNG_INDEX: The read depth of FIFO.											



## 20.2.6 RNG Register Image

Table90 RNG Register Image and Reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h	RNG_CSR	Reserved																										EDGE_SEL	INT	Reserved	ATTACK	Reserved	5128
	reset																											1	0	0	0	0	
004h	RNG_DR	DR[31:0]																															
	reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
008h	Reserved																																
00Ch	RNG_AMA	Reserved		ANA_OUT	Reserved												PD_TRNG [3:0]				Reserved												
	reset				0													1	1	1	1												
010h	RNG_PN	PN[31:0]																															
	reset	0	1	1	0	1	0	0	1	1	1	0	1	1	0	0	0	0	1	0	0	1	1	0	0	0	0	0	1	1	0	0	0
014h	RNG_INDEX	FIFO_RD_OV	Reserved																														RNG_INDEX
	reset		0																														

SeeTable1 for register start addresses.

## 21 System Configuration Register (SYSCFG)

### 21.1 Register Description

See Section1 for details on the abbreviations used in the register descriptions.  
These peripheral registers can be operated in half-word (16-bit) or word (32-bit) format.

#### 21.1.1 Configure the Lock Register (SYSCFG\_LOCK)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK[31:16]															
WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LOCK[15:0]															
WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
Bit	notation	clarification													
31:0	Lock	LOCK: SYSCFG lock and unlock register. Writing 0xCDED3526 unlocks the SYSCFG-related registers. Writing ~(0xCDED3526) locks the SYSCFG-related registers.													

#### 21.1.2 SENSOR Enable Register (SENSOR\_EN)

Address offset: 0xC0

Reset value: 0xAAAA AAAA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
VBAT_VG_EN				VBAT_VL_EN				VBAT_VH_EN				VDD_VL_EN			
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
VDD_VH_EN				VDD_VG_EN				TH_EN				TL_EN			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit	notation	clarification													
31:28	VBAT_VG_EN	VBAT_VG_EN: Battery burr alarm enable. 0xA: Disable other: Enable													
27:24	VBAT_VL_EN	VBAT_VL_EN: Battery low voltage alarm enable. 0xA: Disable the enable other: Enable the enable													
23:20	VBAT_VH_EN	VBAT_VH_EN: Battery high voltage alarm enable. 0xA: Disable other: Enable													
19:16	VDD_VL_EN	VDD_VL_EN: Main power supply low-voltage alarm enable. 0xA: Disable other: Enable													
15:12	VDD_VH_EN	VDD_VH_EN: Main power high-voltage alarm enable. 0xA: Disable the enable other: Enable the enable													
11:8	VDD_VG_EN	VDD_VG_EN: Main power supply glitch alarm enable. 0xA: Disable other: Enable													
7:4	TH_EN	TH_EN: High temperature alarm enable. 0xA: Disable other: Enable													
3:0	TL_EN	TL_EN: Low temperature alarm enable. 0xA: Disable other: Enable													

### 21.1.3 SENSOR Status Register (SENSOR\_STAT)

Address offset: 0xC4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								VBAT_VG_RST	VBAT_VH_RST	VBAT_VL_RST	VDD_VG_RST	VDD_VH_RST	VDD_VL_RST	TH_RST	TL_RST
								r	r	r	r	r	r	r	r

Bit	notation	clarification
31:6	Reserved	Reserved, always reads 0.
7	VBAT_VG_RST	VBAT_VG_RST: Battery glitch alarm indicator signal.
6	VBAT_VH_RST	VBAT_VH_RST: Battery high-voltage alarm indicator signal.
5	VBAT_VL_RST	VBAT_VL_RST: Battery low voltage alarm indicator signal.
4	VDD_VG_RST	VDD_VG_RST: Main power supply glitch alarm indicator signal.
3	VDD_VH_RST	VDD_VH_RST: Main power supply high-voltage alarm indication signal.
2	VDD_VL_RST	VDD_VL_RST: Main power supply low voltage alarm indication signal.
1	TH_RST	TH_RST: High temperature alarm indicator signal.
0	TL_RST	TL_RST: Low temperature alarm indication signal.

### 21.1.4 SENSOR Voltage Glitch Detection Configuration Register (SENSOR\_VG\_ACTION)

Address offset: 0xC8

Reset value: 0x0000 0004

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

Bit	notation	clarification
31:3	Reserved	Reserved, always reads 0.
2	DLY_BYPASS	DLY_BYPASS: Glitch filtering function. 0: Enable the detected glitch filtering function. 1: Disable the detected glitch filtering function.
1:0	DLY_SEL	DLY_SEL: Used to select the width of the glitch that can trigger an attack. 00: 225us 01: 800us 10: 3.2ms 11: 12.8ms

## 21.1.5 Security Algorithm Configuration Register (SSC\_CLK\_EN)

Address offset: 0xCC  
Reset value: 0x0000 0000

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

Bit	notation	clarification
31:4	Reserved	Reserved, always reads 0.
3:2	CRYPT_VAL	CRYPT_VAL: Algorithm frequency division register. 00: No frequency division 01: 2-frequency division 10: 4-frequency division 11: 8-frequency division
1	RCC_SYMC_EN	RCC_SYMC_EN: Symmetric algorithm clock enable. 0: Off 1: On
0	RCC_RNG_EN	RCC_RNG_EN: Random number clock enable. 0: Off 1: On

## 21.1.6 Security Algorithm Configuration Register (SSC\_CLK\_EN)

Address offset: 0xD0  
Reset value: 0x0000 0000

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

Bit	notation	clarification
31:4	Reserved	Reserved, always reads 0.
3	INTP_EN	INTP_EN: Sensor attack interrupt enable 0: Disable 1: Enable
2	ACT_RST	ACT_RST: Sensor attack reset enable 0: Off 1: On
1	VBAT_VG_CLR	VBAT_VG_CLR: Battery voltage glitch removal. 0: Do not remove 1: Remove
0	VDD_VG_CLR	VDD_VG_CLR: Main power supply voltage glitch removal. 0: Do not remove 1: Remove

## 21.1.7 SENSOR Status Register (SENSOR\_STATE\_FLAG)

Address offset: 0xE0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VBAT_VG_FLAG_CLR	VBAT_VH_FLAG_CLR	VBAT_VL_FLAG_CLR	VDD_VG_FLAG_CLR	VDD_VH_FLAG_CLR	VDD_VL_FLAG_CLR	TH_FLAG_CLR	TL_FLAG_CLR	VBAT_VG_FLAG	VBAT_VH_FLAG	VBAT_VL_FLAG	VDD_VG_FLAG	VDD_VH_FLAG	VDD_VL_FLAG	TH_FLAG	TL_FLAG
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	notation	clarification
31:16	Reserved	Reserved, always reads 0.
15	VBAT_VG_FLAG_CLR	VBAT_VG_FLAG_CLR: Clear the battery glitch alarm indication signal.
14	VBAT_VH_FLAG_CLR	VBAT_VH_FLAG_CLR: Clear the battery high-voltage alarm indication signal.
13	VBAT_VL_FLAG_CLR	VBAT_VL_FLAG_CLR: Clear the battery low voltage alarm indication signal.
12	VDD_VG_FLAG_CLR	VDD_VG_FLAG_CLR: Clear the main power supply glitch alarm indication signal.
11	VDD_VH_FLAG_CLR	VDD_VH_FLAG_CLR: Clear the main power high-voltage alarm indication signal.
10	VDD_VL_FLAG_CLR	VDD_VL_FLAG_CLR: Clear the main power low-voltage alarm indication signal.
9	TH_FLAG_CLR	TH_FLAG_CLR: Clear the high-temperature alarm indication signal.
8	TL_FLAG_CLR	TL_FLAG_CLR: Clear the low-temperature alarm indicator signal.
7	VBAT_VG_FLAG	VBAT_VG_FLAG: Battery glitch alarm indicator signal.
6	VBAT_VH_FLAG	VBAT_VH_FLAG: Battery high-voltage alarm indicator signal.
5	VBAT_VL_FLAG	VBAT_VL_FLAG: Battery low voltage alarm indicator signal.
4	VDD_VG_FLAG	VDD_VG_FLAG: Main power supply glitch alarm indicator signal.
3	VDD_VH_FLAG	VDD_VH_FLAG: Main power supply high-voltage alarm indicator signal.
2	VDD_VL_FLAG	VDD_VL_FLAG: Main power supply low voltage alarm indicator signal.
1	TH_FLAG	TH_FLAG: High temperature alarm indication signal.
0	TL_FLAG	TL_FLAG: Low temperature alarm indicator signal.

## 21.1.8 SENSOR Interrupt Status Register (SENSOR\_INT\_FLAG)

Address offset: 0xE4

Reset value: 0x0000 0000

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

Bit	notation	clarification
31:1	Reserved	Reserved, always reads 0.
0	INT_FLAG	INT_FLAG: Interrupt flag.

## 21.1.9 SYSCFG Register Image

Table91 SYSCFG Register Image and Reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0							
000h	SYS_LOCK	LOCK[31:0]																																						
	reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								
004h~04Fh	Reserved																																							
050h~0BFh	OTP Regsiter																																							
0C0h	SENSOR_EN	VBAT_VG_EN		VBAT_VL_EN		VBAT_VH_EN		VDD_VL_EN		VDD_VH_EN		VDD_VG_EN		TH_EN		TL_EN																								
	reset	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0							
0C4h	SENSOR_STAT	Reserved																										VBAT_VB_RS	VBAT_VH_R	VBAT_VL_RS	VDD_VG_RS	VDD_VH_RS	VDD_VL_RST	TH_RST	TL_RST					
	reset																											0	0	0	0	0	0	0	0	0	0			
0C8h	SENSOR_VG_ACTION	Reserved																														DLY_BYPASS		DLY_SEL						
	reset																															1	0	0	0					
0CCh	SYS_CLK_EN	Reserved																														CRYPT_VAL		RCC_SYMC_E						
	reset																															0	0	0	0					
0D0h	SENSOR_ACT	Reserved																															INTP_EN		ACT_RST		VBAT_VG_CLR		VDD_VG_CLR	
	reset																															0	0	0	0	0	0			
0D4h~0DFh	Reserved																																							
0E0h	SENSOR_STATE_FLAG	Reserved																VBAT_VG_FLAG_CLR	VBAT_VH_FLAG_CLR	VBAT_VL_FLAG_CLR	VDD_VG_FLAG_CLR	VDD_VH_FLAG_CLR	VDD_VL_FLAG_CLR	TH_FLAG_CLR	TL_FLAG_CLR	VBAT_VG_FLAG	VBAT_VH_FLAG	VBAT_VL_FLAG	VDD_VG_FLAG	VDD_VH_FLAG	VDD_VL_FLAG	TH_FLAG	TL_FLAG							
	reset																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						
0E4h	SENSOR_INT_FLAG	Reserved																															INT_FLAG							
	reset																																0	0						

SeeTable1 for register start addresses.

## 22 One-Time Programmable (OTP)

### 22.1 Profile

One-Time Programmable (OTP) is a type of non-volatile memory that can be written only once, and cannot be erased or modified after writing.

### 22.2 Register Description

See Section1 for details on the abbreviations used in the register descriptions.  
These peripheral registers can be operated in half-word (16-bit) or word (32-bit) format.

#### 22.2.1 OTP Data 0 Register (OTP\_DATA0)

Address offset: 0x00

Reset value: 0xFFFF XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA0[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
DATA0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit	notation	clarification													
31:0	DATA0	DATA0: DATA0 data area.													

#### 22.2.2 OTP Data 1 Register (OTP\_DATA1)

Address offset: 0x00

Reset value: 0xFFFF XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA1[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
DATA1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit	notation	clarification													
31:0	DATA1	DATA1: DATA1 data area.													

#### 22.2.3 OTP Data 2 Register (OTP\_DATA2)

Address offset: 0x00

Reset value: 0xFFFF XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA2[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
DATA2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit	notation	clarification													
31:0	DATA2	DATA2: DATA2 data area.													

## 22.2.4 OTP Data 3 Register (OTP\_DATA3)

Address offset: 0x00

Reset value: 0xFFFF XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA3[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
DATA3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit	notation	clarification													
31:0	DATA3	DATA3: DATA3 data area.													

## 22.2.5 OTP Data 4 Register (OTP\_DATA4)

Address offset: 0x00

Reset value: 0xFFFF XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA4[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
DATA4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit	notation	clarification													
31:0	DATA4	DATA4: DATA4 data area.													

## 22.2.6 OTP Data 5 Register (OTP\_DATA5)

Address offset: 0x00

Reset value: 0xFFFF XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA5[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
DATA5[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit	notation	clarification													
31:0	DATA5	DATA5: DATA5 data area.													

## 22.2.7 OTP Data 6 Register (OTP\_DATA6)

Address offset: 0x00

Reset value: 0xFFFF XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA6[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
DATA6[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit	notation	clarification													
31:0	DATA6	DATA6: DATA6 data area.													



## 22.2.8 OTP Data 7 Register (OTP\_DATA7)

Address offset: 0x00

Reset value: 0xFFFF XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA7[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
DATA7[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit	notation	clarification													
31:0	DATA7	DATA7: DATA7 data area.													

## 22.2.9 OTP Configuration Register (OTP\_CTRL)

Address offset: 0x20

Reset value: 0x0000 0000

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

Bit	notation	clarification
31:3	Reserved	Reserved, always reads 0.
2	OTP_BUSY	OTP_BUSY: OTP programming status. 0: Programming completed 1: Programming in progress
1	Reserved	Reserved, always reads 0.
0	OTP_WRITE	OTP_WRITE: Enable writing to OTP. 0: Disable write enable. 1: Enable write enable. Note: Before enabling write enable, ensure that the data to be written in the otp_w_data register is correct.

## 22.2.10 OTP Programming Register (OTP\_WR)

Address offset: 0x24

Reset value: 0x0000 0000

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

Bit	notation	clarification
31:16	Reserved	Reserved, always reads 0.
15:8	OTP_ADDR	OTP_ADDR: OTP programming address.
7:0	OTP_W_DATA	OTP_W_DATA: OTP programming data.

## 22.2.11 OTP\_10ns Register (OTP\_10ns)

Address offset: 0x28

Reset value: 0x0000 0038

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CFG_10NS															
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
RCFG_1US															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit	notation	clarification													
31:16	CFG_10NS	CFG_10NS: Configure 10ns time according to PCLK.													
15:0	CFG_1US	CFG_1US: Configure 1us time according to PCLK.													

## 22.2.12 OTP\_LDO Register (OTP\_LDO)

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LDO_EN_DLY[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
LDO_EN_DLY[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit	notation	clarification													
31:0	LDO_EN_DLY	LDO_EN_DLY: Configure the startup time of the 2.5V LDO.													

## 22.2.13 OTP Register Image

Table92 OTP Register Image and Reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
000h	DATA0	DATA0[31:0]																																		
	reset	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x				
004h	DATA1	DATA1[31:0]																																		
	reset	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x				
008h	DATA2	DATA2[31:0]																																		
	reset	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x				
00Ch	DATA3	DATA3[31:0]																																		
	reset	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x				
010h	DATA4	DATA4[31:0]																																		
	reset	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x				
014h	DATA5	DATA5[31:0]																																		
	reset	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x				
018h	DATA6	DATA6[31:0]																																		
	reset	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x				
01Ch	DATA7	DATA7[31:0]																																		
	reset	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x				
020h	OTP_CTRL	Reserved																											OTP_BUSY	Reserved	OTP_WRITE					
	reset																												0		0					
024h	OTP_WR	Reserved															OTP_ADDR						OTP_W_D													
	reset																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
028h	OTP_10NS	CFG_10NS															CFG_1US																			
	reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
02Ch	OTP_LDO	LDO_EN_DLY[31:0]																																		
	reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			

SeeTable1 for register start addresses.

## 23 SDIO Interface (SDIO)

### 23.1 SDIO Main Functions

The SD/SDIOMMC Card Host Module (SDIO) provides an operational interface between the AHB peripheral bus and multimedia cards (MMC), SD memory cards, SDIO cards, and CE-ATA devices. The MultiMediaCard System Specification is published by the MMCA Technical Committee and is available on the MultiMediaCard Association Web site ([www.mmca.org](http://www.mmca.org)).

The CE-ATA System Specification is available on the CE-ATA Working Group's Web site ([www.ce-ata.org](http://www.ce-ata.org)). The main functions of the SDIO are as follows:

- Fully compatible with MultiMediaCard System Specification version 4.2. Supports three different data bus modes: 1-bit (default), 4-bit and 8-bit.
- Fully compatible (forward compatible) with earlier versions of the MultiMediaCard system specification.
- Fully compatible with SD memory card specification version 2.0.
- Fully compatible with SDI/O card specification version 2.0: Supports different data bus modes: 1-bit (default) and 4-bit.
- Fully supports CE-ATA functionality (fully compatible with CE-ATA digital protocol version 1.1).
- Data transfer rate up to 48MHz in 8-bit bus mode.
- Data and command output enable signals for controlling external bi-directional drivers.

**Notes:**

1. SDIO does not have SPI-compatible communication mode
2. In the MultiMediaCard System Specification version 2.11, the SD memory card protocol is defined as a superset of the MultiMediaCard protocol. The I/O portion of an SD card or composite card that only supports I/O mode cannot support many of the commands needed in an SD memory device, and some of the commands here do not work in an SDI/O device, such as the erase command, so SDIO does not support these commands. In addition, some commands in SD memory card and SDI/O card are different, and SDIO does not support these commands either. Details can be found in the SDI/O card datasheet version 1.0. CE-ATA support can be implemented on the MMC interface using the existing MMC command mechanism. the electrical and signaling definitions of the SDIO interface are detailed in the MMC reference. The multimedia card/SD bus connects all cards to the controller. The current version of SDIO can only support one SD/SDIO/MMC4.2 card at a time, but can support multiple MMC version 4.1 or previous versions.

### 23.2 SDIO Bus Topology

Communication on the bus is achieved by transmitting commands and data.

The basic operation on a multimedia card/SD/SDI/O bus is a command/response structure, such that the bus operation realizes the exchange of information under a command or bus mechanism; in addition, some operations have data tokens.

Data transferred on SD/SDIO memory cards is transferred in the form of data blocks; data transferred on MMCs is transferred in the form of data blocks or data streams; and data transferred on CE-ATA devices is also transferred in the form of data blocks.

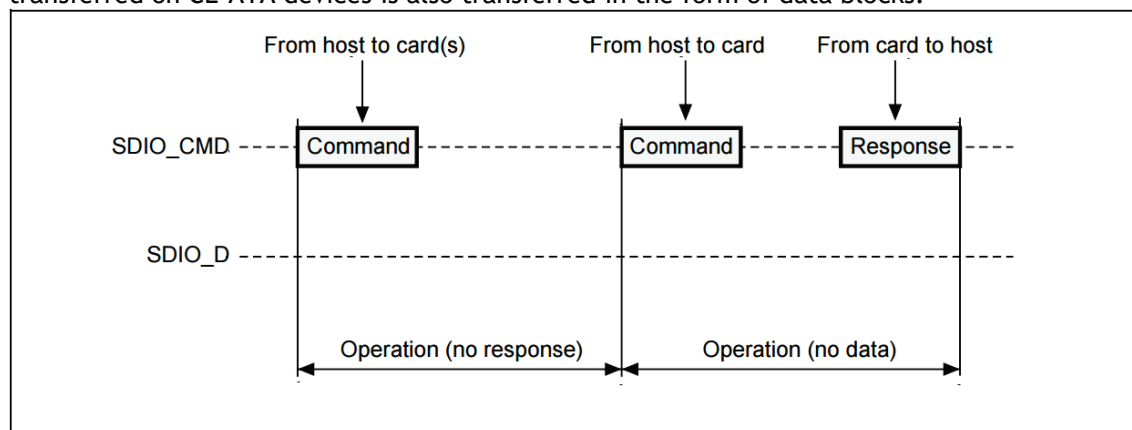


Figure181 SDIO "No Response" and "No Data" Operations

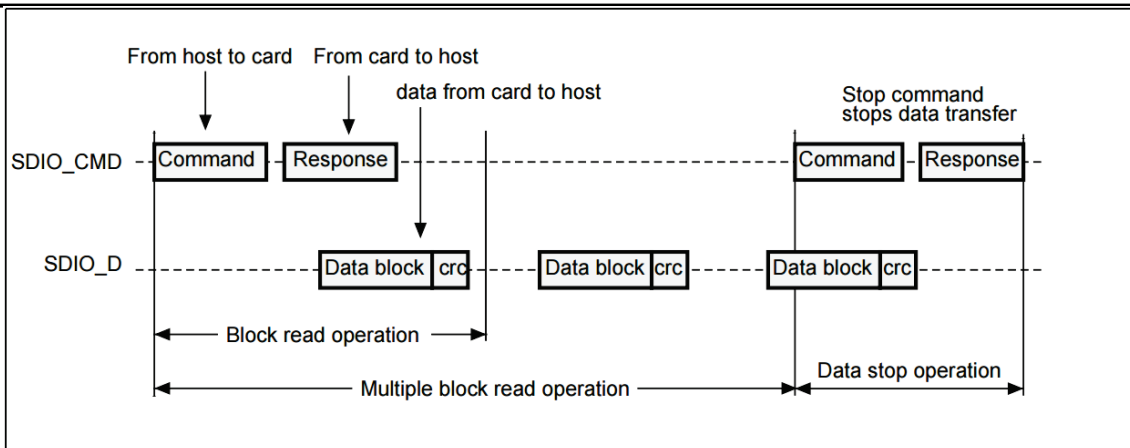


Figure182 SDIO (Multi) Data Block Read Operation

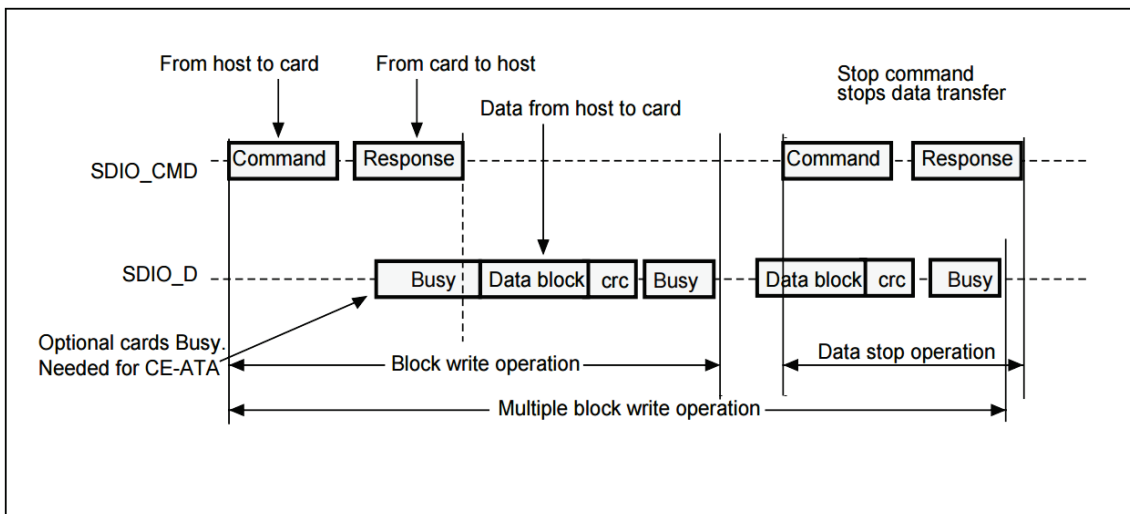


Figure183 SDIO (Multi) Data Block Write Operation

Notes: SDIO (SDIO\_D0 is pulled low) will not send any data when there is a Busy signal.

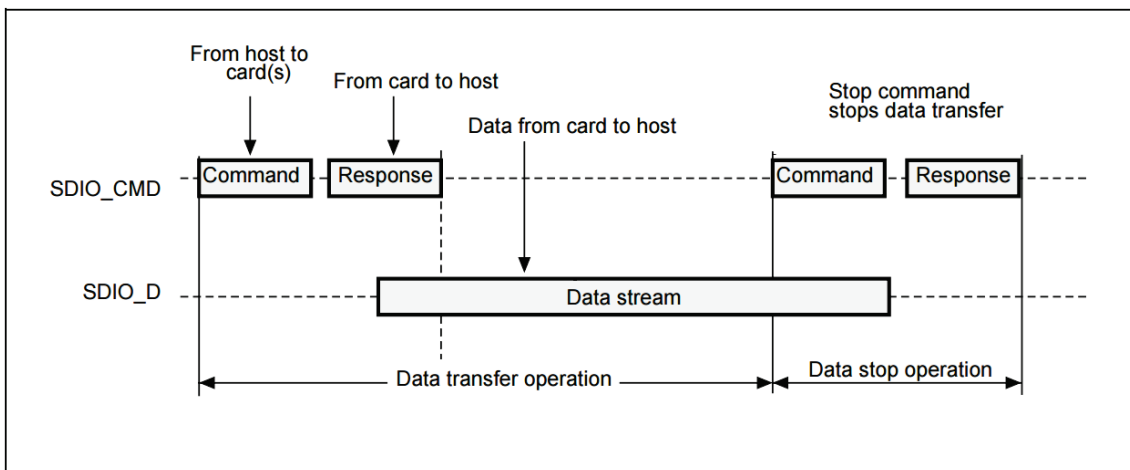


Figure184 SDIO Sequential Read Operation

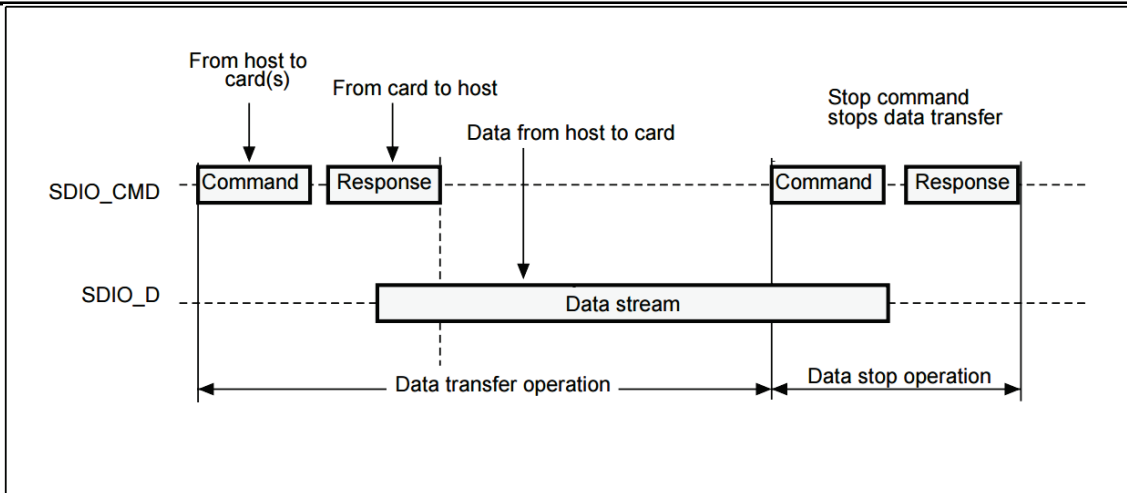


Figure185 SDIO Sequential Write Operation

## 23.3 SDIO Functional Description

The SDIO contains 2 parts:

- SDIO Adapter Module: Realizes all MMC/SD/SDI/O card related functions, such as clock generation, command and data transmission.
- AHB bus interface: operates registers in the SDIO adapter module and generates interrupt and DMA request signals.

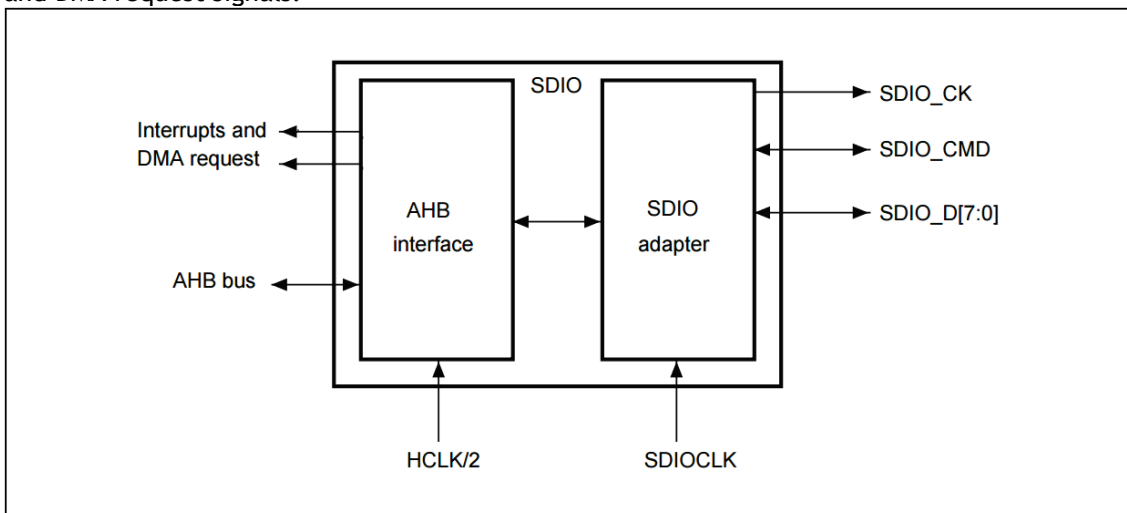


Figure186 SDIO Block Diagram

By default after reset SDIO\_D0 is used for data transfer. After initialization the host can change the width of the data bus.

If a multimedia card is connected to the bus, SDIO\_D0, SDIO\_D[3:0] or SDIO\_D[7:0] can be used for data transfer. MMC version V3.31 and previous versions of the protocol only support 1-bit data lines, so only SDIO\_D0 can be used.

If an SD or SDI/O card is connected to the bus, data transfer can be configured through the host to use SDIO\_D0 or SDIO\_D[3:0]. All data lines operate in push-pull mode.

SDIO\_CMD has two modes of operation:

- Open mode for initialization (only for MMC version V3.31 or before)
- Push-pull mode for command transfer (SD/SDI/O cards and MMCV4.2 also use push-pull drivers during initialization)

SDIO\_CK is the card's clock: 1 bit of command or data is transmitted on the command and data lines per clock cycle. The clock frequency can be varied from 0MHz to 20MHz for MultiMediaCard V3.31 protocols, from 0MHz to 48MHz for MultiMediaCard V4.0/4.2 protocols, and from 0MHz to 25MHz for SD or SDI/O cards.

SDIO uses two clock signals:

- SDIO adapter clock (SDIOCLK=HCLK)

- AHB bus clock (HCLK/2)

The following table applies to the MultiMediaCard/SD/SDIO Card bus:

Table93 SDIO Pin Definitions

pinout	orientations	clarification
SDIO_CK	exports	Multimedia Card/SD/SDIO Card Clock. This is the clock line from the host to the card.
SDIO_CMD	bi-directionality	Multimedia card/SD/SDIO card command. This is a bidirectional command/response signal line.
SDIO_D[7:0]	bi-directionality	Multimedia card/SD/SDIO card data. These are bi-directional data buses.

### 23.3.1 SDIO Adapter

The following figure shows a simplified SDIO adapter block diagram:

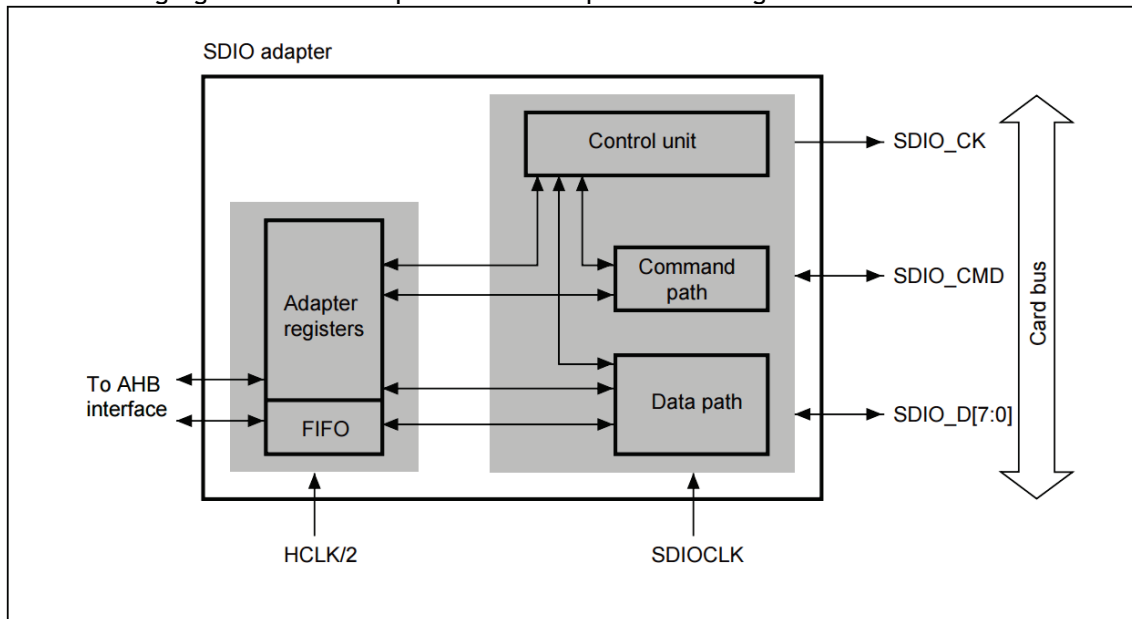


Figure187 SDIO Adapter

The SDIO adapter is the main device (host) of the Multimedia/Encrypted Digital Memory Card bus for connecting a set of Multimedia Cards or Encrypted Digital Memory Cards, which consists of the following 5 parts:

- Adapter Register Module
- control unit
- command channel
- data channel
- Data FIFO

*Notes: Adapter registers and FIFOs use the clock on the AHB bus side (HCLK/2), and control units, command channels, and data channels use the clock on the SDIO adapter side (SDIOCLK).*

#### Adapter Register Module

The Adapter Registers module contains all system registers. The module also generates signals to clear the static markers in the multimedia card, which is generated when a '1' is written to the corresponding bit in the SDIO clear register.

#### Control Unit

The control unit contains the power management functions and the clock division for the memory card. There are three power supply stages:

- Power off
- power activation
- power on

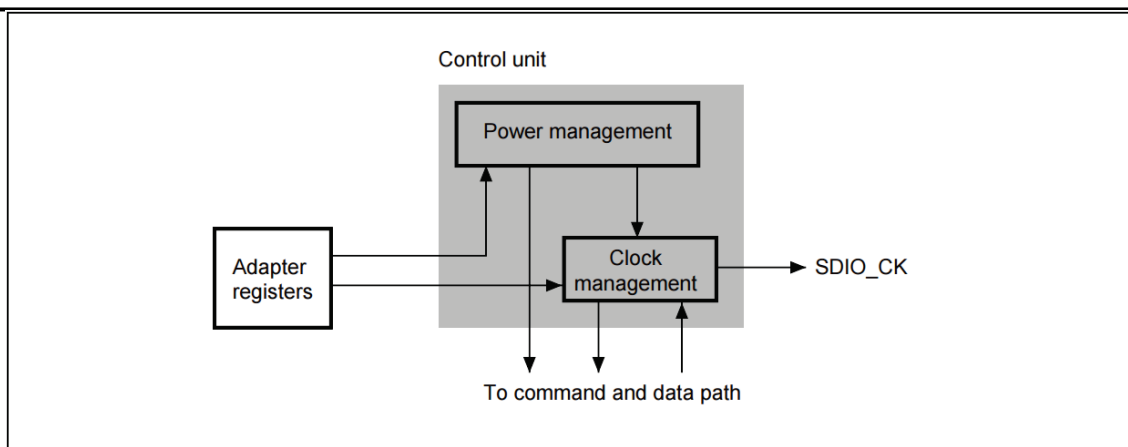


Figure188 Control Unit

The above figure shows the block diagram of the control unit with power management and clock management subunits.

The power management subunit turns off the output signals on the card bus during the power-down and power-up phases.

The Clock Management Subunit generates and controls the SDIO\_CK signal. the SDIO\_CK output can be used in clock divider or clock bypass mode. There is no clock output in the following cases:

- reset (a dislocated joint, an electronic device etc)
- During power down and power up phases
- When the power saving mode is activated and the card bus is idle (8 clock cycles after the command channel and data channel subunits enter the idle phase)

## Command Channel

The Command Channel Unit sends commands to the card and receives responses from the card.

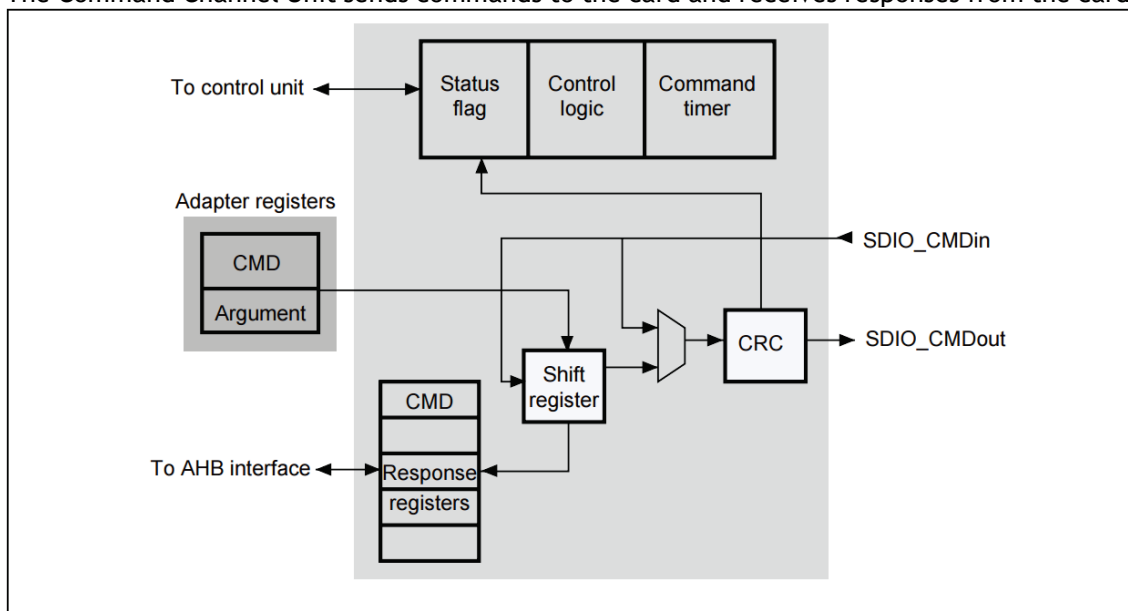


Figure189 SDIO Adapter Command Channel

- Command channel state machine (CPSM)
  - When the command register is written and the enable bit is set, the command is started to be sent. When command sending is complete, the command channel state machine (CPSM) sets the status flag and enters the idle state when a response is not required (see figure below). When a response is received, the received CRC code is compared to the internally generated CRC code and the appropriate status flag is set.



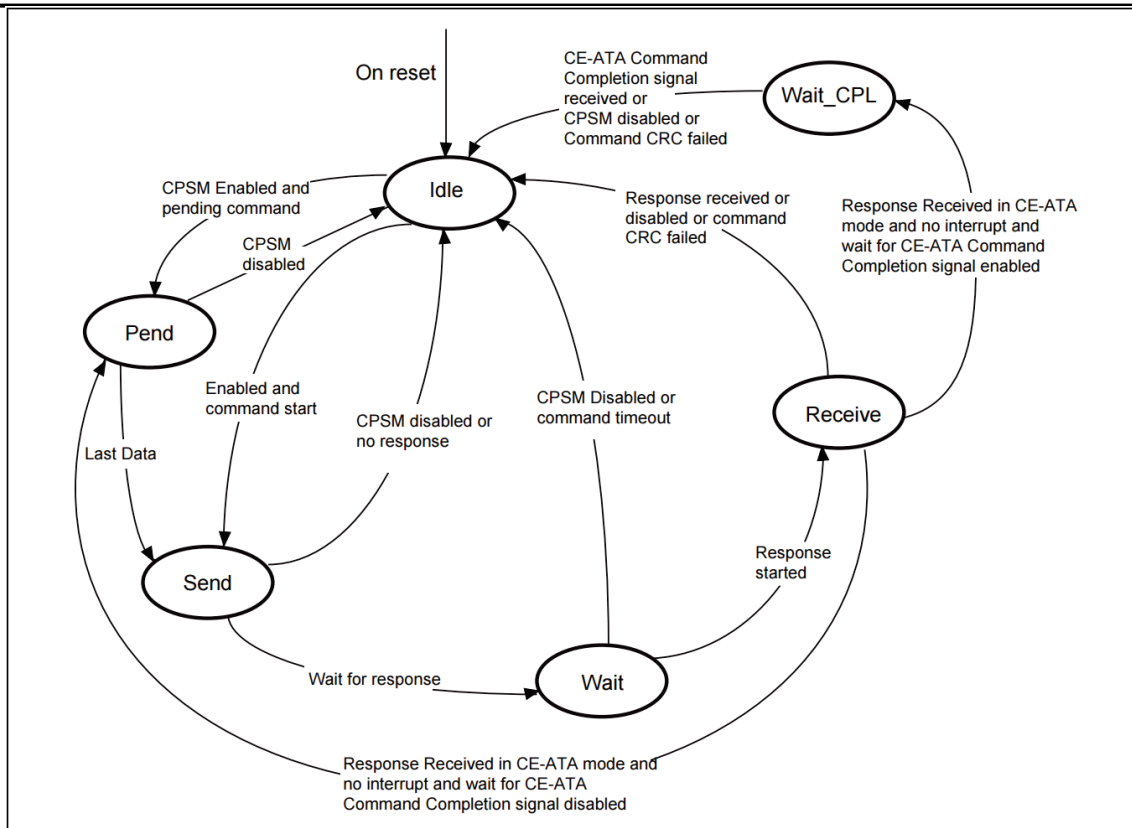


Figure190 Command Channel State Machine (CPSM)

When it enters the Wait state, the command timer starts running; when a timeout is generated before the CPSM enters the Receive state, the timeout flag is set and it enters the Idle state.

**Notes:** The command timeout is fixed at 64 SDIO\_CK clock cycles.  
 If the interrupt bit is set in the command register, the timer is turned off and the CPSM waits for an interrupt request from one of the cards. If the Pend bit is set in the Command Register, the CPSM enters the Pend (Pend) state and waits for the CmdPend signal from the data channel subunit, and when the CmdPend signal is detected, the CPSM enters the Send (Send) state, which triggers the function of the Data Counter to send a stop command.

**Notes:** The CPSM remains in the idle state for at least 8 SDIO\_CK cycles to meet the NCC and NRC timing constraints. NCC is the minimum interval between two host commands; NRC is the minimum interval between a host command and a card response.

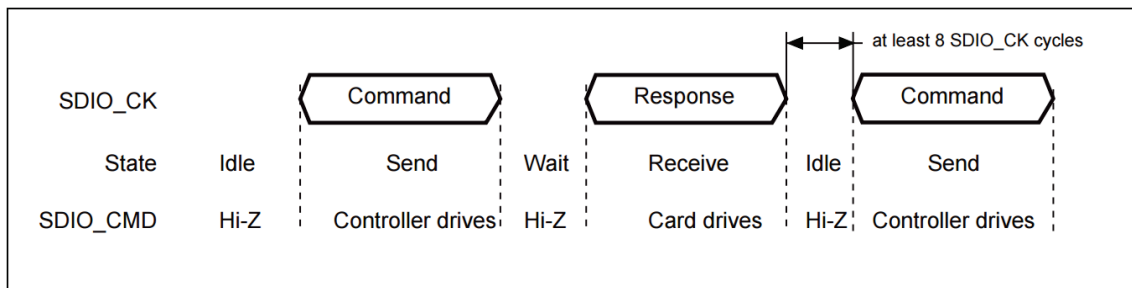


Figure191 SDIO Command Transfer

- command format
  - Command: A command is used to start an operation. The host sends commands with addresses or broadcast commands (broadcast commands are only suitable for MMCV version 3.31 or earlier) to a specified card or to all cards. Commands are transmitted serially on the CMD line. The length of all commands is fixed at 48 bits, and the following table gives the general command format on multimedia cards, SD memory cards, and SDIO cards. The CE-ATA commands are an expansion of the MMCV4.2 commands, and so have the same format.

The command channel operates in half-duplex mode so that commands and responses can be sent and received separately. If the CPSM is not in the transmit state, the SDIO\_CMD output is

in the high-resistance state as shown in Figure 191. The data on the SDIO\_CMD is synchronized with the rising edge of SDIO\_CLK.

Table 94 Command Format

Bit	breadth	numerical value	clarification
47	1	0	starting position
46	1	1	transport bit
[45:40]	6	-	Command Index
[39:8]	32	-	parameters
[7:1]	7	-	CRC7
0	1	1	end position

- Response: The response is sent to the host by a card that has been assigned an address, for MMC V3.31 or previous versions all cards

A response is also sent; a response is an answer to a previously received command. The response is transmitted serially on the CMD line.

SDIO supports 2 response types, both of which have CRC error detection:

- 48-bit short response
- 136-bit long response

**Notes:** If the response does not contain a CRC (e.g., CMD1 response), the device driver should ignore the CRC failure status.

Table 95 Short Response Format

Bit	height	numerical value	clarification
47	1	0	starting position
46	1	1	transport bit
[45:40]	6	-	Command Index
[39:8]	32	-	parameters
[7:1]	7	-	CRC7 (or 1111111)
0	1	1	end position

Table 96 Long Response Format

Bit	height	numerical value	clarification
135	1	0	starting position
134	1	0	transport bit
[133:128]	6	111111	Reserved
[127:1]	127	-	CID or CSD (including internal CRC7)
0	1	1	end position

The Command Register contains the command index (the 6 bits sent to the card) and the command type; the command itself determines whether a response is required and the type of response, 48 bits or 136 bits (see section 23.9.4). The status flags in the command channel are shown in the following table:

Table 97 Command Channel Status Flags

symbolize	clarification
CMDREND	CRC of the response is correct
CCRCFAIL	CRC error in response
CMDSENT	Commands (commands that do not require a response) have been delivered
CTIMEOUT	Response timeout
CMDACT	Command being sent.

The CRC generator calculates the CRC checksum of all bits prior to the CRC code, including the start bit, transmit bit, command index, and command parameter (or card status). For the long response format, the CRC checksum is calculated for the first 120 bits of the CID or CSD; note that the start bit, transmit bit, and six reserved bits in the long response format are not involved in the CRC calculation.

The CRC checksum is a 7-bit value:

$CRC[6:0] = \text{residue} [(M(x) \cdot x^7) / G(x)]$   
 $G(x) = x^7 + x^3 + 1$   
 $M(x) = (\text{start bit}) \cdot x^{39} + \dots + (\text{last bit before CRC}) \cdot x^0$ , or  
 $M(x) = (\text{start bit}) \cdot x^{119} + \dots + (\text{last bit before CRC}) \cdot x^0$ , or

## Data Channel

The data channel subunit transfers data between the host and the card. The following figure shows the block diagram of the data channel.

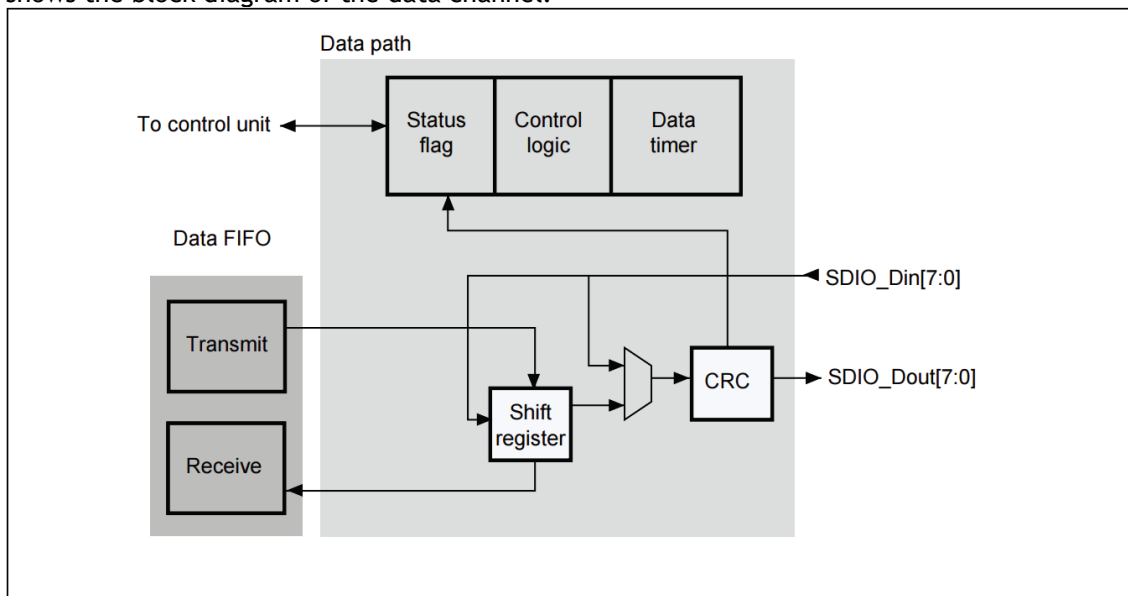


Figure192 Data Channel

The width of the card's data bus can be configured in the Clock Control Register. If 4-bit bus mode is selected, 4 bits of data will be transmitted on the four data signal lines (SDIO\_D[3:0]) each clock cycle; if 8-bit bus mode is selected, 8 bits of data will be transmitted on the eight data signal lines (SDIO\_D[7:0]) each clock cycle; if wide bus mode is not selected, then only 1 bit will be transmitted on SDIO\_D0 each clock cycle Data.

Depending on the direction of transmission (send or receive), the data channel state machine (DPSM) will enter the Wait\_S or Wait\_R state when enabled:

- Send: the DPSM enters the Wait\_S state. If there is data in the transmit FIFO, the DPSM enters the Transmit state, and at the same time the data channel subunit starts sending data to the card.
- Receive: DPSM enters the Wait\_R state and waits for the start bit; when the start bit is received, DPSM enters the receive state, and at the same time the data channel subunit starts to receive data from the card.

## Data channel state machine (DPSM)

The DPSM operates at the SDIO\_CK frequency and the card bus signals are synchronized with the rising edge of SDIO\_CK. The DPSM has six states as shown below:

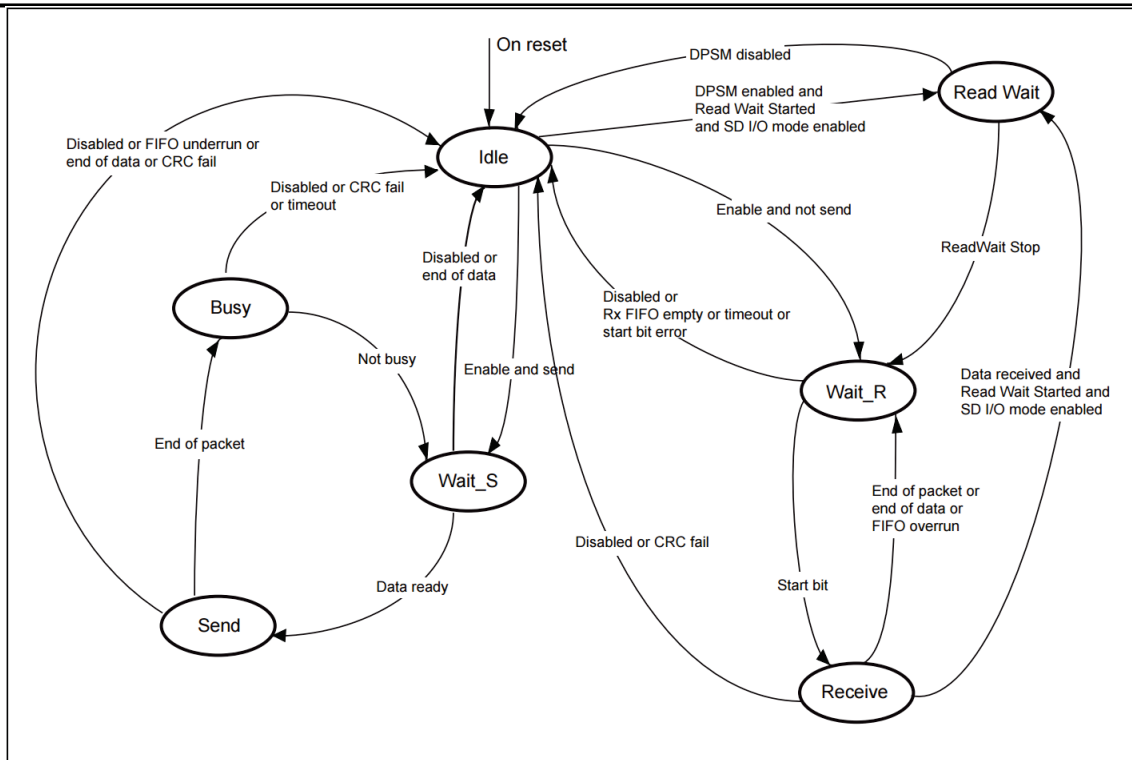


Figure193 Data Channel State Machine (DPSM)

- **Idle:** The data channel is not working and the SDIO\_D[7:0] outputs are in high resistance state. When the data control register is written and the enable bit is set, the DPSM loads a new value for the data counter and enters the Wait\_S or Wait\_R state depending on the data direction bit.
- **Wait\_R:** If the data counter is equal to 0, the DPSM enters into the idle (Idle) state when the receive FIFO is empty. If the data counter is not equal to 0, the DPSM waits for a start bit on SDIO\_D. If the DPSM receives a start bit before the timeout, it enters the Receive state and loads the data block counter. If the DPSM times out before a start bit is detected, or a start bit error occurs, the DPSM enters the idle state and sets the timeout status flag.
- **Receive:** the received serial data is combined into bytes and written to the data FIFO. the data transfer mode can be block or stream depending on the setting of the Transfer Mode bit in the Data Control Register:
  - In block mode, when the data block counter reaches 0, the DPSM waits to receive the CRC code, if the received code matches the internally generated CRC code, the DPSM enters into the Wait\_R state, otherwise the CRC failure status flag is set while the DPSM enters into the idle state.
  - In stream mode, the DPSM receives data when the data counter is not 0. When the counter is 0, it writes the remaining data in the shift register to the data FIFO while the DPSM enters the Wait\_R state. If a FIFO overflow error is generated, the DPSM sets the error flag of the FIFO and enters the Idle state.
- **Wait\_S:** If the data counter is 0, DPSM enters the idle state; otherwise DPSM waits for the data FIFO empty flag to disappear and then enters the transmit state.

**Notes:** The DPSM will remain in the Wait\_S state for at least 2 clock cycles to meet the timing requirements of NWR, which is the interval between the receipt of a response from the card and the start of data transmission by the host.

- **Send:** the DPSM starts sending data to the card device. Depending on the setting of the Transfer Mode bit in the Data Control Register, the data transfer mode can be block transfer or stream transfer:
  - In block mode, when the data block counter reaches 0, the DPSM sends the internally generated CRC code followed by the end bit and enters the busy state.
  - In stream mode, when the enable bit is high while the data counter is not 0, the DPSM sends data to the card device and then enters the idle state.
 If a FIFO underflow error is generated, the DPSM sets the FIFO error flag and enters the idle state.
- **Busy:** DPSM waiting for CRC status flag:

- If the correct CRC status is not received, the DPSM enters the idle state and sets the CRC failure status flag.
- If the correct CRC status is received, the DPSM enters the Wait\_S state when SDIO\_D0 is not low (card is not busy).

When a timeout occurs while the DPSM is busy, the DPSM sets the data timeout flag and enters the idle state.

The data timer is enabled when the DPSM is in the Wait\_R or busy state, the data timer is enabled and can generate a data timeout error:

- When sending data, a timeout is generated if the DPSM is busy for more than the programmed timeout interval.
- When receiving data, if not all data is received and the DPSM is in the Wait\_R state for more than the timeout interval set by the program, a timeout is generated.

- **Data:** Data can be transferred from the host computer to the card or in reverse. Data is transferred on the data line. Data is stored in a 32-word FIFO, each word being 32 bits wide.

Table98 Data Token Format

clarification	starting position	digital	CRC16	end position
block data	0	-	there are	1
streaming data	0	-	not have	1

## Data FIFO

The data FIFO (first-in-first-out) subunit is a data buffer with transmit and receive units.

The FIFO contains a data buffer 32 bits wide per word for a total of 32 words, and transmit and receive circuitry. Because the data FIFO operates in the AHB clock region (HCLK/2), all signals connected to the SDIO clock region (SDIOCLK) are resynchronized.

Depending on the TXACT and RXACT flags, the FIFO can be turned off, enabled to transmit, or enabled to receive. TXACT and RXACT are set by the data channel subunit and are mutually exclusive:

- When TXACT is active, the transmit FIFO represents the transmit circuit and data buffer
- When RXACT is active, the receive FIFO represents the receive circuit and data buffer

- **Transmit FIFO:** When the transmit function of SDIO is enabled, data can be written to the transmit FIFO through the AHB interface.

The transmit FIFO has 32 consecutive addresses. The transmit FIFO has a data output register that contains the data word pointed to by the read pointer. When the data channel subunit is loaded with the shift register, it moves the read pointer to the next data and transfers out the data.

If the transmit FIFO is not enabled, all status flags are inactive. When sending data, the data channel subunit sets TXACT to active.

Table99 Send FIFO Status Flags

symbolize	clarification
TXFIFO	This flag is high when all 32 transmit FIFO words have valid data.
TXFIFOE	This flag is high when all 32 transmit FIFO words have no valid data.
TXFIFOHE	This flag is high when 8 or more transmit FIFO words are empty. This flag can be used as a DMA request.
TXDAVL	This flag is high when the transmit FIFO contains valid data. This flag means just the opposite of TXFIFOE.
TXUNDERR	This flag is high when an underflow error occurs. This flag is cleared when the SDIO clear register is written.

- **Receive FIFO:** When the data channel subunit receives a data word, it writes the data into the FIFO, and the write pointer is automatically added one after the write operation is finished; at the other end, there is a read pointer always pointing to the current data in the FIFO. If the receive FIFO is closed, all status flags are cleared and the read and write pointers are reset. The data channel subunit sets RXACT when data is received. The following table lists the status flags of the receive FIFO. The receive FIFO can be accessed through 32 consecutive addresses.

Table100 Receive FIFO Status Flags

symbolize	clarification
RXFIFO	This flag is high when all 32 receive FIFO words have valid data.
RXFIFOE	This flag is high when all 32 receive FIFO words have no valid data.
RXFIFOHF	This flag is high when 8 or more receive FIFO words have valid data. This flag can be used as a DMA request.
RXDAVL	This flag is high when the receive FIFO contains valid data. This flag means just the opposite of RXFIFOE.

## 23.3.2 SDIO AHB Interface

The AHB interface generates interrupt and DMA requests and accesses the SDIO interface registers and data FIFOs. It contains a data channel, register decoder, and interrupt/DMA control logic.

### SDIO Interrupt

The interrupt control logic generates an interrupt request when at least one of the selected status flags is high. There is a mask register for selecting the conditions under which an interrupt can be generated, and if the corresponding mask flag is set, the corresponding status flag can generate an interrupt.

### SDIO/DMA Interface: The process of transferring data between SDIO and memory

In the following example, the host controller uses CMD24(WRITE\_BLOCK) to transfer 512 bytes from the host to the MMC card, and the DMA controller is used to populate the SDIO's FIFO with data from memory.

1. Executive Card Recognition Process
2. Increase SDIO\_CK frequency
3. Send CMD7 Command Selection Card
4. Configure the DMA2 as follows.
  - a) Enables the DMA2 controller and clears all interrupt flag bits.
  - b) Set the source address register of DMA2 channel 4 to the base address of the memory buffer, and the destination address register of DMA2 channel 4 to the address of the SDIO\_FIFO register
  - c) Set DMA2 channel 4 control registers (memory incremental, non-peripheral incremental, peripheral and source data widths are word widths)
  - d) Enable DMA2 channel 4
5. Send CMD24(WRITE\_BLOCK) with the following operation:
  - a) Set the SDIO data length register (the SDIO data clock register should be set before performing the card recognition process)
  - b) Set the SDIO parameter register to the address in the card to which the data needs to be transmitted
  - c) Set the SDIO command registers: CmdIndex is set to 24 (WRITE\_BLOCK); WaitRest is set to 1 (SDIO card host waits for a response); CPSMEN is set to 1 (enables the SDIO card host to send commands), and keep the other fields as their Reset values.
  - d) Wait for SDIO\_STA[6]=CMDREND interrupt, and then set the SDIO data registers: DTEN is set to 1 (enable the SDIO card host to send data); DTDIR is set to 0 (the controller to the direction of the card); DTMODE is set to 0 (block data transfer); DMAEN is set to 1 (enable DMA); DBLOCKSIZE is set to 9 (512 DBLOCKSIZE is set to 9 (512 bytes); other fields do not need to be set.
  - e) Wait for SDIO\_STA[10]=DBCKEND
6. Query the DMA channel's enable status register to confirm that no channels are still in the enable state.

## 23.4 Card Function Description

### 23.4.1 Card Recognition Mode

In Card Recognition Mode, the host resets all cards, detects the operating voltage range, recognizes the card, and sets the relative address (RCA) for each card on the bus. In Card Recognition Mode, all data communications use only the Command Signal Line (CMD).

### 23.4.2 Card Reset

The GO\_IDLE\_STATE command (CMD0) is a software reset command that places the multimedia card and SD memory in an idle state. The IO\_RW\_DIRECT command (CMD52) resets the SDI/O cards. Upon power-up or execution of CMD0, the outputs of all cards are in a high-resistance state while all cards are initialized to a default relative card address (RCA=0x0001) and default driver register settings (lowest speed, maximum current drive capability).



---

### 23.4.3 Confirmation of Operating Voltage Range

All cards can communicate with the SDIO card host using any voltage within the specified range, and the minimum and maximum voltage VDD values that can be supported are defined by the operating condition register (OCR) on the card.

Cards with card identification numbers (CID) and card specific data (CSD) stored in internal memory can only transmit this information under data transmission VDD conditions. When the VDD ranges of the SDIO card host module and the card do not match, the card will not be able to complete the identification cycle or send CSD data; therefore, in the event of a VDD range mismatch, the SDIO card host can use the following special commands to identify and reject the card: SEND\_OP\_COND (CMD1), SD\_APP\_OP\_COND (SD memory card's ACMD41), and IO\_SEND\_OP\_COND (CMD5 for SDI/O cards). The SDIO card host generates the required VDD voltage when executing these commands. Cards that cannot transfer data within the specified voltage range will be disconnected from the bus and enter an inactive state.

Using these commands, which do not include the voltage range as an operand, the SDIO card host is able to query each card and place cards that are not in this range in an inactive state before determining the common voltage range. The SDIO card host can make such queries when the SDIO card host is able to select a common voltage range or when the user needs to know if the card is functional.

### 23.4.4 Card Recognition Process

There is a difference between the card recognition process for multimedia cards and SD cards; for multimedia cards, the card recognition process starts with the clock frequency F<sub>od</sub>, and all SDIO\_CMD outputs are driven open to allow parallel connection of the cards in the process, and the recognition process is as follows:

1. Bus activated
2. The SDIO card host broadcast sends the SEND\_OP\_COND(CMD1) command and receives the operating conditions
3. The response is a "line with" the contents of the operating condition registers for all cards.
4. Incompatible cards are placed in an inactive state
5. SDIO card host broadcast sends ALL\_SEND\_CID(CMD2) to all activated cards
6. All active cards transmit their CID numbers serially at the same time. Those cards that detect that the output CID bits do not match the data on the command line must stop transmitting and wait for the next identification cycle. Eventually only one card can successfully transmit the complete CID to the SDIO card host and enter the recognition state.
7. The SDIO card host sends the SET\_RELATIVE\_ADDR (CMD3) command to this card, and this new address is called the Relative Card Address (RCA), which is shorter than the CID and is used to address the card. At this point, the card goes into standby and no longer responds to the new identification process, while its output drive changes from open-circuit to push-pull mode.

8. The SDIO card host repeats steps 5 through 7 above until a timeout condition is received. For SD cards, the card recognition process starts at the clock frequency F<sub>od</sub>, and all SDIO\_CMD outputs are push-pull driven instead of open-circuit driven, and the recognition process is as follows:

1. Bus activated
2. The SDIO card host broadcast sends the SEND\_APP\_OP\_COND (ACMD41) command
3. The response obtained is the contents of the operating condition registers for all cards
4. Incompatible cards are placed in an inactive state
5. SDIO card host broadcast sends ALL\_SEND\_CID(CMD2) to all activated cards
6. All activated cards send back their unique card identification number (CID) and enter an identification state.
7. The SDIO card host sends the SET\_RELATIVE\_ADDR (CMD3) command and an address to an activated card. This new address is called the Relative Card Address (RCA), which is shorter than the CID and is used to address the card. At this point, this card goes to standby. the SDIO card host can send this command again to change the RCA, and the card's RCA will be the last assignment.
8. The SDIO card host repeats steps 5 through 7 above for all activated cards.

For SDI/O cards, the card recognition process is as follows:

1. Bus activated
2. SDIO card host sends IO\_SEND\_OP\_COND (CMD5) command

3. The response you get is the contents of the card's operating condition registers
4. Incompatible cards are placed in an inactive state
5. The SDIO card host sends the SET\_RELATIVE\_ADDR (CMD3) command and an address to an activated card. This new address is called the Relative Card Address (RCA), which is shorter than the CID and is used to address the card. At this point, this card goes to standby. the SDIO card host can send this command again to change the RCA, and the card's RCA will be the last assignment.

### 23.4.5 Write Data Block

When the Write Data Block command (CMD24-27) is executed, the host transmits one or more blocks of data from the host to the card, along with a CRC code at the end of each block. A card that supports the Write Data Block command should always be able to receive data blocks as defined by WRITE\_BL\_LEN. If the CRC checksum is incorrect, the card indicates the error via the SDIO\_D signal line, the transmitted data is discarded without being written, and all subsequent (in multi-block write mode) transmitted data blocks are ignored.

If the host transmits partial data and the cumulative data length is not aligned with the data block, when block Offset is not allowed (parameter WRITE\_BLK\_MISALIGN of the CSD is not set), the card will detect a block Offset error before the first misaligned block (ADDRESS\_ERROR error bit in the status register is set). The write operation is also aborted when the host attempts to write a write-protected region, at which point the card sets the WP\_VIOLATION bit. Setting the CID and CSD registers does not require the block length to be set in advance, and the transmitted data is CRC-protected. If a portion of the CSD or CID register is stored in ROM, this unchangeable portion must match the corresponding portion of the receive buffer, and if there is an inconsistency, the card will report an error without modifying the contents of any of the registers. Some cards take a long or even unpredictable time to finish writing a block of data. After receiving a block of data and completing the CRC check, the card starts the write operation and pulls the SDIO\_D signal line low if its write buffer is full and it can no longer accept new data from a new WRITE\_BLOCK command. The host can query the status of the card at any time using SEND\_STATUS(CMD13), and the card will return the current status. The READY\_FOR\_DATA status bit indicates whether the card is ready to accept new data or whether a write operation is still in progress. The host can use CMD7 (Select Another Card) to unselect a card and place that card in the disconnected state, which releases the SDIO\_D signal line without interrupting an incomplete write operation; when a card is reselected, it will again indicate the busy state by pulling the SDIO\_D signal line low if a write operation is still in progress and the write buffer is still not available.

### 23.4.6 Read Data Block

In read block mode, the basic unit of data transfer is the data block, whose size is defined in the CSD (READ\_BL\_LEN). If READ\_BL\_PARTIAL is set, it is also possible to transfer smaller data blocks. A smaller data block is one in which the start and end addresses are completely contained in a single physical block, and READ\_BL\_LEN defines the size of the physical block. To ensure that the data is transmitted correctly, each data block is followed by a CRC checksum. CMD17(READ\_SINGLE\_BLOCK) initiates a read data block operation, and the card returns to the transmit state at the end of the transmission. CMD17(READ\_SINGLE\_BLOCK) initiates a read data block operation, and the card returns to the transmit state at the end of the transmission.

CMD18(READ\_MULTIPLE\_BLOCK) initiates a read operation of multiple consecutive data blocks. The host can abort a multi-block read operation at any time, regardless of the type of operation. The operation can be aborted by sending a stop transfer command.

If the card detects an error (e.g., out-of-bounds, address mismatch, or internal error) during a multiple-block read operation (of any type), it stops the data transfer and remains in the data state; at this point, the host must abort the operation by sending a stop-transfer command. A read error is reported in the response to the stop transfer command.

If the host sends a stop-transmission command when the card has transmitted the last data block of a defined number of multiple data block operations, because the card is no longer in the data state at this point, the host gets an illegal command in response. If the host transmits a partial block of data and the cumulative data length cannot be aligned with the physical block while not allowing block Offset, the card detects a block alignment error at the occurrence of the first unaligned block and sets the ADDRESS\_ERROR error flag in the status register.



## 23.4.7 Data Stream Operations, Data Stream Writes and Data Stream Reads (Multimedia Cards Only)

In data streaming mode, data is transferred byte by byte, while there is no CRC after each data block.

Data Stream Write (multimedia cards only)

WRITE\_DAT\_UNTIL\_STOP (CMD20) starts data transfer from the SDIO card host to the card, starting at the specified address and continuing until the SDIO card host issues a stop command. If partial block transfers are allowed (with the CSD parameter WRITE\_BL\_PARTIAL set), the data stream can start and stop at any address in the card's address space; otherwise, the data stream can only start and stop at the boundaries of the data block. Because the number of data to be transmitted is not set in advance, CRC checksums cannot be used. If the maximum address of the memory is reached when sending data, the subsequently transmitted data will be discarded even if the SDIO card host does not send a stop command.

The maximum clock frequency for a data stream write operation can be calculated by the following equation

$$\text{Maximumspeed} = \text{Min}(\text{TRANSPEED}, \frac{(8 * 2^{\text{writeblen}})(-NSAC)}{\text{TAAC} * \text{R2WFACTOR}})$$

- Maximumspeed=Maximum write frequency
- TRANSPEED = maximum data transfer rate
- writeblen=Maximum write block length
- NSAC = data read operation time in CLK cycles2
- TAAC = Data Read Operation Time 1
- R2WFACTOR = write speed factor

If the host attempts to use a higher frequency, the card may not be able to process the data and stops programming while the OVERRUN error bit is set in the status register, discarding all subsequently transmitted data and (in the Received Data state) waiting for a stop command. If the host attempts to write to a write-protected area, the write operation will be aborted while the card sets the WP\_VIOLATION bit.

### Data Streaming Read (MultiMediaCard only)

READ\_DAT\_UNTIL\_STOP (CMD11) controls the data stream data transfer.

This command requires the card to read data from the specified address until the SDIO card host sends STOP\_TRANSMISSION(CMD12). Because of the delay in serial command transmission, there is a delay in the execution of the stop command, and data transfer stops after the end bit of the stop command. If the maximum address of the memory is reached when data is sent and the SDIO card host does not send a stop command, the subsequently transmitted data will be invalid data.

The maximum clock frequency for a data stream read operation can be calculated by the following equation

$$\text{Maximumspeed} = \text{Min}(\text{TRANSPEED}, \frac{n(8 * 2^{\text{readblen}})(-NSAC)}{\text{TAAC} * \text{R2WFACTOR}})$$

- Maximumspeed=Maximum write frequency
- TRANSPEED = maximum data transfer rate
- readblen=Maximum read block length
- NSAC = data read operation time in CLK cycles2
- TAAC = Data Read Operation Time 1
- R2WFACTOR = write speed factor

If the host attempts to use a higher frequency, the card will not be able to process the data transfer, at which point the card sets the UNDERRUN error bit in the status register, aborts the data transfer and waits for a stop command in the data state.

## 23.4.8 Erase: Group Erase and Sector Erase

The erasure unit of a multimedia card is the erase group, which is calculated in write data blocks, the basic writing unit of the card. The size of the erase group is a card-specific parameter, defined in the CSD.

The host can erase a contiguous range of erase groups, and there are three steps to begin an erase operation.

First, the host defines the start address of the continuous range using the ERASE\_GROUP\_START (CMD35) command, then defines the end address of the continuous range using the ERASE\_GROUP\_END (CMD36) command, and finally sends the erase command, ERASE (CMD38), to start the erase operation. The address field of the erase command is the erase group address in bytes. The card discards the portion that is not aligned to the size of the erase group and aligns the address boundary to the boundary of the erase group.

If an erase command is not received as described above, the card sets the ERASE\_SEQ\_ERROR bit in the status register and waits for the first step again.

If a command other than the SEND\_STATUS and erase commands are received, the card sets the ERASE\_RESET bit in the status register to release the erase sequence and execute the new command.

If the erase range contains write-protected data blocks, these blocks are not erased and only the unprotected blocks are erased while the card sets the WP\_ERASE\_SKIP status bit in the status register.

Card low SDIO\_D signal during the erase process. The actual erase time can be long and the host can use CMD7 to unselect the card.

## 23.4.9 Wide Bus Selection and Deselection

The wide bus (4-bit bus width) operation mode can be selected or not with the SET\_BUS\_WIDTH (ACMD6) command, and the default bus width is 1-bit after power-up or the GO\_IDLE\_STATE (CMD0) command. The SET\_BUS\_WIDTH (ACMD6) command is valid only in the transmission state, i.e., the bus width can be changed only after the card is selected with the SELECT/ DESELECT\_CARD (CMD7) command to change the bus width after the card has been selected.

## 23.4.10 Conservation Management

The SDIO card host module supports three protection methods:

1. Internal card protection (card management)
2. Mechanical write-protect switch (managed by SDIO card host module only)
3. Password-managed card lock operation

### Internal Card Write Protection

Card data can be protected from being overwritten or erased. By setting the write-protect bit in the CSD permanently or temporarily, the manufacturer or content provider can permanently write-protect the entire card. For cards that support a set of sectors to be write-protected by setting the WP\_GRP\_ENABLE bit in the CSD, a portion of the data can be protected and the write-protection can be programmatically changed. The basic unit of write protection is the CSD parameter WP\_GRP\_SIZE sectors. the SET\_WRITE\_PROT and CLR\_WRITE\_PROT commands control the protection of the specified group. the SEND\_WRITE\_PROT command is similar to the single block read command in that the card sends a block of data containing 32 write-protected bits (representing 32 write-protected groups starting from a specified address), followed by a 16-bit block of data containing 32 write-protected bits (representing 32 write-protected groups starting from a specified address). The card sends a block of data containing 32 write-protect bits (representing 32 write-protect groups starting at the specified address) followed by a 16-bit CRC code. The address field of the write-protect command is a group address in bytes. card will truncate all addresses below the group size.

### Mechanical Write Protect Switch

On the side of the card is a mechanical slide switch that allows the user to set or clear the card's write protection. When the slide switch is placed in the small open window position, the card is write-protected, and when the slide switch is placed in the small closed window position, the contents of the card can be changed. There is also a switch on the corresponding part of

the card's slot that instructs the SDIO card host module whether the card is write-protected or not. The internal circuitry of the card is unaware of the position of the write-protect switch.

### Password Protection

The password protection feature allows the SDIO card host module to use a password to apply a lock or unlock to the card. The password is stored in the 128-bit PWD register and its length is set in the 8-bit PWD\_LEN register. These registers are non-volatile, i.e., their contents are not lost when power is lost. A locked card is able to respond to and execute the appropriate commands, i.e., it allows the SDIO card host module to perform operations such as reset, initialization, and querying the status, but it does not allow manipulation of the data in the card. When a password is set (i.e., the value of PWD\_LEN is not 0), the card is automatically in the locked state after power-up. As with the CSD and CID register write commands, the Lock/Unlock command is only valid in the Transmit state, where there is no address parameter in the command, but the card has been selected. The Card Lock/Unlock command has the structure and bus operation type of a single data block write command, where the transmitted data block contains all the information required for the command (password setting mode, PWD contents, and lock/unlock indication). The length of the command data block is defined by the SDIO card host module before the card lock/unlock command is sent, and the command structure is shown in

Table 114.

The bits are set as follows:

- ERASE: Setting this bit will perform a forced erase, all other bits must be 0. Only the command byte is sent.
- LOCK\_UNLOCK: Set this bit to lock the card, LOCK\_UNLOCK and SET\_PWD can be set at the same time, but not at the same time as CLR\_PWD.
- CLR\_PWD: Set this bit to clear the password data.
- SET\_PWD: Set this bit to save the password data to memory.
- PWD\_LEN: Defines the length of the password in bytes.
- PWD: password (new or in use, depending on the command)

The following sections list the command sequences for setting/clearing passwords, locking/unlocking, and force erasing.

### Setting a Password

1. Select a card (SELECT/DESELECT\_CARD, CMD7).
2. Define the length of the data block to be sent in 8-bit card lock/unlock mode (SET\_BLOCKLEN, CMD16), 8-bit PWD\_LEN, the number of bytes of the new password. When the password is changed, the length of the data block for sending commands must take into account both the length of the old and new passwords.
3. A LOCK/UNLOCK (CMD42) command is sent on the data line in the appropriate data block length and contains a 16-bit CRC code. The data block contains the mode of operation (SET\_PWD=1), length (PWD\_LEN), and password (PWD). When a password is changed, the length value (PWD\_LEN) contains the lengths of both the old and new passwords, and the PWD field contains the old password (the one being used) and the new password.
4. When the old password matches, the new password and its length are stored in the PWD and PWD\_LEN fields, respectively. If the old password sent does not match the desired password (length or content), the LOCK\_UNLOCK\_FAILED error bit in the status register is set while the password remains unchanged.

The password length field (PWD\_LEN) indicates whether or not a password is currently set; if the field is non-zero, a password is used and the card is automatically locked at power-up. Without powering up, if a PIN is set, the card can be locked immediately by setting the LOCK\_UNLOCK bit or sending an additional lock command.

### Clear Password

1. Select a card (SELECT/DESELECT\_CARD, CMD7).
2. Defines the length of the data block to be sent in 8-bit card lock/unlock mode (SET\_BLOCKLEN, CMD16), 8-bit PWD\_LEN, the number of bytes of the currently used password.
3. When the password matches, the PWD field is cleared and PWD\_LEN is set to 0. If the delivered password does not match the desired password (length or content), the LOCK\_UNLOCK\_FAILED error bit in the status register is set while the password remains unchanged.

### Snap-Lock

1. Select a card (SELECT/DESELECT\_CARD, CMD7)  
Defines the length of the block of data (SET\_BLOCKLEN, CMD16) to be sent in 8-bit card lock/unlock mode (see byte 0 of the
2. Table114 ), the 8-bit PWD\_LEN, and the number of bytes of the current password.
3. A LOCK/UNLOCK (CMD42) command is sent on the data line in the appropriate data block length and contains a 16-bit CRC code. The data block contains the mode of operation (LOCK\_UNLOCK=1), length (PWD\_LEN), and password (PWD).
4. When the password matches, the card is locked and the CARD\_IS\_LOCKED status bit in the status register is set. If the sent PIN does not match the desired PIN (length or content), the LOCK\_UNLOCK\_FAILED error bit in the status register is set and the locking operation fails.

Setting the password and locking the card can be done in the same sequence of operations, in which case the SDIO card host module sets the password as described in the previous steps, but the LOCK\_UNLOCK bit needs to be set in step 3 of the Send New Password command.

If a password has ever been set (PWD\_LEN is not 0), the card is automatically unlocked at power-on reset. Performing a lock operation on a card that is already locked or on a card that does not have a PIN will result in a failure and set the LOCK\_UNLOCK\_FAILED error bit in the status register.

### Card Unlocking

1. Select a card (SELECT/DESELECT\_CARD, CMD7)  
Defines the length of the block of data (SET\_BLOCKLEN, CMD16) to be sent in 8-bit card lock/unlock mode (see byte 0 of the
2. Table114 ), the 8-bit PWD\_LEN, and the number of bytes of the current password.
3. A LOCK/UNLOCK (CMD42) command is sent on the data line in the appropriate data block length and contains a 16-bit CRC code. The data block contains the mode of operation (LOCK\_UNLOCK=0), length (PWD\_LEN), and password (PWD).
4. When the passwords match, the card lock is released and the CARD\_IS\_LOCKED bit in the status register is cleared. If the delivered PIN does not match the desired PIN (length or content), the LOCK\_UNLOCK\_FAILED error bit in the status register is set while the card remains locked.

The unlocked state is only valid during the current power-up, and as long as the PWD field is not cleared, the card will be automatically locked after the next power-up. Attempting to perform an unlock operation on a card that is already unlocked will cause the operation to fail and set the LOCK\_UNLOCK\_FAILED error bit in the status register.

### Forced Erase

If the user forgets the password (the contents of the PWD), the card can be used after erasing all the contents of the card. The Force Erase operation erases all data and passwords from the card.

1. Select a card (SELECT/DESELECT\_CARD, CMD7)  
Set the block length sent (SET\_BLOCKLEN, CMD16) to 1. Only the 8-bit card lock/unlock byte is sent (see byte 0 of
2. Table114 ).
3. A LOCK/UNLOCK (CMD42) command is sent on the data line in the appropriate data block length and contains a 16-bit CRC code. The data block contains the mode of operation (ERASE=1) all other bits are zeros.
4. When the ERASE bit is the only bit in the data field, everything in the card is erased, including the PWD and PWD\_LEN fields, while the card is no longer locked. If any other bit is not 0, the LOCK\_UNLOCK\_FAILED error bit in the status register is set and the data in the card remains unchanged while the card remains locked.

Attempting to perform an erase operation on a card that has been unlocked causes the operation to fail and sets the LOCK\_UNLOCK\_FAILED error bit in the status register.

## 23.4.11 Card Status Register

The response format R1 contains a 32-bit card status field that is used to send card status information (which could potentially reside in a local status register) to the card host. Unless otherwise specified, the status returned by the card is always associated with the previous command.

Table101 defines the different status messages. The abbreviations in the table for the type and clear condition fields are defined as follows: type:

- E:Error bit
- S:Status bit
- R: Detection bit and set according to the actual command response
- X:Detect bit, set in the execution of the command. the SDIO card host queries the status of the card by sending a status command to read out these bits.

Clearance conditions:

- A: Based on the current status of the card
- B: Always associated with a previous command. It is cleared by receiving the correct command (with a command delay).
- C: Read to clear

Table101 Card Status

Bit	name (of a thing)	typology	numerical value	clarification	Removal conditions
31	ADDRESS_OUT_OF_RANGE	ERX	'0' = no error '1' = error	The address parameter in the command is outside the allowed range of the card. A multi-block or stream read/write operation (even if starting from a legal address) attempts to read or write more than the capacity of the card.	C
30	ADDRESS_MISALIGN		'0' = no error '1' = error	The first data block defined by the address parameter in the command (against the current block length) is not aligned with the physical block of the card. A multi-block or stream read/write operation (even if starting from a legal address) attempts to read or write a block of data that is not aligned with a physical block.	C
29	BLOCK_LEN_ERROR		'0' = no error '1' = error	The parameter of the SET_BLOCKLEN command is outside the maximum allowable range of the card, or a previously defined block length is illegal for the current command (e.g., the host sends a write command where the current block length is less than the minimum allowed by the card, and at the same time does not allow a partial block to be written).	C
28	ERASE_SEQ_ERROR		'0' = no error '1' = error	Erase commands were sent in the wrong order.	C
27	ERASE_PARAM	EX	'0' = no error '1' = error	An illegal erase group was selected for the erase.	C
26	WP_VIOLATION	EX	'0' = no error '1' = error	Attempts to program a write-protected data block.	C
25	CARD_IS_LOCKED	SR	'0' = card unlocked '1' = card locked	When this bit is set, it indicates that the card is locked.	A
24	LOCK_UNLOCK_FAILED	EX	'0' = no error '1' = error	There is an error in the sequence of commands in locking/unlocking or an incorrect password has been detected.	C
23	COM_CRC_ERROR	ER	'0' = no error '1' = error	CRC checksum error in previous command.	B
22	ILLEGAL_COMMAND	ER	'0' = no error '1' = error	For the current card status, the command is illegal.	B
21	CARD_ECC_FAILED	EX	'0' = Success '1' = Failure	The ECC checksum is implemented internally on the card, but fails when correcting the data.	C
20	CC_ERROR	ER	'0' = no error '1' = error	(Undefined in the standard) An error occurred within the card, unrelated to commands from the host.	C

19	ERROR	EX	'0' = no error '1' = error	A (undefined in the standard) card-internal error (e.g., read or write error) related to the execution of a previous host command was generated.	C
18	Reserved				
17	Reserved				
16	CID/CSD_OVERWRITE	EX	'0' = no error '1' = error	It can be any of the errors described below: n has been written to the CID register and cannot be overwritten The read-only portion of the nCSD does not match the contents of the card nAttempts to perform a copy or permanent write-protected reverse operation, i.e., to restore the original state or to remove write protection.	C
15	WP_ERASE_SKIP	EX	'0' = not protected '1' = protected	Only a portion of the address space is erased when a write-protected data block already exists is encountered.	C
14	CARD_ECC_DISABLED	SX	'0' = allowed '1' = not allowed	No internal ECC is used to execute the command.	A
13	ERASE_RESET		'0' = clear '1' = set	The sequence into the erase process is aborted because of the receipt of a command outside of the erase sequence (not a CMD35, CMD36, CMD38, or CMD13 command).	C
12: 9	CURRENT_STATE	SR	0=Idle 1=Ready 2 = Identification 3 = Standby 4=Send 5 = Data 6 = Receiving 7 = Programming 8 = Disconnect 9=Busy test 10 to 15 = reserved	The state of the card's state machine when a command is received. If the execution of a command results in a change of state, this change will be reflected in the response to the next command. These four bits are interpreted as decimal numbers 0 through 15.	B
8	READY_FOR_DATA	SR	'0' = not ready '1' = ready	Corresponds to the signal that the buffer on the bus is empty.	
7	SWITCH_ERROR	EX	'0' = no error '1' = conversion error	The card did not convert to the desired mode as required by the SWITCH command.	B
6	Reserved				
5	APP_CMD	SR	'0' = not allowed '1' = allowed	The card expects ACMD, or indicates that the command has been interpreted as an ACMD command.	C
4	Reserved for SDI/O cards				
3	AKE_SEQ_ERROR	ER	'0' = no error '1' = error	There is an error in the order of validation.	C
2	Reserved for application-related commands.				
1,0	A test pattern reserved for manufacturers.				

## 23.4.12 SD Status Register

The SD status contains status bits related to specific functions of the SD memory card and some status bits related to future applications. The length of the SD status is a 512-bit block of data. Upon receipt of an ACMD13 command (CMD55, then CMD13), the contents of this register are transmitted to the SDIO card host. The ACMD13 command can only be sent when the card is in the transmit state (the card has been selected).

Table 102 defines the different SD status register information. The abbreviations in the table regarding the type and clear condition fields are defined as follows: type:

- E: Error bit
- S: Status bit
- R: Detection bit and set according to the actual command response



- X: Detect bit, set in the execution of the command. the SDIO card host queries the status of the card by sending a status command to read out these bits.

Clearance conditions:

- A: Based on the current status of the card
- B: Always associated with a previous command. It is cleared by receiving the correct command (with a command delay).
- C: Read to clear

Table102 SD Status

bit	name (of a thing)	typology	numerical value	clarification	Removal conditions
511:510	DAT_BUS_WIDTH	SR	'00'=1 (default) '01' = Reserved '10' = 4 bits wide '11' = reserved	The current data bus width defined by the SET_BUS_WIDTH command.	A
509	SECURED_MODE	SR	'0' = not in confidential mode '1' = in confidential mode	The card is in the confidential mode of operation (see "SD Confidentiality Specification" for details).	A
508:496	Reserved				
495:480	SD_CARD_TYPE	SR	'00xxh' = SD memory card in physical specification version 1.01~2.00 ('x' means any value). Defined cards are: '0000'=Universal SD Read/Write Card '0001'=SDROM card	The lower 8 bits of this field can be used in the future to define different variants of SD memory cards (each bit can be used to define a different SD type). The high 8 bits can be used to define SD cards that do not adhere to the current SD physical layer specification.	A
479:448	SIZE_OF_PROTECTED_AREA	SR	Size of protected area (see description below)	(see note below)	A
447:440	SPEED_CLASS	SR	Type of card speed (see description below)	(see note below)	A
439:432	PERFORMANCE_MOVE	SR	Transfer performance in 1MB/sec (see description below)	(see note below)	A
431:428	AU_SIZE	SR	Size of AU (see note below)	(see note below)	A
427:424	Reserved				
423:408	ERASE_SIZE	SR	Number of AUs that can be erased at one time	(see note below)	A
407:402	ERASE_TIMEOUT	SR	Erase the timeout value for the range specified by UNIT_OF_ERASE_AU	(see note below)	A
401:400	ERASE_OFFSET	SR	Fixed offset value added during erasure	(see note below)	A
399:312	Reserved				
311:0	Reserved for producers				

### SIZE\_OF\_PROTECTED\_AREA

Standard capacity cards and high capacity cards set this bit differently. For standard capacity cards, the capacity of the protected area is calculated by the following formula:

Protected Area = SIZE\_OF\_PROTECTED\_AREA \* MULT \* BLOCK\_LEN The unit of SIZE\_OF\_PROTECTED\_AREA is MULT \* BLOCK\_LEN.

For high-capacity cards, the capacity of the protected area is calculated by the following formula:

The unit of protected\_area = SIZE\_OF\_PROTECTED\_AREA \* SIZE\_OF\_PROTECTED\_AREA is bytes.

## SPEED\_CLASS

These 8 bits indicate the type of speed and can be calculated by calculating the value of PW/2 (PW is the performance of the write).

Table103 Speed Type Codes

SPEED_CLASS	numerical definition
00h	Type 0
01h	Type 2
02h	Type 4
03h	Type 6
04h-FFh	Reserved

## PERFORMANCE\_MOVE

These 8 bits indicate the movement performance (Pm) in units of 1MB/sec. If the card does not move data in RUs (Record Units), Pm should be considered infinite. Setting this field to FFh indicates infinity.

Table104 Mobile Performance Codes

PERFORMANCE_MOVE	numerical definition
00h	undefined
01h	1MB/sec
02h	2MB/sec
.....	.....
FEh	254MB/s
FFh	boundless

## AU\_SIZE

These 4 bits indicate the length of the AU, and the value is a multiple of 16K bytes in units of powers of two.

Table105 AU\_SIZE codes

AU_SIZE	numerical definition
00h	undefined
01h	16KB
02h	32KB
03h	64KB
04h	128KB
05h	256KB
06h	512KB
07h	1MB
08h	2MB
09h	4MB
Ah-Fh	Reserved

Depending on the capacity of the card, the maximum AU length is defined in the table below. The card can set any AU length between the RU length and the maximum AU length.

Table106 Maximum AU length

quantitative (science)	16MB-64MB	128MB-256MB	512MB	1GB-32GB
Maximum AU length	512KB	1MB	2MB	4MB

## ERASE\_SIZE

This 16-bit field gives the NERASE, and ERASE\_TIMEOUT defines the timeout when NERASE AUs are erased. The host should determine the appropriate number of AUs to be erased in a single operation so that the host can display the progress of the erase operation. If this field is 0, timeout calculations for erases are not supported.



Table107 ERASE\_SIZE codes

ERASE_SIZE	numerical definition
0000h	Timeout calculation for erasure is not supported
0001h	1 AU
0002h	2 AU
0003h	3 AU
.....	.....
FFFFh	65535 AU

#### ERASE\_TIMEOUT

These 6 bits give the TERASE, and when multiple AUs indicated by ERASE\_SIZE are erased, this value gives the erase timeout counted from the offset. The range of ERASE\_TIMEOUT can be defined up to 63 seconds, and the card manufacturer can choose the appropriate combination of ERASE\_SIZE and ERASE\_TIMEOUT for the specific implementation by first Determine ERASE\_TIMEOUT and then ERASE\_SIZE.

Table108 Erase Timeout Codes

ERASE_TIMEOUT	numerical definition
00	Timeout calculation for erasure is not supported
01	1 second
02	2 seconds.
03	3 seconds.
.....	.....
63	63 seconds.

#### ERASE\_OFFSET

These 2 bits give the TOFFSET, and this value is meaningless when ERASE\_SIZE and ERASE\_TIMEOUT are both 0.

Table109 Erase Offset Codes

ERASE_OFFSET	numerical definition
0h	0 seconds
1h	1 second
2h	2 seconds.
3h	3 seconds.

### 23.4.13 I/O Mode of SD

#### I/O Interrupt for SD

In order for the SDI/O card to interrupt the multimedia card/SD module, there is a pin on the SD interface with an interrupt function - pin 8. In 4-bit SD mode this pin is SDIO\_D1, which is used by the card to request an interrupt from the multimedia card/SD module. The interrupt function is optional for each card or function within the card. interrupts for SDI/O are level active, i.e., the interrupt signal line must remain active (low) until recognized and responded to by the MultiMediaCard/SD Module, and invalid (high) after the interrupt process is complete. After the MultiMediaCard/SD module has serviced the interrupt request, the interrupt status bits are cleared by an I/O write operation that writes the appropriate bits to the SDI/O card's internal registers. The interrupt outputs of all SDI/O cards are active low and the Multimedia Card/SD module provides pull-up resistors on all data lines (SDIO/D[3:0]). The MultiMediaCard/SD module samples pin 8 (SDIO\_D/IRQ) during the interrupt phase and performs interrupt detection; values on this signal line are ignored at all other times.

Both memory operations and I/O operations have interrupt phases, and the definition of an interrupt phase for a single data block operation is different from the definition of an interrupt phase for a multiple data block transfer operation.

#### I/O Suspend and Resume for SD

In a multi-function SDI/O card or a card with both I/O and memory functions, multiple devices (I/O and memory) share the MMC/SD bus. In order to enable multiple devices in the MMC/SD

module to share the bus, SDI/O cards and composite cards may optionally implement a suspend/resume concept; if a card supports suspend/resume, the MMC/SD module is able to temporarily stop a data transfer operation (pause) for one function or memory, thereby yielding the bus to other functions or memories with higher priority, and after this higher-priority transfer is completed, then the MMC/SD module is able to temporarily stop the data transfer operation (pause) for one function or memory. After the higher-priority transfer is completed, the originally paused transfer is resumed. Support for suspend/resume operation is optional. To perform a pause/resume operation on the MMC/SD bus, proceed as follows:

1. Determine the current function of the SDIO\_D[3:0] signal lines
2. Requesting a low priority or slow operation pause
3. Wait for the suspend operation to complete and confirm that the device is suspended
4. Starting high-priority transmission
5. Wait for high priority transmission to finish
6. Resuming suspended operations

#### **SDI/O Read Wait**

The optional Read Wait (RW) operation is only available for SD cards in 1-bit or 4-bit mode. The read-wait operation allows the MMC/SD module to ask a card to temporarily stop data transfer while it is reading multiple registers (IO\_RW\_EXTENDED, CMD53), while allowing the MMC/SD module to send commands to other functions in the SDI/O device. To determine whether a card supports the read wait protocol, the MMC/SD module should detect the internal registers of the card. The read wait time is related to the interrupt phase.

## **23.4.14 Command and Response**

### **Application-Related and Generic Commands**

The SD Card Host Module system is used to provide a standardized interface that is suitable for a wide range of application types, but at the same time takes into account the functionality of specific users and applications, so two types of generic commands are defined in the standard: application-related commands (ACMD) and generic commands (GEN\_CMD).

When the card receives an APP\_CMD(CMD55) command, the card expects the next command to be an Application Related Command. An application-related command (ACMD) has the same format structure as a normal multimedia card and can use the same CMD number, and because it appears after the APP\_CMD(CMD55), the card recognizes it as an ACMD command. If following APP\_CMD(CMD55) is not a defined application-related command, it is considered to be a standard command; e.g., there is an SD\_STATUS(ACMD13) application-related command that would be interpreted as SD\_STATUS(ACMD13) if it received CMD13 immediately after APP\_CMD(CMD55); but if the card receives a CMD7 immediately after APP\_CMD(CMD55) and this card does not define ACMD7, it will be interpreted as a standard CMD7(SELECT/DESELECT\_CARD) command.

To use the manufacturer's customized ACMD, the SD card host needs to do the following:

1. Send APP\_CMD(CMD55) command  
The card sends back a response to the multimedia/SD card module indicating that the APP\_CMD bit is set and waiting for an ACMD command.
2. Send the specified ACMD

The card sends back a response to the multimedia/SD card module indicating that the APP\_CMD bit is set and that the received command has been correctly parsed according to the ACMD commands; if a non-ACMD command is sent, the card will follow normal multimedia card command processing while clears the APP\_CMD bit in the card's status register.

If an illegal command is sent (whether ACMD or CMD), it will be handled incorrectly as a standard illegal multimedia card command.

The bus operation procedure of the GEN\_CMD command is the same as that of the single data block read/write commands (WRITE\_BLOCK, CMD24 or READ\_SINGLE\_BLOCK, CMD17); in this case, the parameter of the command indicates the direction of the data transfer instead of the address, and the data block has a user-defined format and meaning.

Before sending the GEN\_CMD(CMD56) command, the card must be selected (the state machine is in the transmit state), and the length of the data block is defined by SET\_BLOCKLEN(CMD16). The response to the GEN\_CMD(CMD56) command is in R1b format.

### **Command Type**

There are four different types of application-related and generic commands:

1. Broadcast Command (BC): sent to all cards, no response returned.
2. Broadcast Command with Response (BCR): sends to all cards and receives responses back from all cards.
3. Command with addressing (point-to-point) (AC): sent to the selected card, data transfer not included on the SDIO\_D signal line.
4. Data Transfer Command with Addressing (Point-to-Point) (AC): sent to the selected card, contains the data transfer on the SDIO\_D signal line.

### Command Format

See Table94 for the command format.

### Commands for MultiMediaCard/SD Card Module

Table110 Block Transfer Based Write Commands

CMD Index	typology	parameters	response format	abridge	clarification
CMD23	ac	[31:16]=0 [15:0]=Number of data blocks	R1	SET_BLOCK_COUNT	Defines the number of blocks to be transferred in a subsequent multi-block read or write command.
CMD24	adtc	[31:0]=Data address	R1	WRITE_BLOCK	Write a block of the length selected by the SET_BLOCKLEN command.
CMD25	adtc	[31:0]=Data address	R1	WRITE_MULTIPLE_BLOCK	Write data blocks consecutively until a STOP_TRANSMISSION command is received or the specified number of blocks is reached.
CMD26	adtc	[31:0]=fill bits	R1	PROGRAM_CID	Programs the card's identification register. This command can only be sent once for each card. There is a hardware mechanism in the card that prevents multiple programming operations. Normally this command is reserved for the manufacturer.
CMD27	adtc	[31:0]=Fill Bits	R1	PROGRAM_CSD	Program the bits programmable in the card's CSD.
CMD28	ac	[31:0]=Data address	R1b	SET_WRITE_PROT	If the card has a write-protect feature, this command sets the write-protect bit for the specified group. The write-protect feature is set in the card's special data area (WP_GRP_SIZE).
CMD29	ac	[31:0]=Data address	R1b	CLR_WRITE_PROT	If the card is write-protected, this command clears the write-protect bit for the specified group.
CMD30	adtc	[31:0]=Write-protect data address	R1	SEND_WRITE_PROT	If the card is write-protected, this command asks the card to send the status of the write-protect bit.
CMD31	Reserved				

Table111 Block Transfer Based Write Protect Commands

CMD Index	typology	parameters	response format	abridge	clarification
CMD28	c	[31:0]=Data address	R1b	SET_WRITE_PROT	If the card is write-protected, this command sets the write-protect bit for the specified group. The write-protect attribute is set in the card-specific data field (WP_GRP_SIZE).
CMD29	c	[31:0]=Data address	R1b	CLR_WRITE_PROT	If the card is write-protected, this command clears the write-protect bit for the specified group.
CMD30	dte	[31:0]=Write-protect data address	R1	SEND_WRITE_PROT	If the card is write-protected, this command asks the card to send the status of the write-protect bit.
CMD31	Reserved				

Table112 Erase Commands

CMD Index	typology	parameters	response format	abridge	clarification
CMD32 ... CMD34	Reserved. For backward compatibility with older versions of the pair media card protocol, these command codes cannot be used.				
CMD35	ac	[31:0]=Data address	R1	ERASE_GROUP_START	Set the address of the first erase group within the selected erase range.
CMD36	ac	[31:0]=Data address	R1	ERASE_GROUP_END	Set the address of the last erase group within the selected continuous erase range.
CMD37	Reserved. For backward compatibility with older versions of the pair media card protocol, this command code cannot be used.				
CMD38	ac	[31:0]=fill bits	R1	ERASE	Erases the previously selected data block.

Table113 I/O Mode Commands

CMD Index	typology	parameters	response format	abridge	clarification
CMD39	ac	[31:16]=RCA [15]=Register write flags [14:8]=Register address [7:0]=Register data	R4	FAST_IO	Used to write and read 8-bit (register) data fields. This command specifies a card and register, and also provides the data written if the write flag is set. The R4 response contains the data read from the specified register. This command accesses application-related registers not defined in the multimedia card standard.
CMD40	bcr	[31:0]=Fill Bits	R5	GO_IRQ_STATE	Put the system in interrupt mode.
CMD41	Reserved				

Table114 Locking Commands

CMD Index	typology	parameters	response format	abridge	clarification
CMD42	adtc	[31:0]=fill bits	R1b	LOCK_UNLOCK	Set/clear the password or lock/unlock the card. The length of the data block is set by the SET_BLOCKLEN command.
CMD43 ... CMD54	Reserved.				

Table115 Application-related commands

CMD Index	typology	parameters	response format	abridge	clarification
CMD55	ac	[31:16]=RCA[15:0]=fill bits	R1	APP_CMD	Indicates that the next command on the card is an application-related command rather than a standard command.
CMD56	adtc	[31:1] = Fill Bit [0]=RD/WR	-	-	Either used to transfer a block of data to the card or to read a block of data from the card in a general purpose or application related command. The length of the data block is set by the SET_BLOCKLEN command.
CMD57 ... CMD59	Reserved.				
CMD60 ... CMD63	Reserved for producers.				

## 23.5 Response Format

All responses are transmitted on the SDIO\_CMD signal line via the MCCMD command. Transmission of a response always starts at the leftmost side of the bit string corresponding to the response word, and the length of the response word is related to the type of response. A response always has a start bit (always 0) that follows the direction bit of the transmission (card = 0). The value labeled x in the table below indicates a variable part. All responses are CRC-protected except for the R3 response type. Each command code word has an end bit (always 1).

There are five response types and their formats are defined below:

### 23.5.1 R1 (Normal Response Command)

Code length = 48 bits. Bits 45:40 indicate the command index to be responded to, which has a value between 0 and 63. The status of the card is encoded by 32 bits.

Table116 R1 Response

Bit	field width	numerical value	clarification
47	1	0	starting position
46	1	0	transport bit
[45:40]	6	X	Command Index
[39:8]	32	X	card status
[7:1]	7	X	CRC7
0	1	1	end position

### 23.5.2 R1b

Same format as R1, but with the option to send a busy signal on the data line. Upon receipt of these commands, the card may become busy depending on the status prior to receiving the command.

### 23.5.3 R2 (CID, CSD Registers)

Code length = 136 bits. the contents of the CID register will be sent as a response to CMD2 and CMD10. the contents of the CSD register will be sent as a response to CMD9. the card sends only the bits [127...1] of CID and CSD. The card sends out only the bits [127..1] of CID and CSD, and on the receiving end bit 0 of these registers is replaced by the end bit of the response. The card indicates that it is performing an erase operation by pulling MCDAT low; the actual erase operation may be so long that the host can send a CMD7 command to uncheck this card.

Table117 R2 Response

Bit	field width	numerical value	clarification
135	1	0	starting position
134	1	0	transport bit
[133:128]	6	'111111'	Command Index
[127:1]	127	X	card status
0	1	1	end position

### 23.5.4 R3 (OCR Register)

Code length = 48 bits. the contents of the OCR register will be issued as a response to CMD1. The level code is defined as: limited voltage window= low, card busy= low.

Table118 R3 Response

Bit	field width	numerical value	clarification
47	1	0	starting position
46	1	0	transport bit
[45:40]	6	'111111'	Reserved
[39:8]	32	X	OCR Register
[7:1]	7	'1111111'	Reserved
0	1	1	end position

### 23.5.5 R4 (Fast I/O)

Code length = 48 bits. The parameter field contains the RCA of the specified card, the address of the register to be read or written, and its contents.

Table119 R4 Response

Bit	field width	numerical value	clarification
47	1	0	starting position
46	1	0	transport bit
[45:40]	6	'100111'	CMD39
[39:8] Parameter fields	[31:16]	16	X
	[15:8]	8	X
	[7:0]	8	X
[7:1]	7	X	Read the contents of the register
0	1	1	CRC7
			end position

## 23.5.6 R4b

Suitable for SDI/O cards only: an SDIO card receiving CMD5 will return a unique SDIO response R4.

Table120 R4b response

Bit		field width	numerical value	clarification
47		1	0	starting position
46		1	0	transport bit
[45:40]		6	X	Reserved
[39:8] Parameter fields	39	16	X	The card is ready.
	[38:36]	3	X	Number of I/O functions
	35	1	X	current memory
	[34:32]	3	X	padding position
	[31:8]	24	X	I/OORC
[7:1]		7	X	Reserved
0		1	1	end position

When an SDI/O card receives the command CMD5, the I/O portion of the card is enabled and is able to respond normally to all subsequent commands. The enabled state of the I/O card will remain until the next reset, power failure, or receipt of the CMD52 command for I/O reset. Note that an SD card containing only the memory function can respond to the CMD5 command, and its correct response can be: current memory = 1, number of I/O functions = 0. An SD card designed to contain only the memory function according to the SD Memory Card Specification Version 1.0 can detect the CMD5 command as an illegal command and not respond to it. A host that can handle I/O cards will send the CMD5 command and if the card returns response R4, the host will determine the card configuration based on the data in the R4 response.

## 23.5.7 R5 (Interrupt Request)

For multimedia cards only. Code length = 48 bits. If this response is generated by the host, the RCA field in the parameter is 0x0.

Table121 R5 Response

Table 12: RS Response				
Bit		field width	numerical value	clarification
47		1	0	starting position
46		1	0	transport bit
[45:40]		6	'101000'	CMD40
[39:8] Parameter fields	[31:16]	16	X	Successful card or host RCA [31:16]
	[15:0]	16	X	Undefined. Can be used as interrupt data.
[7:1]		7	X	CRC7
0		1	1	end position

## 23.5.8 R6 (Interrupt Request)

For SDI/O cards only. This is the normal response of a memory device to a CMD3 command.

Table122 R6 Response

Bit		field width	numerical value	clarification
47		1	0	starting position
46		1	0	transport bit
[45:40]		6	'101000'	CMD40
[39:8] Parameter fields	[31:16]	16	X	Successful card or host RCA [31:16]
	[15:0]	16	X	Undefined. Can be used as interrupt data.
[7:1]		7	X	CRC7
0		1	1	end position

---

When a CMD3 command is sent to a card with only I/O capability, the status bits [23:8] of the card will change; at this point, the 16 bits in the response will be the values in the SD card with only I/O capability:

- Bit 15 = COM\_CRC\_ERROR
- Bit 14 = ILLEGAL\_COMMAND
- Bit 13 = ERROR
- Bits [12:0] = reserved

## 23.6 SDIO I/O Card Specific Operations

The following functions are SDIO card-specific operations.

- SDIO read wait operation implemented by the SDIO\_D2 signal line.
- SDIO read wait operation realized by stopping the clock.
- SDIO suspend/resume operation (write and read suspend)
- SDIO Interrupt

The SDIO supports these operations only if the SDIO\_DCTRL[11] bit is set; the exception is read pause, which requires no special hardware operation.

### 23.6.1 SDIO I/O Read Wait Operation using SDIO\_D2 Signal Line

The read wait process can be started before the first data block is received by enabling the data channel (setting the SDIO\_DCTRL[0] bit), enabling SDIO specific operations (setting the SDIO\_DCTRL[11] bit), starting the read wait (SDIO\_DCTRL[10]=0 and SDIO\_DCTRL[8]=1), and at the same time, the direction of data transfer is from the card to the SDIO host (SDIO\_DCTRL[1]=1). SDIO host (SDIO\_DCTRL[1]=1), the DPSM will directly enter the read wait state from idle. In the read wait state, after 2 SDIO\_CLK clock cycles, the DPSM drives SDIO\_D2 to '0', in this state, if the RWSTOP bit (SDIO\_DCTRL[9]) is set, the DPSM will stay in the wait state for 2 more SDIO\_CLK clock cycles, (according to the SDIO specification) and drive SDIO\_D2 to '1'. Then the DPSM starts waiting to receive data from the card. When receiving a block of data, the DPSM will not enter read wait even if start read wait is set, the read wait process will start after the CRC is received. RWSTOP must be cleared to start a new read wait operation. During read wait, the SDIO host can monitor the SDIO interrupt on SDIO\_D1.

### 23.6.2 SDIO Read Wait Operation using Stop SDIO\_CLK

If the SDIO card cannot support the read wait operation as described previously, the SDIO can stop SDIO\_CLK to enter read wait (set SDIO\_DCTRL as described in the 23.6.1 section, but set SDIO\_DCTRL[10]=1), and the DPSM stops the clock after 2 SDIO\_CLK cycles following the reception of the end bit of the current block of data, and after the read wait start bit is set restores the clock.

Because SDIO\_CLK is stopped, any command can be sent to the card. During read wait, the SDIO host can monitor the SDIO interrupt on SDIO\_D1.

### 23.6.3 SDIO Suspend/Resume Operation

The SDIO can pause a write operation while sending data to the card. Setting the SDIO\_CMD[11] bit indicates to the CPSM that the current command is a pause command. The CPSM analyzes the response, and upon receipt of an ACK from the card (pause accepted), it acknowledges that it has entered the idle state upon receipt of the CRC of the current data block.

The hardware does not save the number of transmit data blocks remaining after ending the pause operation.

The write operation can be suspended by software: stop DPSM (SDIO\_DCTRL[0]=0) when receiving ACK from the card to the suspend command, DPSM can enter the idle state.

Pause read operation: the DPSM waits in the Wait\_r state and has sent the complete packet before stopping the data transfer into pause. The application then continues to read the Rx FIFO until the FIFO becomes empty, and finally the DPSM automatically enters the idle state.



---

#### 23.6.4 SDIO Interrupt

When the SDIO\_DCTRL[11] bit is set, the SDIO host monitors the SDIO interrupt on the SDIO\_D1 signal line.

### 23.7 CE-ATA Specific Operations

The following are CE-ATA specific operations:

- Sending a command completion signal can shut down the CE-ATA device.
- Receive command completion signal from CE-ATA device
- Use status bits and/or interrupts to signal the completion of the CE-ATA command to the CPU.

These operations are supported by the SDIO host only when the SDIO\_CMD[14] bit is set, i.e., the SDIO host supports these operations only for the CMD61 command of the CE-ATA.

#### 23.7.1 Command Completion Instructions Close

If the "Allow CMD end bit" in SDIO\_CMD[12] is not set and the "Non-interrupt enable bit" in SDIO\_CMD[13] is set, command completion is issued 8 bit cycles after a short response is received. Turn off the signal.

The CPSM enters the suspend state by writing the close sequence "00001" in the command shift register and 43 in the command counter. 8 cycles later, a trigger moves the CPSM to the transmit state. When the command counter reaches 48, the CPSM becomes idle because there is no response to wait for.

#### 23.7.2 Command Completion Indication Enable

If the "Allow CMD end bit" in SDIO\_CMD[12] is set and the "Non-interrupt enable bit" in SDIO\_CMD[13] is set, the CPSM waits for the command completion signal in the Waitcpl state. When a '0' is received on the CMD signal, the CPSM enters the idle state. No new commands can be sent for 7 bit cycles. Then, for the last 5 cycles (beyond the 7 cycles mentioned above), the CMD signal changes to '1' in push-pull mode.

#### 23.7.3 CE-ATA Interrupt

Command completion is notified to the CPU by the status bit SDIO\_STA[23], which can be cleared using the clear bit SDIO\_ICR[23].

Depending on the setting of the mask bit SDIO\_MASKx[23], the SDIO\_STA[23] status bit can generate an interrupt on each interrupt line.

#### 23.7.4 Suspension of CMD 61

If the "Command Completion Indication Off" signal has not been sent, but the CMD61 command needs to be aborted, the command state machine must be turned off. It then becomes idle and the CMD12 command can be sent. During this operation, the "Command completion indication off" signal is not transmitted.

### 23.8 Hardware Flow Control

FIFO underflow (transmit mode) and overflow (receive mode) errors can be avoided using the hardware flow control feature.

The procedure is to stop SDIO\_CK and freeze the SDIO state machine, suspending the data transfer when the FIFO is not able to transmit and receive data. Only the state machine driven by SDIOCLK is frozen and the AHB interface is still working. The FIFO can still be read or written to even when flow control is in effect.

The SDIO\_CLKCR[14] bit must be set to '1' to enable hardware flow control. After reset, the hardware flow control function is turned off.

### 23.9 SDIO Register

The device communicates with the system through a 32-bit control register that can be operated on the AHB.

These peripheral registers must be operated in word (32-bit) format.



3 This register cannot be written to for 7 HCLK clock cycles after writing data. For SDI/O cards, SDIO\_CK can be stopped during read wait, when the SDIO\_CLKCR register does not control SDIO\_CK.

### 23.9.3 SDIO Parameter Register (SDIO\_ARG)

Address offset: 0x08

Reset value: 0x0000 0000

The SDIO\_ARG register contains the 32-bit command parameter that will be sent to the card as part of the command.

Bit	notation	clarification
31:0	CMDARG	<b>CMDARG:</b> Command argument Command parameters are part of the commands sent to the card, and if a command contains a parameter, this register must be loaded before writing the command to the command register.

### 23.9.4 SDIO Command Register (SDIO\_CMD)

Address offset: 0x0C

Reset value: 0x0000 0000

The SDIO\_CMD register contains the command index and command type bits. The command index is sent to the card as part of the command. The command type bit controls the command channel state machine (CPSM).

Bit	notation	clarification
31:15	Reserved	Reserved, always reads 0.
14	ATACMD	<b>ATACMD:</b> CE-ATA command If this bit is set, CPSM goes to CMD61.
13	nIEN	<b>nIEN:</b> not interrupt enable If this bit is not set, the interrupt of the CE-ATA device is enabled.
12	ENCMDcompl	<b>ENCMDcompl:</b> Enable CMDcompletion If this bit is set, the command completion signal is enabled.
11	SDIOSuspend	<b>SDIOSuspend:</b> SDI/O suspend command If this bit is set, the command to be sent is a pause command (only for SDIO cards).
10	CPSMEN	<b>CPSMEN:</b> Command path state machine (CPSM) Enable bit If this bit is set, CPSM is enabled.
9	WAITPEND	<b>WAITPEND:</b> CPSM Waits for ends of data transfer (CmdPend internal signal) (CPSM Waits for ends of data transfer (CmdPend internal signal)) If this bit is set, the CPSM waits for the end of the data transmission before starting to send a command.
8	WAITINT	<b>WAITINT:</b> CPSM waits for interrupt request (CPSM waits for interrupt request) If this bit is set, the CPSM turns off command timeout control and waits for interrupt requests.
7:6	WAITRESP	<b>WAITRESP:</b> Wait for response bits These 2 bits indicate whether the CPSM needs to wait for a response, and if so, the type of response. 00: No response, expect CMDSENT flag 01: short response, expect CMDREND or CCRCFAIL flag 10: No response, expect CMDSENT flag 11: Long response, expecting CMDREND or CCRCFAIL flag
5:0	CMDINDEX	<b>CMDINDEX:</b> Command index The command index is sent to the card as part of the command.

Notes:1 This register cannot be written within 7 HCLK clock cycles after writing data.

2 The multimedia card can send 2 types of responses: a short response that is 48 bits long, or a long response that is 136 bits long. The SD card and the SDI/O card can only send a short

response. The parameters can be varied according to the type of the response, and the software will differentiate between the types of responses according to the commands that are sent. The CE-ATA device only sends a short response. The CE-ATA device only sends a short response. The SDIO card can send a short response.

## 23.9.5 SDIO Command Response Register (SDIO\_RESPCMD)

Address offset: 0x10

Reset value: 0x0000 0000

The SDIO\_RESPCMD register contains the command index from the last command response received. If a command response is transmitted that does not contain a command index (long response or OCR response), the contents of the RESPCMD field are unknown even though it should contain 111111b (the reserved field value in the response).

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

Bit	notation	clarification
31:6	Reserved	Reserved, always reads 0.
5:0	RESPCMD	<b>RESPCMD</b> : response command index (Response command index) Read-only bit containing the command index from the last command response received.

## 23.9.6 SDIO Response 1..4 Registers (SDIO\_RESPx)

Address offset: 0x14+4\*(x-1), where x=1..4

Reset value: 0x0000 0000

The SDIO\_RESP1/2/3/4 registers contain information about the status of the card, i.e., part of the response received.

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

Bit	notation	clarification
31:0	CARDSTATUSx	<b>CARDSTATUSx</b> : See table below.

Depending on the response status, the card's status length is either 32 or 127 bits.

Table 123 Response Type and SDIO\_RESPx Registers

Register	short response	long response
SDIO_RESP1	Card Status [31:0]	Card status [127:96]
SDIO_RESP2	need not	Card status [95:64]
SDIO_RESP3	need not	Card status [63:32]
SDIO_RESP4	need not	Card status [31:1]

## 23.9.7 SDIO Data Timer Register (SDIO\_DTIMER)

Address offset: 0x24

Reset value: 0x0000 0000

The SDIO\_DTIMER register contains the data timeout in card bus clock cycles.

A counter loads values from the SDIO\_DTIMER register and counts decrementally when the data channel state machine (DPSM) enters the Wait\_R or Busy states, and sets the timeout flag if the counter decrements to 0 when the DPSM is in these states.

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

Bit	notation	clarification
31:0	DATETIME	<b>DATETIME</b> : Data timeout period Data timeout in card bus clock cycles.

Note that the data timer register and data length register must be written before writing to the data control register for data transfer.

## 23.9.8 SDIO Data Length Register (SDIO\_DLEN)

Address offset: 0x28

Reset value: 0x0000 0000

The SDIO\_DLEN register contains the length of the data bytes to be transferred. This value is loaded into the data counter when the data transfer begins.

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

Bit	notation	clarification
31:25	Reserved	Reserved, always reads 0.
25:0	DATALENGTH	DATALENGTH: Data length value The number of data bytes to be transferred.

Notes: For block data transfers, the value in the Data Length Register must be a multiple of the data block length (see SDIO\_DCTRL). The Data Timer Register and Data Length Register must be written before writing to the Data Control Register for data transfer.

## 23.9.9 SDIO Data Control Register (SDIO\_DCTRL)

Address offset: 0x2C

Reset value: 0x0000 0000

The SDIO\_DCTRL register controls the data channel state machine (DPSM).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved																				SDIOEN	RWMOD	RWSTOP	RWSTART	DBLOCKSIZE				DMAEN	DTMODE	DTDIR	DTEN	
																				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	notation	clarification
31:12	Reserved	Reserved, always reads 0.
11	SDIOEN	SDIOEN: SDI/O enable functions (SD I/O enable functions) If this bit is set, the DPSM performs SDI/O card-specific operations.
10	RWMOD	RWMOD: Read wait mode 0: Stop SDIO_CK control read wait; 1: Use SDIO_D2 to control read wait.
9	RWSTOP	RWSTOP: Read wait stop (Read wait stop) 0: If RWSTART is set, performs read wait; 1: If RWSTART is set, stop read wait.
8	RWSTART	RWSTART: read wait start (Read wait start) Set this bit to start a read wait operation.
7:4	DBLOCKSIZE	DBLOCKSIZE: Data block length (Data block size) When block data transfer mode is selected, this field defines the data block length: 0000: Block length = 20 = 1 byte; 0001: Block length = 21 = 2 bytes; 0010: Block length = 22 = 4 bytes; 0011: Block length = 23 = 8 bytes; 0100: (decimal 4) block length = 24 = 16 bytes; 0101: (decimal 5) block length = 25 = 32 bytes; 0110: (decimal 6) block length = 26 = 64 bytes; 0111: Block length = 27 = 128 bytes; 1000: Block length = 28 = 256 bytes; 1001: Block length = 29 = 512 bytes; 1010: Block length = 210 = 1024 bytes; 1011: Block length = 211 = 2048 bytes; 1100: Block length = 212 = 4096 bytes; 1101: Block length = 213 = 8192 bytes; 1110: Block length = 214 = 16384 bytes; 1111: Reserved.
3	DMAEN	DMAEN: DMA enable bit 0: Turn off DMA; 1: Enable DMA.
2	DTMODE	DTMODE: Data transfer mode selection 0: Block data transfer; 1: Streaming data transfer.
1	DTDIR	DTDIR: Data transfer direction selection 0: Controller to card; 1: Card to controller.

0	DTEN	DTEN: Data transfer enabled bit If this bit is set to 1, the data transfer starts. Depending on the DTSIR direction bit, the DPSM enters the Wait_S or Wait_R state, or the DPSM enters the read wait state if the RWSTART bit is set at the beginning of the transfer. It is not necessary to clear the enable bit at the end of a data transfer, but SDIO_DCTRL must be changed to allow a new data transfer.
---	------	--

### 23.9.10 SDIO Data Counter Register (SDIO\_DCOUNT)

Address offset: 0x30

Reset value: 0x0000 0000

When the DPSM enters the Wait\_R or Wait\_S state from the idle state, the SDIO\_DCOUNT register is loaded with a value from the data length register (see SDIO\_DLEN), and the value of this counter is decremented until it decreases to 0 during the data transfer, and then the DPSM enters the idle state with the end-of-data-state flag DATAEND set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								DATACOUNT																							
								r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
Bit	notation							clarification																							
31:25	Reserved							Reserved, always reads 0.																							
25:0	DATACOUNT							DATACOUNT: Data count value Reading this register returns the number of data bytes to be transferred; writing this register has no effect.																							

### 23.9.11 SDIO Status Register (SDIO\_STA)

Address offset: 0x34

Reset value: 0x0000 0000

SDIO\_STA is a read-only register that contains two types of flags:

- Static flags (bits [23:22, 10:0]): writing to the SDIO interrupt clear register (see SDIO\_ICR) clears these bits.
- Dynamic flags (bits [21:11]): the state of these bits changes according to the portion of the logic to which they correspond (e.g., the FIFO full and empty flags go high or low in response to data writes to the FIFO).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								CEATAEND	SDIOIT	RXDVAL	TXDVAL	RXFIFOE	TXFIFOE	RXFIFOE	TXFIFOE	RXFIFOE	TXFIFOE	RXACT	TXACT	CMDACT	DBCKEND	STBITERR	DATAEND	CMDSENT	CMDREND	RXOVERR	TXUNDERR	DTIMEOUT	CTIMEOUT	DCRCFAIL	CCRCFAIL
								r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
Bit	notation							show																							
31:24	Reserved							Reserved, always reads 0.																							
23	CEATAEND							CEATAEND: CE-ATA command completion signal received for CMD61																							
22	SDIOIT							SDIOIT: SDIO interrupt received																							
21	RXDVAL							RXDVAL: Data available in receive FIFO (Data available in receive FIFO)																							
20	TXDVAL							TXDVAL: Data available in transmit FIFO (Data available in transmit FIFO)																							
19	RXFIFOE							RXFIFOE: Receive FIFO empty (Receive FIFO empty)																							
18	TXFIFOE							TXFIFOE: transmit FIFO empty (Transmit FIFO empty) If hardware flow control is used, the TXFIFOE signal becomes active when the FIFO contains 2 words.																							
17	RXFIFOE							RXFIFOE: Receive FIFO full (Receive FIFO full) If hardware flow control is used, the RXFIFOE signal becomes active when the FIFO is 2 words short of full.																							
16	TXFIFOE							TXFIFOE: Send FIFO full (Transmit FIFO full)																							
15	RXFIFOE							RXFIFOE: Receive FIFO half full (Receive FIFO half full): there are at least 8 words left in the FIFO.																							
14	TXFIFOE							TXFIFOE: Transmit FIFO half empty: at least 8 more words can be written in the FIFO.																							
13	RXACT							RXACT: Data receive inprogress																							

12	TXACT	TXACT: Data transmit inprogress
11	CMDACT	CMDACT: Command transfer inprogress
10	DBCKEND	DBCKEND: Data block sent/received(CRC check passed))
9	STBITERR	STBITERR: Start bit not detected on all data signals in wide bus mode (Start bit not detected on all data signals in wide bus mode)
8	DATAEND	DATAEND: Data end (data counter, SDIO_DCOUNT=0)(Data end (data counter,SDID COUNT,is zero))
7	CMDSENT	CMDSENT: Command sent (no response required)
6	CMDREND	CMDREND: response received (CRC detection successful) (Command response)
5	RXOVERR	RXOVERR: Received FIFO overrun error
4	TXUNDERR	TXUNDERR: Transmit FIFO underrun error
3	DTIMEOUT	DTIMEOUT: Data timeout
2	CTIMEOUT	CTIMEOUT: Command response timeout The command timeout is a fixed value of 64 SDIO_CK clock cycles.
1	DCRCFAIL	DCRCFAIL: Data block sent/received (CRC detection failure)
0	CCRCFAIL	CCRCFAIL: Command response received (CRC detection failure) (Command response received)

### 23.9.12 SDIO Clear Interrupt Register (SDIO\_ICR)

Address offset: 0x38

Reset value: 0x0000 0000

SDIO\_ICR is a write-only register and writing a '1' to the corresponding register bit will clear the corresponding bit in the SDIO\_STA status register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								CEATAENDC	SDIOITC	Reserved											DBCKENDC	STBITERRC	DATAENDC	CMDSENTC	CMDRENDC	RXOVERRC	TXUNDERRC	DTIMEOUTC	CTIMEOUTC	DCRCFAILC	CCRCFAILC

RW RW

RW RW RW RW RW RW RW RW RW RW RW

Bit	notation	clarification
31:24	Reserved	Reserved, always reads 0.
23	CEATAENDC	CEATAENDC: CEATAEND flag clear bit (CEATAEND flag clear bit) Software sets this bit to clear the CEATAEND flag.
22	SDIOITC	SDIOITC: SDIOIT flag clear bit (SDIOIT flag clear bit) Software sets this bit to clear the SDIOIT flag.
21:11	Reserved	Reserved, always reads 0.
10	DBCKENDC	DBCKENDC: DBCKEND flagclear bit Software sets this bit to clear the DBCKEND flag.
9	STBITERRC	STBITERRC: STBITERR flagclear bit (STBITERR flagclear bit) Software sets this bit to clear the STBITERR flag.
8	DATAENDC	DATAENDC: DATAEND flag clear bit (DATAEND flag clear bit) Software sets this bit to clear the DATAEND flag.
7	CMDSENTC	CMDSENTC: CMDSENT flag clear bit Software sets this bit to clear the CMDSENT flag.
6	CMDRENDC	CMDRENDC: CMDREND flag clear bit Software sets this bit to clear the CMDREND flag.
5	RXOVERRC	RXOVERRC: RXOVERR flag clear bit Software sets this bit to clear the RXOVERR flag.
4	TXUNDERRC	TXUNDERRC: TXUNDERR flag clear bit (TXUNDERR flag clear bit) Software sets this bit to clear the TXUNDERR flag.
3	DTIMEOUTC	DTIMEOUTC: DTIMEOUT flag clear bit Software sets this bit to clear the DTIMEOUT flag.
2	CTIMEOUT	CTIMEOUT: CTIMEOUT flag clear bit Software sets this bit to clear the CTIMEOUT flag.
1	DCRCFAILC	DCRCFAILC: DCRCFAIL flag clear bit Software sets this bit to clear the DCRCFAIL flag.
0	CCRCFAILC	CCRCFAILC: CCRCFAIL flag clear bit. (CCRCFAIL flag clear bit) Software sets this bit to clear the CCRCFAIL flag.



## 23.9.13 SDIO Interrupt Mask Register (SDIO\_MASK)

Address offset: 0x3C

Reset value: 0x0000 0000

In the corresponding position '1', the SDIO\_MASK interrupt mask register determines which status bit generates the interrupt.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								CEATAENDIE	SDIOITIE	RXDVALIE	TXDVALIE	RXFIFOEIE	TXFIFOEIE	RXFIFOEIE	TXFIFOEIE	RXFIFOHIE	TXFIFOHIE	RXACTIE	TXACTIE	CMDACTIE	DBCKENDIE	STBITERRIE	DATAENDIE	CMDSENIE	CMDRENIE	RXOVERRIE	TXUNDERRIE	DTMEOUTIE	CTIMEOUTIE	DCRCFAILIE	CCRCFAILIE
								RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit	notation	clarification
31:24	Reserved	Reserved, always reads 0.
23	CEATAENDIE	<b>CEATAENDIE:</b> Allow CE-ATA command completion signal received interrupt enable This bit is set/cleared by software to allow/disable the interrupt generation function upon receipt of the CE-ATA command completion signal. 0: No interrupt generated when CE-ATA command completion signal is received 1: Generate interrupt when CE-ATA command completion signal is received
22	SDIOITIE	<b>SDIOITIE:</b> allow SDIO mode interrupt received interrupt (SDIO mode interrupt received interrupt enable) This bit is set/cleared by software to allow/disable the SDIO mode interrupt has received interrupt function. 1: SDIO mode interrupt has been received without generating an interrupt 0: SDIO mode interrupt has been received to generate an interrupt
21	RXDVALIE	<b>RXDVALIE:</b> Data available in RxFIFO interrupt enable This bit is set/cleared by software to allow/disable valid interrupts for data in the receive FIFO. 0: Receiving data in FIFO is valid without generating an interrupt 1: Receiving data in FIFO effectively generates an interrupt
20	TXDVALIE	<b>TXDVALIE:</b> Data in the transmit FIFO is valid to generate an interrupt (Data available in RxFIFO interrupt enable) This bit is set/cleared by software to allow/disable valid interrupts for data in the transmit FIFO. 0: send data in FIFO valid without generating an interrupt 1: Send data in FIFO to generate interrupt effectively
19	RXFIFOEIE	<b>RXFIFOEIE:</b> Receive FIFO empty interrupt (RxFIFO empty interrupt enable) This bit is set/cleared by software to allow/disable receive FIFO air break. 0: Receive FIFO empty without generating an interrupt 1: Receive FIFO empty generate interrupt
18	TXFIFOEIE	<b>TXFIFOEIE:</b> send FIFO empty generate interrupt (TxFIFO empty interrupt enable) This bit is set/cleared by software to allow/disable transmit FIFO air breaks. 0: send FIFO empty without generating interrupt 1: Send FIFO empty to generate interrupt
17	RXFIFOEIE	<b>RXFIFOEIE:</b> Receive FIFO full interrupt enable This bit is set/cleared by software to allow/disable the receive FIFO full interrupt. 0: Receive FIFO full without generating an interrupt 1: Receive FIFO full to generate interrupt
16	TXFIFOEIE	<b>TXFIFOEIE:</b> Send FIFO full interrupt enable This bit is set/cleared by software to allow/disable the transmit FIFO full interrupt. 0: send FIFO full without generating interrupt 1: Send FIFO full to generate interrupt
15	RXFIFOHIE	<b>RXFIFOHIE:</b> Receive FIFO half full generate interrupt (RxFIFO half full interrupt enable) This bit is set/cleared by software to allow/disable the receive FIFO half-full interrupt. 0: Receive FIFO half full without generating an interrupt 1: Receive FIFO half-full interrupt generation
14	TXFIFOHIE	<b>TXFIFOHIE:</b> send FIFO half empty generate interrupt (TxFIFO half empty interrupt enable) This bit is set/cleared by software to allow/disable the transmit FIFO half-empty interrupt. 0: Send FIFO half empty without generating an interrupt 1: Send FIFO half empty to generate an interrupt



13	RXACTIE	<b>RXACTIE:</b> Data receiving acting interrupt enable This bit is set/cleared by software to allow/disable the data being received interrupt. 0: Data is being received without generating an interrupt 1: Data being received generates an interrupt
12	TXACTIE	<b>TXACTIE:</b> Data transmit acting interrupt enable This bit is set/cleared by software to allow/disable data transmit acting interrupt. 0: Data is being sent without generating an interrupt 1: Data being sent generates an interrupt
11	CMDACTIE	<b>CMDACTIE:</b> Command acting interrupt enable is being transmitted. This bit is set/cleared by software to allow/disable the transmit command interrupt in progress. 0: Command is being transmitted without generating an interrupt 1: Command being transmitted generates an interrupt
10	DBCKENDIE	<b>DBCKENDIE:</b> data block end interrupt enable This bit is set/cleared by software to allow/disable the end of data block transfer interrupt. 0: End of data block transfer without generating an interrupt 1: Interrupt generated at the end of data block transfer
9	STBITERRIE	<b>STBITERRIE:</b> Start bit error interrupt enable This bit is set/cleared by software to allow/disable the start bit error interrupt. 0: start bit error does not generate an interrupt 1: Start bit error generates an interrupt
8	DATAENDIE	<b>DATAENDIE:</b> Data transfer end interrupt (Dataendinterruptenable) This bit is set/cleared by software to allow/disable the end-of-data-transfer interrupt. 0: End of data transfer without generating an interrupt 1: End of data transfer generates an interrupt
7	CMDSENTIE	<b>CMDSENTIE:</b> Command sent interrupt enable This bit is set/cleared by software to allow/close the command has been sent interrupt. 0: Command has been sent without generating an interrupt 1: Command has been sent to generate an interrupt
6	CMDRENDIE	<b>CMDRENDIE:</b> Command response received interrupt enable This bit is set/cleared by software to allow/disable the receive-to-response interrupt. 0: response received without generating an interrupt 1: Receiving a response generates an interrupt
5	RXOVERRIE	<b>RXOVERRIE:</b> Receive FIFO overrun error interrupt enable This bit is set/cleared by software to allow/disable the receive FIFO overflow error interrupt. 0: Receive FIFO overflow error without generating interrupt 1: Receive FIFO overflow error generates an interrupt
4	TXUNDERRIE	<b>TXUNDERRIE:</b> Transmit FIFO underrun error interrupt enable This bit is set/cleared by software to allow/disable the transmit FIFO underflow error interrupt. 0: Send FIFO underflow error without generating an interrupt 1: Send FIFO overflow error to generate an interrupt
3	DTIMEOUTIE	<b>DTIMEOUTIE:</b> Data timeout interrupt enable This bit is set/cleared by software to allow/disable the data timeout interrupt. 0: Data timeout does not generate an interrupt 1: Data timeout generates an interrupt
2	CTIMEOUTIE	<b>CTIMEOUTIE:</b> Command timeout interrupt enable This bit is set/cleared by software to allow/disable command timeout interrupts. 0: Command timeout does not generate an interrupt 1: Command timeout generates an interrupt
1	DCRCFAILIE	<b>DCRCFAILIE:</b> Data block CRC detection failure generates interrupt (Data CRC fail interrupt enable) This bit is set/cleared by software to allow/disable data block CRC detection failure interrupts. 0: Data block CRC detection failure does not generate an interrupt 1: Data block CRC detection failure generates an interrupt
0	CCRCFAILIE	<b>CCRCFAILIE:</b> Command CRC fail detection generate interrupt (Command CRC fail interrupt enable) This bit is set/cleared by software to allow/disable command CRC detection failure interrupts. 0: Command CRC detection failure does not generate an interrupt 1: Command CRC detection failure generates an interrupt

## 23.9.14 SDIO FIFO Counter Register (SDIO\_FIFOCNT)

Address offset: 0x48

Reset value: 0x0000 0000

The SDIO\_FIFOCNT register contains the number of data words that have not yet been written to or read from the FIFO. When the data transfer enable bit DTEN is set in the data control register (SDIO\_DCTRL) and the DPSM is idle, the FIFO counter loads the value from the data length register (see SDIO\_DLEN). If the data length is not aligned to a word (multiple of 4), the last remaining 1 to 3 bytes are treated as a word.

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

Bit	notation	clarification
31:24	Reserved	Reserved, always reads 0.
23:0	FIFOCOUNT	FIFOCOUNT: The number of data words that will be written to or read from the FIFO.

## 23.9.15 SDIO Data FIFO Register (SDIO\_FIFO)

Address offset: 0x80

Reset value: 0x0000 0000

A receive and transmit FIFO is a 32-bit wide read or write set of registers that contains 32 registers at 32 consecutive addresses, and the CPU can read and write multiple operands using the FIFO.

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

Bit	notation	instructions
31:0	FIFODATA	FIFODATA: Receive and transmit FIFO data The FIFO data occupies 32 32-bit words addressed: (SDIO base address + 0x80) to (SDIO base address + 0xFC)

## 23.9.16 SDIO Register Image

The following table summarizes the SDIO registers.

Table 23.9.16 SDIO register images

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
0x00	SDIO_POWER	Reserved																										PWRCTRL									
0x04	SDIO_CLKCR	Reserved																		HWFC_EN	NEGEDGE	WIDBUS	BYPASS	PWRSAPV	CLKEN	CLKDIV											
0x08	SDIO_ARG	CMDARG																																			
0x0C	SDIO_CMD	Reserved																		CE-ATACMD	nEN	ENCMDcompl	SDIOsuspend	CPSMEN	WAITPEND	WAITINT	WAITRESP	CMDINDEX									
0x10	SDIO_RESPCMD	Reserved																										RESPCMD									
0x14	SDIO_RESP1	CARDSTATUS1																																			
0x18	SDIO_RESP2	CARDSTATUS2																																			
0x1C	SDIO_RESP3	CARDSTATUS3																																			
0x20	SDIO_RESP4	CARDSTATUS4																																			
0x24	SDIO_DTIMER	DATATIME																																			
0x28	SDIO_DLEN	Reserved						DATALENGTH																													
0x2C	SDIO_DCTRL	Reserved																		SDIOEN	RWMOD	RWSTOP	RWSTART	DBLOCKSIZE				DMAEN	DTMODE	DTDIR	DTEN						
0x30	SDIO_DCOUNT	Reserved						DATACOUNT																													
0x34	SDIO_STA	Reserved										CEATAEND	SDIOIT	RXDAVL	TXDAVL	RXFIOE	TXFIOE	RXFIOF	TXFIOF	RXFIOHF	TXFIOHE	RXACT	TXACT	CMDACT	DBCKEND	STBITERR	DATAEND	CMDSENT	CMDREND	RXOVERR	TXUNDERR	DTMEOUT	CTIMEOUT	DCRCFAIL	CCRCFAIL		
0x38	SDIO_ICR	Reserved										CEATAENDC	SDIOITC	Reserved										DBCKENDC	STBITERRC	DATAENDC	CMDSENTC	CMDREND	RXOVERR	TXUNDERR	DTMEOUTC	CTIMEOUTC	DCRCFAILC	CCRCFAILC			
0x3C	SDIO_MASK	Reserved										CEATAENDIE	SDIOITIE	RXDAVLIE	TXDAVLIE	RXFIOEIE	TXFIOEIE	RXFIOFIE	TXFIOFIE	RXFIOHFIE	TXFIOHEIE	RXACTIE	TXACTIE	CMDACTIE	DBCKENDIE	STBITERRIE	DATAENDIE	CMDSENTIE	CMDRENDIE	RXOVERRIE	TXUNDERRIE	DTMEOUTIE	CTIMEOUTIE	DCRCFAILIE	CCRCFAILIE		
0x48	SDIO_FIFOCNT	Reserved						FIFOCOUNT																													
0x80	SDIO_FIFO	FIFODATA																																			

See Table 23.9.1 for register start addresses.

## 24 USB Full Speed Device Interface (USB)

### 24.1 Introduction to USB

The USB peripheral implements the interface between the USB 2.0 full-speed bus and the APB1 bus.

USB peripherals support USB suspend/resume operation, which can stop the device clock for low power consumption.

### 24.2 USB Main Features

- Compliant with USB 2.0 full-speed device specifications
- Configurable from 1 to 8 USB endpoints
- CRC (Cyclic Redundancy Check) generation/checksum, reverse non-return-to-zero (NRZI) encoding/decoding and bit-filling
- Support for synchronized transmission
- Double buffer mechanism supporting batch/synchronized endpoints
- Support USB suspend/resume operation
- Frame-lock clock pulse generation

**Notes:** USB and CAN share a dedicated 512-byte SRAM memory for data transmission and reception, so it is not possible to use USB and CAN at the same time (the shared SRAM is accessed by the USB and CAN modules in a mutually exclusive way). USB and CAN can be used in an application at the same time but not at the same time.

The following diagram shows the block diagram of the USB peripherals

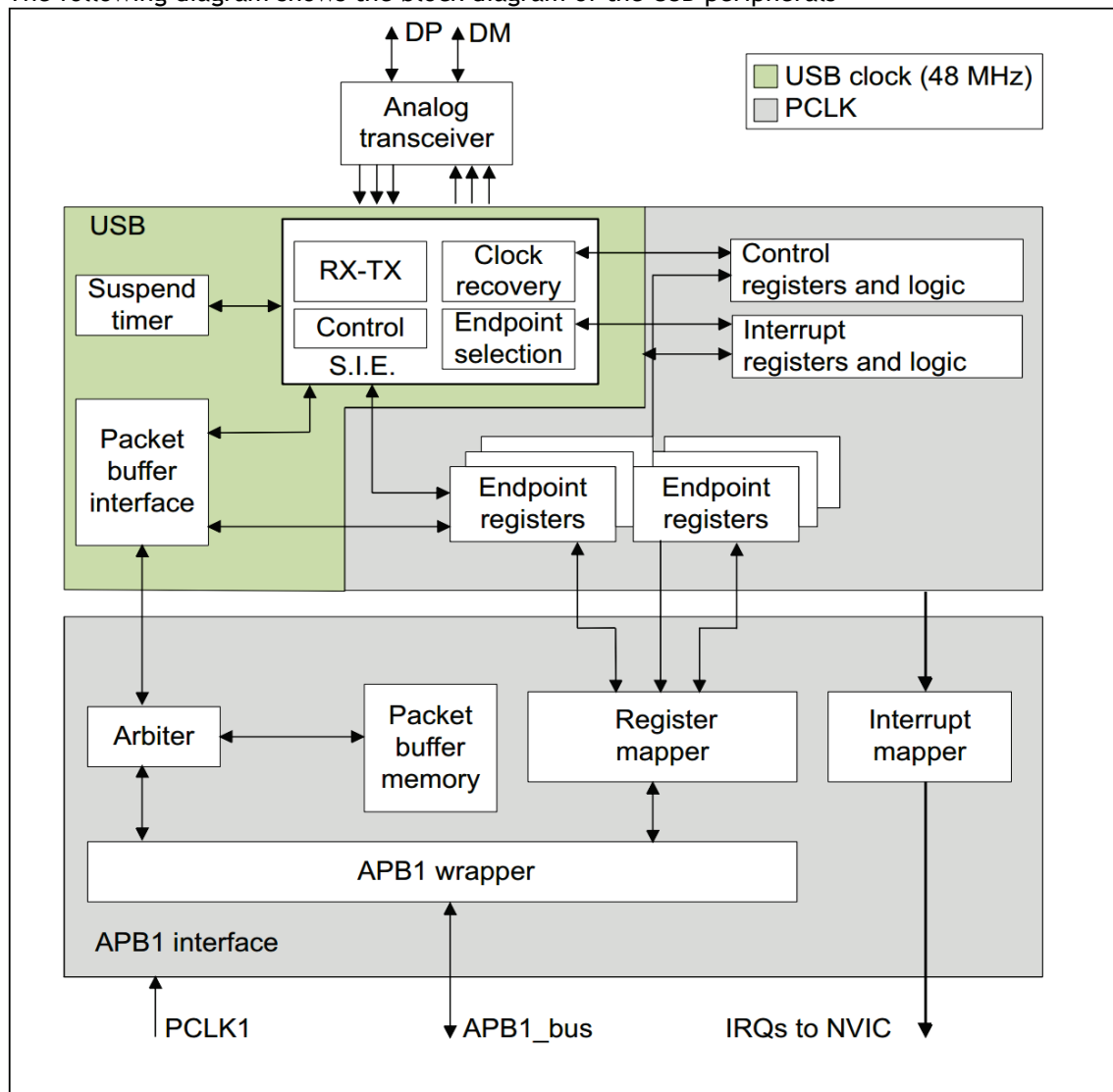


Figure194 USB Device Block Diagram

## 24.3 USB Function Description

The USB module provides a USB-compliant communication link between the functions realized by the PC host and the microcontroller. The data transfer between the PC host and the microcontroller is accomplished by sharing a dedicated data buffer that can be directly accessed by the USB peripheral. The size of this dedicated data buffer is determined by the number of endpoints used and the maximum data packet size of each endpoint, with a maximum of 512 bytes of buffer per endpoint for up to 16 unidirectional or 8 bidirectional endpoints. The USB module communicates with the PC host and implements the detection of token packets, the processing of data sending/receiving and the processing of handshake packets in accordance with the USB specification. The entire format of the transmission is done in hardware, which includes CRC generation and checksum.

Each endpoint has a buffer description block that describes the buffer address, size, and number of bytes to be transferred for that endpoint.

When the USB module recognizes a valid token grouping for a function/endpoint, (if data transfer is required and the endpoint is configured) the associated data transfer occurs thereafter. The USB module implements the exchange of data between the port and the dedicated buffer via an internal 16-bit register. After all data transfers have been completed, the appropriate handshake packet is sent or received, if required, depending on the direction of the transfer.

At the end of the data transfer, the USB module will trigger an interrupt associated with the endpoint, which can be determined by the microcontroller by reading the status register and/or utilizing different interrupt handlers:

- Which endpoint needs to be served
- What type of transmission was in progress when errors such as bit fill, format, CRC, protocol, missing ACK, buffer overflow/buffer underrun, etc. were generated.

The USB module provides a special double-buffer mechanism for synchronous transfers and high-throughput bulk transfers, which ensures that when the microcontroller uses one buffer, the USB peripheral can always use the other.

It is always possible to put the USB module in low power mode (SUSPEND mode) by writing to the control register at any time when it is not needed. In this mode, no quiescent current consumption is generated and the USB clock is slowed down or stopped. The USB module can be woken up in low-power mode by detecting a data transfer on the USB line. It is also possible to connect a specific interrupt input source directly to the wake-up pin in order to enable the system to immediately resume the normal clocking system and to support the direct starting or stopping of the clocking system.

### 24.3.1 USB Function Module Description

The USB module implements all the features of the standard USB interface and consists of the following parts:

- **Serial Interface Controller (SIE):** This module includes functions such as frame header synchronization field identification, bit padding, CRC generation and checksum, PID verification/generation, and handshake packet processing. It interacts with the USB transceiver and uses the virtual buffer provided by the packet buffer interface to store localized data. It also generates signals based on USB events, and endpoint-related events such as end of transmission or correct reception of a packet, such as Start of Frame, USB reset, data errors, etc. These signals are used to generate interrupts.
- **Timer:** The function of this module is to generate a clock pulse synchronized with the start-of-frame telegram and detect a global hang condition (of the host) in a state where no data is transmitted for 3ms.
- **Packet Buffer Interface:** This module manages those temporary local memory units used for sending and receiving. It allocates the appropriate buffer according to the SIE requirements and locates to the address of the memory area pointed to by the endpoint register. It automatically increments the address after each byte transfer until the end of the data packet transfer. It records the number of bytes transferred and prevents buffer overflow.
- **Endpoint-related registers:** Each endpoint has a register associated with it that describes the endpoint type and current state. For unidirectional and single-buffer endpoints, a single register can be used to implement two different endpoints. A total of 8 registers can be used to implement up to 16 unidirectional/single buffer endpoints or 7 double buffer endpoints or a combination of these. For example, 4 double buffered endpoints and 8 single buffered/unidirectional endpoints can be implemented at the same time.

- **Control Registers:** These registers contain status information for the entire USB module and are used to trigger USB events such as recovery, low power consumption, etc.
- **Interrupt Registers:** These registers contain interrupt mask information and logging information for interrupt events. Configuring and accessing these registers allows you to obtain information about the interrupt source, interrupt status, etc., and to clear the status flags of pending interrupts.

*Notes: Endpoint 0 is always used as the control endpoint in single buffer mode.*

The USB module is connected to the APB1 bus through the APB1 interface component, which consists of the following parts:

- **Packet Buffer:** Data packets are cached in a packet buffer, which is controlled by the packet buffer interface and creates data structures. The application software can access this buffer directly. It is 512 bytes in size and consists of 256 16-bit words.
- **Arbiter:** This component handles memory requests from the APB1 bus and the USB interface. It resolves bus conflicts by providing higher access priority to APB1 and always reserves half of the memory bandwidth for USB to complete the transfer. It implements a virtual dual-port SRAM using a time-division multiplexing strategy, i.e., it allows applications to access the memory while the USB is transferring. This strategy also allows multi-byte APB1 transfers of arbitrary length.
- **Register Mapping Unit:** This part maps the various byte-width and bit-width registers of the USB module into a 16-bit wide memory set that can be addressed by the APB1.
- **APB1 Package:** This part provides the interface to APB1 for buffers and registers and maps the entire USB module to the APB1 address space.
- **Interrupt Mapping Unit:** maps USB events that may generate interrupts to three different NVIC request lines:
  - USB low priority interrupt (channel 20): can be triggered by all USB events (correct transfer, USB reset, etc.). The firmware should first determine the source of the interrupt before handling it.
  - USB High Priority Interrupt (channel 19): can only be triggered by correct transfer events for synchronous and double-buffered bulk transfers, in order to guarantee maximum transfer rates.
  - USB wake-up interrupt (channel 42): triggered by a wake-up event in USB suspend mode.

## 24.4 Issues to Consider in Programming

In the following sections, the interaction process between the USB module and the application program is described, which is helpful to simplify the development of the application program.

### 24.4.1 System Reset and Power-on Reset

When a system reset or power-on reset occurs, the first thing the application program needs to do is to provide the clock signals required by the USB module, and then clear the reset signals so that the program can access the USB module's registers. The initialization process after a reset is described below:

First, the clock of the register unit is activated by the application program, and then the relevant control bits of the device clock management logic unit are configured to clear the reset signal.

Second, the PDWN bit of the CNTR register must be configured to turn on the analog portion associated with the USB transceiver, which requires special handling. This bit turns on the internal reference voltage that powers the endpoint transceiver. Since turning on the internal voltage requires a startup time (tSTARTUP in the datasheet), during which the USB transceiver is in an uncertain state, it is necessary to wait for a certain period of time after setting the PDWN bit of the CNTR register before clearing the reset signal of the USB module (clearing the FRES bit of the CNTR register), and the contents of the ISTR register to clear the reset signal of the USB module before enabling any other operation of the unit. The reset signal of the USB module must be cleared (clearing the FRES bit on the CNTR register), and the contents of the ISTR register must be cleared in order to clear the unprocessed false interrupt flag before enabling any other unit operation.

Finally, the application needs to provide the USB module with the 48MHz clock defined by the standard by configuring the appropriate control bits of the device clock management logic.

When the system is reset, the application should initialize all registers and packet buffer description tables needed to enable the USB module to generate normal interrupts and complete data transfers. All registers not related to endpoints need to be initialized according

to the needs of the application (e.g. interrupt enable selection, packet buffer address selection, etc.). Next follow the USB reset processing (see next paragraph).

### USB Reset (RESET Interrupt)

When a USB reset occurs, the USB module enters the system reset state described in the previous sections: communication is disabled at all endpoints (the USB module will not respond to any packets). After a USB reset, the USB module is enabled and the default control endpoint with address 0 (endpoint 0) needs to be enabled as well. This can be accomplished by configuring the EF bit of the USB\_DADDR register, the EP0R register and the associated packet buffer. During the enumeration phase of the USB device, the host will assign a unique address to the device, which must be written into the ADD[6:0] bits of the USB\_DADDR register, along with configuring the other required endpoints.

When a reset interrupt is generated, the application program must enable the transmission of endpoint 0 within 10ms after the interrupt is generated.

### Structure and Use of Grouping Buffers

Each bi-directional endpoint can receive or send data. The received data is stored in a dedicated buffer specified by that endpoint, while another buffer is used to hold the data to be sent. Access to these buffers is implemented by the packet buffer interface module, which makes a buffer access request and waits for an acknowledgement message before returning. To prevent conflicts between the microcontroller and the USB module for buffer accesses, the buffer interface module uses an arbitration mechanism so that half of the cycles of the APB1 bus are used for microcontroller accesses, and the other half is guaranteed to be accessed by the USB module. In this way, the microcontroller and the USB module access the packet buffer as if it were a dual-port SRAM, and even if the microcontroller accesses the buffer consecutively, no access conflict arises.

The USB module uses a fixed clock, which according to the USB standard is fixed at 48 MHz. the clock on the APB1 bus can be greater or less than this frequency.

*Notes: To meet the system requirements for USB data transfer rate and packet buffer interface, the frequency of the APB1 bus clock must be greater than 8MHz to avoid data buffer overflow or underruns*

Each endpoint corresponds to two packet buffers (typically one for sending and the other for receiving). These buffers can be located anywhere in the entire packet storage area, since their addresses and lengths are defined in the buffer description table, which is similarly located in the packet buffers, whose addresses are determined by the USB\_BTABLER register.

Each table entry of the buffer description table is associated with an endpoint register, which consists of four 16-bit words, so that the starting address of the buffer description table is aligned by 8 bytes (the lowest 3 bits of the register are always '000'). Section0 describes the Buffer Description Table entries in detail. In the case of unsynchronized non-double buffered unidirectional endpoints, only one packet buffer is required (i.e. a packet buffer in the transmit direction).

Other unused endpoints or buffer description table entries in an unused direction can be used for other purposes. Synchronous and double-buffered bulk endpoints have special packet buffer handling (see Section24.4.3 : Synchronous Transmission and Section24.4.2 : Double-Buffered Endpoints, respectively). The following figure depicts the relationship between buffer description table entries and packet buffer regions.



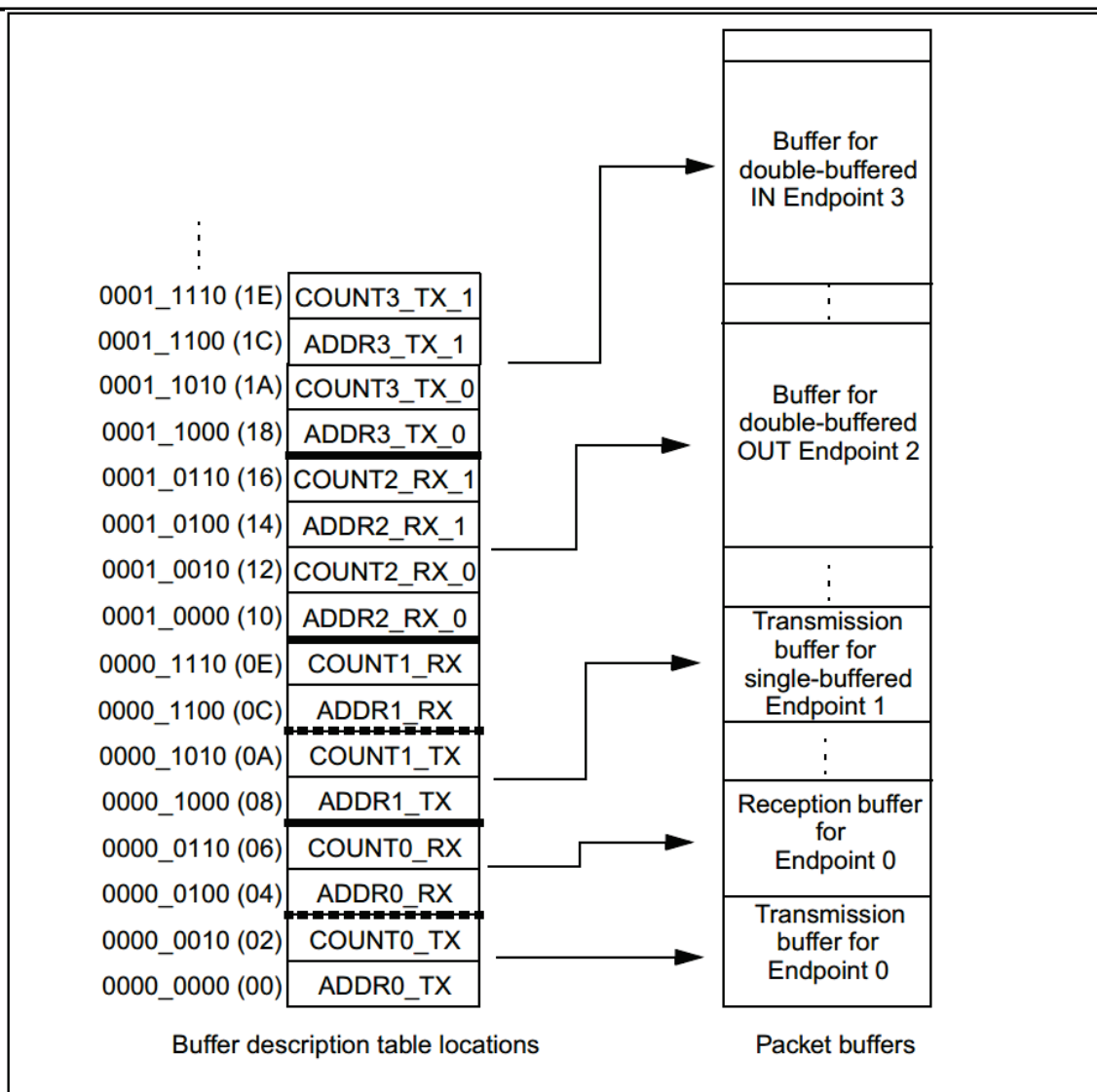


Figure195 Locating the corresponding buffer description table entries for grouped buffers  
Whether receiving or sending, the packet buffer is used from the bottom. the USB module does not change the contents of any buffer beyond the area of the buffer currently allocated to it. If the buffer receives a packet of data larger than itself, it will only receive data up to its own size and discard the rest, i.e. a so-called buffer overflow exception has occurred.

### Endpoint initialization

The first step in initializing an endpoint is to write the appropriate value to the ADDRn\_TX or ADDRn\_RX register so that the USB module can locate the data to be transmitted or prepare a buffer to receive the data. The EP\_TYPE bit of the USB\_EPnR register determines the basic type of the endpoint, and the EP\_KIND bit determines the special characteristics of the endpoint. As the sender, you need to set the STAT\_TX bit of the USB\_EPnR register to enable the endpoint and configure the COUNTn\_TX bit to determine the send length. As the receiver, you need to set the STAT\_RX bit to enable the endpoint and set the BL\_SIZE and NUM\_BLOCK bits to determine the size of the receive buffer to detect buffer overflow exceptions. For unidirectional endpoints for asynchronous non-double-buffered bulk transfers, only one register in the direction of transfer needs to be set. Once the endpoint is enabled, the application program can no longer modify the value of the USB\_EPnR register and where ADDRn\_TX/ADDRn\_RX, COUNTn\_TX/COUNTn\_RX are located because these values are modified by the hardware in real time. When the data transfer is complete, a CTR interrupt is generated, at which point the above registers can be accessed and the new transfer re-enabled.



### IN Packet (For Data Transmission)

When a one IN token packet is received, the USB module will access the corresponding ADDRn\_TX and COUNTn\_TX registers according to the table entries of the buffer description table and store the values in these registers into the internal 16-bit registers ADDR and COUNT (which are not accessible by the application program), if the received address matches a configured endpoint address. At this point, the USB module starts to send DATA0 or DATA1 packets according to the DTOG\_TX bit and accesses the buffer (please refer to the paragraph 'Structure and Usage of Packet Buffers'). After the IN packet has been transmitted, the first byte read from the buffer is loaded into the output shift register and transmission begins. After the last data byte has been transmitted, the calculated CRC will be sent. If the packet received by the

The corresponding endpoint is invalid and will send a NAK or STALL handshake packet without data based on the STAT\_TX bit on the USB\_EPnR register.

The ADDR internal register is used as a pointer to the current buffer, and the COUNT register is used to keep track of the number of bytes remaining to be transferred. The USB bus uses low byte first to transfer data read from the buffer. The data is read starting from the data packet buffer pointed to by ADDRn\_TX and is COUNTn\_TX/2 words long. If the data packet being sent is an odd number of bytes, only the lower 8 bits of the last word are used.

After receiving an ACK in response from the host, the USB\_EPnR register is updated with the following values: the DTOG\_TX bit is flipped, the STAT\_TX bit is '10' to invalidate the endpoint, and the CTR\_TX bit is set. The application program needs to identify the USB endpoint generating the interrupt by the EP\_ID and DIR bits of the USB\_ISTR register. The interrupt service program for the CTR\_TX event needs to first clear the interrupt flag bit, then prepare the data buffer that needs to be sent, update the COUNTn\_TX to the number of bytes that need to be transferred next time, and then finally set the STAT\_TX bit to '11' (endpoint valid) to enable data transmission again. When the STAT\_TX bit is '10' (the endpoint is in NAK state), any IN request sent to that endpoint will be NAKed, and the USB host will retransmit the IN request until the endpoint confirms that the request is valid. The above procedure is required to be followed to avoid losing the next IN transmission request immediately following the last CTR interrupt request.

### OUT Packet and SETUP Packet (For Data Reception)

The USB module handles both types of packets in essentially the same way; special handling of the SETUP packet is described in more detail below in the section on controlling transmission. When an OUT or SETUP packet is received, if the address matches that of a valid endpoint, the USB module accesses the buffer description table, finds the ADDRn\_RX and COUNTn\_RX registers associated with that endpoint, and saves the value of the ADDRn\_RX register in the internal ADDR register. At the same time, COUNT is reset and the values of BL\_SIZE and NUM\_BLOCK read from COUNTn\_RX are used to initialize the internal 16-bit register BUF\_COUNT, which is used for detecting buffer overflows (all internal registers are not accessible by the application program). The USB module organizes the subsequently received data in a word-wise manner (the first received is the low byte) and stores it in the packet buffer pointed to by ADDR. At the same time, the BUF\_COUNT value is automatically decremented and the COUNT value is automatically incremented. When the end-of-data packet signal is detected, the USB module verifies the correctness of the received CRC. If no error occurs in the transmission, an ACK handshake packet is sent to the host. Even if a CRC error or some other type of error (bit padding, framing error, etc.) occurs, the data will still be saved in the packet buffer, at least up to the point where the error occurred, except that no ACK packet will be sent and the ERR bit in the USB\_ISTR register will be set. In this case, the application program usually does not need to interfere with the processing and the USB module will automatically recover from the transmission error and be ready for the next transmission. If the endpoint corresponding to the received packet is not ready, the USB module will send a NAK or STALL packet according to the STAT\_RX bit of the USB\_EPnR register and the data will not be written to the receive buffer.

The value of ADDRn\_RX determines the start address of the receive buffer, the length of which is determined by the length of the data packet containing the CRC (i.e., valid data length + 2), but cannot exceed the length of the buffer as defined by BL\_SIZE and NUM\_BLOCK. If the length of the received data packet exceeds the buffer's range, the data that exceeds the range will not be written to the buffer, and the USB module will report that the buffer has overflowed

---

and send a STALL handshake packet to the host to notify that this transmission has failed, and no interrupt will be generated.

If the transfer completes correctly, the USB module sends the ACK handshake packet and the value of the internal COUNT register is copied to the corresponding COUNTn\_RX register, the values of BL\_SIZE and NUM\_BLOCK remain unchanged and do not need to be rewritten. The USB\_EPnR register is updated in the following way: the DTOG\_RX bit is flipped, STAT\_RX= 10 (NAK) to invalidate the endpoint, and the CTR\_RX bit (which will trigger an interrupt if the CTR interrupt has been enabled). If an error or buffer overflow occurs during transmission, none of the previously listed actions will occur. When a CRT interrupt occurs, the application program needs to first identify which endpoint is the subject of the interrupt request based on the EP\_ID and DIR bits of the USB\_ISTR register. When processing a CTR\_RX interrupt event, the application program first identifies the type of transmission (based on the SETUP bit in the USB\_EPnR register), while clearing the interrupt flag bit, and then reads the COUNTn\_RX register pointed to by the associated Buffer Description Table table entry to obtain the total number of bytes transferred for this transmission. After processing the received data, the application program needs to set the STAT\_RX bit in USB\_EPnR to '11' to enable the next transmission. When the STAT\_RX bit is '10' (NAK), any OUT request sent to the endpoint will be NAK'd, and the PC host will keep retransmitting the NAK'd packet until it receives an ACK handshake packet from the endpoint. The sequence of operations described above is necessary to avoid losing another OUT packet request immediately following the last CTR interrupt.

### Control Transmission

The control transmission consists of 3 phases, first the SETUP phase in which the host sends the SETUP packet, then the data phase in which the host sends zero or more data, and finally the status phase, which consists of data packets in the opposite direction of the data phase. The SETUP transmission occurs at the control endpoints only, and it is very similar to the process of transmitting the OUT packet. Enabling SETUP transmission requires setting the STAT\_TX bit and the STAT\_RX bit to 10 (NAK) in addition to initializing the DTOG\_TX bit to '1' and the DTOG\_RX bit to '0', respectively, and letting the application program decide whether the subsequent transmission is IN or OUT based on the corresponding fields of the SETUP packet. the control endpoint is notified of the transmission each time it occurs.

Any CTR\_RX interrupt must check the SETUP bit of the USB\_EPnR register to identify whether it is a normal OUT packet or a SETUP packet. The USB device should be able to determine the number of bytes and direction of the data phase transmission from the corresponding data in the SETUP packet, and be able to send a STALL packet to reject the data transmission in case of an error. Therefore, in the data phase, any unused direction should be set to STALL and the transmission of the last data packet of the data phase in its opposite direction should still be set to the NAK state when the transmission of the last data packet of the data phase is started so that even if the host computer changes the direction of the transmission immediately (enters into the status phase), it can still be maintained as a state waiting for the control transmission to end. At the end of a successful control transfer, the application program can change NAK to VALD, and then to STALL if there is an error in the control transfer. At this time, if the status packet is sent from the host to the device, the STATUS\_OUT bit (EP\_KIND in the USB\_EPnR register) should be set so that only then, if a non-zero-length data is received during a status transfer packets, only then will a transmission error be generated. After completing the status transfer phase, the application program should clear the STATUS\_OUT bit and set STAT\_RX to VALID to indicate that it is ready to receive a new command request, and STAT\_TX to NAK to indicate that it will not accept a request for a data transfer until the transmission of the next SETUP packet is complete.

The USB specification defines that a SETUP packet cannot be responded to with a non-ACK handshake packet, and if a SETUP packet fails to be transmitted, the next SETUP packet is triggered. Therefore, responding to a host's SETUP packet with a NAK or STALL packet is prohibited.

When the STAT\_RX bit is set to '01' (STALL) or '10' (NAK), if a SETUP packet is received, the USB module receives the packet, starts the data transfer requested by the packet, and sends back an ACK handshake packet. If the application program processes the previous CTR\_RX event the USB module receives another SETUP packet (i.e., CTR\_RX remains set), the USB module drops the received SETUP packet and does not answer any handshake packets as a way of simulating a reception error and forcing the host to send the SETUP packet again. This is done to avoid losing another SETUP packet transmission immediately following a CTR\_RX interrupt.

## 24.4.2 Double Buffered Endpoints

The USB standard not only defines different types of endpoints for different transmission modes, but also describes the system requirements needed for these data transfers. Among them, bulk endpoints are suitable for transmitting large batches of data between a host PC and a USB device because the host can utilize as much bandwidth as possible to transmit data in bulk in a single frame, resulting in increased transmission efficiency. However, when the USB device processes the previous data transmission and receives a new packet of data, it responds with a NAK packet, causing the host PC to keep retransmitting the same packet of data until the device responds with an ACK packet when it can process the data. Such retransmissions take up a lot of bandwidth and affect the bulk transfer rate, so a double buffering mechanism for bulk endpoints was introduced to improve the data transfer rate.

When using the double-buffering mechanism, a data transfer from a unidirectional endpoint will use both the receive and transmit data buffers for that endpoint. The data flip-flop bit is used to select which of the two buffers is currently being used, allowing the application program to operate on one buffer while the USB module accesses the other. For example, for an OUT packet transmission to a double-buffered batch endpoint, the USB module saves data from the PC host into one buffer while the application program can manipulate the data in the other buffer (the situation is the same for IN packets).

Since the management mechanism of switching buffers requires the use of all four buffer description table entries, which are used to represent the address pointers and buffer sizes of the two buffers in each direction, the USB\_EPnR register used to implement the double-buffered bulk endpoint must be configured as unidirectional. Therefore, only the STAT\_RX bit (as a double-buffered bulk receive endpoint) or the STAT\_TX bit (as a double-buffered bulk transmit endpoint) needs to be set. If a bi-directional double-buffered bulk endpoint is required, two USB\_EPnR registers must be used. In order to take advantage of double buffering as much as possible and to achieve high transfer rates, the flow control process for the double-buffered bulk endpoint is slightly different from that of the other endpoints. It sets the endpoint to the NAK state only when an access conflict occurs in the buffer, instead of setting the endpoint to the NAK state after every successful transfer.

The DTOG bit is used to identify the storage buffer currently used by the USB module. The buffer in the receive direction of the double-buffered bulk endpoint is identified by DTOG\_RX (bit 14 of the USB\_EPnR register), while the buffer in the transmit direction of the double-buffered bulk endpoint is identified by DTOG\_TX (bit 6 of the USB\_EPnR register). Also, the USB module needs to know which buffer is currently being used by the application program to avoid conflicts. Since there are 2 DTOG bits in the USB\_EPnR register and the USB module only uses one of them to identify the buffer used by the hardware, the application program can use the other bit to identify which buffer is currently being used, this new identification is called the SW\_BUF bit. The following table lists the relationship between the DTOG bit and the SW\_BUF bit of the USB\_EPnR register when the double-buffered bulk endpoint is implementing transmit and receive operations.

Table125 Double Buffered Batch Endpoint Buffer Identification Definitions

buffer identifier	act as a sending endpoint	serve as receiving endpoint
DTOG	DTOG_TX (bit 6 of the USB_EPnR register)	DTOG_RX (bit 14 of the USB_EPnR register)
SW_BUF	Bit 14 of the USB_EPnR register	Bit 6 of the USB_EPnR register

The buffer currently used by the USB module is identified by the DTOG bit, while the buffer used by the application is identified by the SW\_BUF bit, which are identified in the same way, as described in the following table.

Table126 Buffer usage identifiers for double-buffered batch endpoints

Endpoint Type	DTOG bit	SW_BUF bit	Buffers used by the USB module	Buffers used by the application
IN endpoint	0	1	ADDRn_TX_0/COUNTn_TX_0	ADDRn_TX_1/COUNTn_TX_1
	1	0	ADDRn_TX_1/COUNTn_TX_1	ADDRn_TX_0/COUNTn_TX_0
	0	0	None <sup>(1)</sup>	ADDRn_TX_0/COUNTn_TX_0
	1	1	None <sup>(1)</sup>	ADDRn_TX_0/COUNTn_TX_0
OUT endpoint	0	1	ADDRn_RX_0/COUNTn_RX_0	ADDRn_RX_1/COUNTn_RX_1
	1	0	ADDRn_RX_1/COUNTn_RX_1	ADDRn_RX_0/COUNTn_RX_0
	0	0	None <sup>(1)</sup>	ADDRn_RX_0/COUNTn_RX_0

	1	1	None <sup>(1)</sup>	ADDRn_RX_0/COUNTn_RX_0
--	---	---	---------------------	------------------------

(1). Endpoints are in the NAK state

A double-buffered batch endpoint can be set up in the following way:

- Set the EP\_TYPE bit of the USB\_EPnR register to '00' to define the endpoint as the bulk endpoint
- Set the EP\_KIND bit of the USB\_EPnR register to '1' to define the endpoint as a double buffered endpoint

The application program initializes the DTOG and SW\_BUF bits based on the buffer used at the start of the transfer; this takes into account the data rollover characteristics of these two bits. After setting the DBL\_BUF bit, the USB module will operate the flow control of the endpoints according to the double-buffered bulk endpoints after each completed transmission and continue until DBL\_BUF becomes invalid. At the end of each transmission, the CTR\_RX bit or CTR\_TX bit will be set to '1' depending on the direction of transmission of the endpoint. At the same time, the hardware will set the corresponding DTOG bit, completely independent of the software to implement the buffer swapping mechanism. After the DBL\_BUF bit is set, at the end of each transmission, the value of the STAT bit of the double-buffered batch endpoint will not be affected by the transmission process as in the case of the other types of endpoints, but it will always remain at '11' (valid). However, if a buffer access conflict occurs between the USB module and the application program (i.e., DTOG and SW\_BUF are the same value, see Table 126) when a transmission request is received for a new packet of data, the status bit will be set to '10' (NAK). The application responds to a CTR interrupt by first clearing the interrupt flag and then processing the completed data transfer. After the application accesses the buffer, it needs to flip the SW\_BUF bit to notify the USB module that the block has become available. As a result, the number of NAK packets transferred in a double-buffered batch is only determined by how fast or slow the application can process a single data transfer: if the data processing time is less than the time it takes to complete a single data transfer on the USB bus, no retransmission occurs, and at this point, the data transfer rate is limited only by the USB host.

The application can also disregard the special control flow for double-buffered batch endpoints and directly write any state other than '11' in the STAT bit of the corresponding USB\_EPnR register, in which case the USB module will execute the flow according to the written state and ignore the actual usage of the buffer.

### 24.4.3 Synchronous Transmission

The USB standard defines a transmission method that requires maintaining a fixed and precise data rate at full speed: synchronous transmission. Synchronous transfers are generally used for transmitting audio streams, compressed video streams, and other data that have strict data rate requirements. If an endpoint is defined as a "synchronous endpoint" during enumeration, the USB host allocates a fixed bandwidth for each frame and ensures that each frame carries exactly one IN packet or OUT packet (the type of packet is determined by the direction of transmission of the endpoint). In order to meet the bandwidth requirements, there is no error retransmission in synchronous transmission; this also means that synchronous transmission has no handshake protocol, i.e., no ACK packets are sent after a data packet is sent or received. Similarly, synchronous transmission only transmits packets with a PID (Packet ID) of DATA0, and does not use the data rollover mechanism.

It can be made a synchronous endpoint by setting the USB\_EPnR register EP\_TYPE to '10'. There is no handshaking mechanism for synchronous endpoints, and the STAT\_RX bit and STAT\_TX bit of the USB\_EPnR register can only be set to '00' (disabled) and '11' (valid), respectively, according to the instructions in the USB standard. Synchronous transfer simplifies software application development by implementing a double-buffering mechanism, which also uses two buffers to ensure that when the USB module uses one of the buffers, the application can access the other.

The buffers used by the USB module are identified by different DTOG bits depending on the direction of transmission. (The DTOG\_RX bit in the same register is used to identify the receive synchronization endpoint and the DTOG\_TX bit is used to identify the transmit synchronization endpoint.) See the table below.

Table127 Buffer usage identifiers for synchronization endpoints

Endpoint Type	DTOG bit value	Buffers used by the USB module	Buffers used by the application
IN endpoint	0	ADDRn_TX_0/COUNTn_TX_0	ADDRn_TX_1/COUNTn_TX_1
	1	ADDRn_TX_1/COUNTn_TX_1	ADDRn_TX_0/COUNTn_TX_0
OUT endpoint	0	ADDRn_RX_0/COUNTn_RX_0	ADDRn_RX_1/COUNTn_RX_1
	1	ADDRn_RX_1/COUNTn_RX_1	ADDRn_RX_0/COUNTn_RX_0

As with double-buffered bulk endpoints, a single USB\_EPnR register can only handle data transfers in one direction for synchronized endpoints; if a synchronized endpoint is required to be valid in both directions of transmission, two USB\_EPnR registers need to be used.

The application program needs to initialize the DTOG bit according to the data grouping of the first transmission; its value also needs to take into account the data flip-flop characteristics of the DTOG\_RX or DTOG\_TX bits. The CTR\_RX bit or the CTR\_TX bit of the USB\_EPnR register is bit at the completion of each transfer. At the same time, the associated DTOG bit is flipped by hardware, thus making the operation of the swap buffer completely independent of the application program. At the end of the transfer, the STAT\_RX or STAT\_TX bits do not change because synchronous transfers do not have a handshaking mechanism, so they do not require any flow control and are always set to '11' (valid). In synchronous transfers, even if a CRC error or buffer overflow occurs in the OUT packet, this transfer is still seen as correct and the CTR\_RX interrupt event can be triggered; however, the hardware sets the ERR bit of the USB\_ISTR register when a CRC error occurs, alerting the application program that the data may be corrupted.

## 24.4.4 Suspend/Resume Events

The USB standard defines a special device state, i.e., the suspend state, in which the average current consumption on the USB bus does not exceed 500 uA. This current limit is critical for bus-powered USB devices, while self-powered devices do not need to strictly adhere to such a current consumption limit. the USB host enters the suspend state with a 3 milliseconds without sending any signal flag. Normally the USB host sends a SOF every millisecond, and when the USB module detects three consecutive SOF packet loss events, it can determine that the host has sent a hang request, and then it will set the SUSP bit in the SB\_ISTR register to trigger a hang interrupt. The USB device enters into the hang state, and then it will be awakened by the Wake Up! After the USB device enters the suspend state, it will be woken up by the "Wake Up" sequence. The so-called "wake-up" sequence can be initiated by the USB host or triggered by the USB device itself; however, only the USB host can end the "wake-up" sequence. The suspended USB module must at least have the ability to detect the RESET signal, which it will perform as a normal reset operation.

The actual hang operation process is different for different USB devices, as different actions are required to reduce power consumption. The following describes a typical suspend operation, focusing on how the application responds to the USB module's SUSP signal.

1. Setting FSUSP in the USB\_CNTR register to '1' will cause the USB module to enter the suspend state. once the USB module enters the suspend state, detection of SOF stops immediately to avoid a new SUSP event while the USB is suspended.
2. Eliminates or reduces quiescent current consumption of modules other than USB modules.
3. Setting the LP\_MODE position of the USB\_CNTR register to '1' will eliminate the quiescent current consumption of the analog USB transceiver, but still detect the wake-up signal.
4. Optionally, the external oscillator and the device's PLL can be turned off to stop any activity within the device.

When a USB event occurs while the device is in the suspend state, the device will be woken up and the "wake up" routine will need to be called to restore the system clock, and USB data transfer. If the device is woken up by a USB reset operation, it should be ensured that the wake-up process does not take more than 10 milliseconds (see "USB Protocol Specification"). When the USB module is in the suspend state, the wake-up or reset event needs to clear the LP\_MODE bit in the USB\_CNTR register. Even though the wake-up event can immediately trigger a WKUP interrupt event, the interrupt service program handling the WKUP interrupt must be very careful due to the long delay time required to restore the system clock; in order to shorten the system wake-up time, it is recommended to write the wake-up code directly after the pending code, so that it can be executed quickly in the wake-up code after the system clock is restarted. To prevent or minimize accidental wake-up of the system by disturbances such as ESD (exit



from the pending mode is an asynchronous event), the data lines are filtered during the pending process with a filter width of approximately 70nS.

Here's how the wake-up operation works:

1. Starts the external oscillator and the PLL of the device (this item is optional).
2. Clear the FSUSP bit of the USB\_CNTR register.
3. The RXDP and RXDM bits of the USB\_FNR register can be used to determine what triggered the wake-up event, as shown in Table128 , which also lists both the actions that the software should take in each case. If desired, the end of a wake-up or reset event can be known by detecting when these two bits become '10' (representing the idle bus state). In addition, at the end of the reset event, the RESET bit of the USB\_ISTR register is set to '1', and if the RESET interrupt is enabled, a mid interrupt is generated. This interrupt should be handled as a normal reset operation.

Table128 Wake-up event detection

The state of [RXDP, RXDM]	wake-up event	Action to be performed by the application
00	reset (a dislocated joint, an electronic device etc)	not have
10	None (bus interference)	Revert to the suspended state
01	resume a pending (computer)	not have
11	Undefined value (bus interference)	Revert to the suspended state

The device may not be woken up by an event related to the USB module (e.g. a mouse movement can wake up the whole system). In this case, the wakeup sequence can be initiated by first setting the RESUME bit of the USB\_CNTR register to '1', and then clearing it to 0 between 1ms-15ms (this interval can be implemented using the ESOF interrupt, which occurs every 1ms during normal kernel operation).After the RESUME bit is cleared to 0, the wakeup process will be performed by the The host PC completes the process, and the RXDP and RXDM bits of the USB\_FNR register can be used to determine if the wakeup is complete.

*Notes: The RESUME bit can only be set when the USB module is set to the suspend state (setting the FSUSP bit of the USB\_CNTR register to '1').*

## 24.5 USB Register Description

There are three types of registers in the USB module::

- General-purpose class registers: interrupt registers and control registers
- Endpoint class registers: endpoint configuration registers and status registers
- Buffer description table class registers: registers used to determine the address where data packets are stored

The base address of the Buffer Description Table class register is specified by the USB\_BTABLER register, and the base address of all other registers is the USB module base address 0x4000 5C00. Since the APB1 bus is addressed in 32-bit terms, all 16-bit registers are address-aligned to 32-bit words. The same address alignment is used for the packet buffer memory starting at 0x4000 6000.

Refer to Section1 for some of the abbreviations used in register descriptions. These peripheral registers can be operated in half-word (16-bit) or word (32-bit) format.

## 24.5.1 General-Purpose Register

This set of registers is used to define the operating mode of the USB module, the handling of interrupts, the address of the device and the number to read the current frame.

### USB Control Register (USB\_CNTR)

Address offset: 0x40

Reset value: 0x0003

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTRM	PMAOVRM	ERRM	WKUPM	SUSPM	RESETM	SOFM	ESOFM	Reserved			RESUME	FSUSP	LPMODE	PDWN	FRES
rw	rw	rw	rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bit	notation	clarification
15	CTRM	<b>CTRM:</b> Correct transfer (CTR) interrupt mask bit (Correct transfer interrupt mask) 0: Correct Transfer (CTR) interrupt disabled 1: The Correct Transfer (CTR) interrupt enable generates an interrupt when the corresponding bit in the interrupt register is set to 1.
14	PMAOVRM	<b>PMAOVRM:</b> Packet memory area over/underrun interrupt mask bit (PMAOVRM) 0: PMAOVR interrupt disabled 1: The PMAOVR interrupt enable generates an interrupt when the corresponding bit in the interrupt register is set to 1.
13	ERRM	<b>ERRM:</b> Error interrupt mask bit (Error interrupt mask) 0: Error interrupt disabled 1: Error interrupt enable, generates an interrupt when the corresponding bit of the interrupt register is set to 1.
12	WKUPM	<b>WKUPM:</b> Wakeup interrupt mask bit (Wakeup interrupt mask) 0: Wake-up interrupt disabled 1: Wake-up interrupt enable generates an interrupt when the corresponding bit in the interrupt register is set to 1.
11	SUSPM	<b>SUSPM:</b> Suspend mode interrupt mask bit (Suspend mode interrupt mask) 0: Suspend (SUSP) Interrupt Disable 1: Suspend (SUSP) interrupt enable, which generates an interrupt when the corresponding bit of the interrupt register is set to one.
10	RESETM	<b>RESETM:</b> USB reset interrupt mask bit (USB reset interrupt mask) 0: USBRESET interrupt disabled 1: USBRESET interrupt enable generates an interrupt when the corresponding bit of the interrupt register is set to 1.
9	SOFM	<b>SOFM:</b> Start of frame interrupt mask bit (Start of frame interrupt mask) 0: SOF interrupt disabled 1: SOF interrupt enable, generates an interrupt when the corresponding bit of the interrupt register is set to 1.
8	ESOFM	<b>ESOFM:</b> Expected start of frame interrupt mask bit (ESOFM) 0: ESOF interrupt disabled 1: ESOF interrupt enable generates an interrupt when the corresponding bit in the interrupt register is set to 1.
7:5	Reserved	Reserved bits, always 0
4	RESUME	<b>RESUME:</b> Resume request Setting this bit will send a wake-up request to the PC host. According to the USB protocol, if this bit remains valid for 1ms to 15ms, the host computer will perform a wake-up operation on the USB module.
3	FSUSP	<b>FSUSP:</b> Force suspend The SUSP interrupt is triggered when there is no data communication on the USB bus for 3ms, at which time the software must set this bit. 0: Invalid 1: Entering suspend mode, the clock and static power consumption of the USB analog transceiver remain. If you need to enter the low power state (bus-powered class of devices), the application program needs to set FSUSP and then LP_MODE.
2	LP_MODE	<b>LP_MODE:</b> Low-power mode This mode is used to reduce power consumption in the USB suspend state. In this mode, all static power consumption is turned off except for the power supply from the external pull-up resistor, and the system clock will be stopped or reduced to a certain frequency to minimize power consumption. activity (wake-up events) on the USB bus will reset this bit (software can also reset this bit). 0: Non-low power mode 1: Low power mode
1	PDWN	<b>PDWN:</b> Power down mode (Power down) This mode is used to completely disable the USB module. When this bit is set, the USB module cannot be used. 0: Exit power-down mode 1: Entering power-down mode
0	FRES	<b>FRES:</b> Force USB Reset 0: Clear USB reset signal

		1: Forces a reset on the USB module, similar to the reset signal on the USB bus. the USB module will remain in the reset state until the software clears this bit. If the USB reset interrupt is enabled, a reset interrupt will be generated.
--	--	--

### USB Interrupt Status Register (USB\_ISTR)

Address offset: 0x44

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTR	PMAOVR	ERR	WKUP	SUSP	RESET	SOF	ESOF	Reserved			DIR	EP_ID[3:0]			
r	rcw0	rcw0	rcw0	rcw0	rcw0	rcw0	rcw0				r	r	r	r	r

This register contains status information for all interrupt sources for the application program to acknowledge the event that generated the interrupt request.

The high 8 bits of the register each indicate an interrupt source. These bits are set by hardware when the relevant event occurs, and if the corresponding bit on the USB\_CNTR register is also set, the corresponding interrupt will be generated. The interrupt service program needs to check each bit and clear the corresponding status bit after performing the necessary operation, otherwise the interrupt signal line will remain high and the same interrupt will be triggered again. If more than one interrupt flag is set at the same time, only one interrupt will be generated. The application program can handle transfer completion interrupts in different ways to minimize the latency of the interrupt response. After an endpoint successfully completes a transfer, the CTR bit is set by hardware and an interrupt is generated if the corresponding bit on USB\_CNTR is also set. The interrupt flags associated with the endpoint are independent of the CTRM bit in the USB\_CNTR register. These two interrupt flag bits will remain active until the application program clears the associated interrupt pending bit in the USB\_EPnR register (the CTR bit is a read-only bit). The USB module has two sources of interrupt requests:

High-priority USBIRQ: Used for interrupt requests for high-priority endpoints (synchronous and double-buffered bulk endpoints) that cannot be masked.

Low Priority USBIRQ: Used for other interrupt events, either low priority non-maskable interrupts or maskable interrupts identified by the high 8 bits of the USB\_ISTR register.

For endpoint-generated interrupts, the application program can use the DIR register and EP\_ID read-only bits to identify which endpoint generated the interrupt request and call the appropriate interrupt service program.

When handling multiple interrupt events occurring at the same time, the user can prioritize these events by checking the order of each bit of the USB\_ISTR register in the interrupt service program. The interrupt flag needs to be cleared after processing the interrupt of the corresponding bit. Upon completion of an interrupt service, another interrupt request will be generated to request the processing of the remaining interrupt events.

To avoid accidentally clearing certain bits, it is recommended to use a load instruction that writes '1' for all bits that do not need to be changed, and '0' for bits that need to be cleared. For this register, it is not recommended to use the read-modify-write flow, because between read and write operations, the hardware may need to set certain bits that will be cleared on write.

Each bit is described in detail below:

Bit	notation	clarification
15	CTR	CTR : Correct transfer This bit is set by hardware after an endpoint has completed a data transfer correctly. The application program can use the DIR and EP_ID bits to identify which endpoint completed a correct data transfer. This bit is read-only for the application
14	PMAOVR	PMAOVR : Packet memory area over/underrun This bit is set by hardware when the microcontroller has not responded to a request to access the USB packet buffer for an extended period of time. the USB module is normally in the This bit is set when: an ACK handshake packet is not sent during reception, or a bit-fill error occurs during transmission, in either case the host requests a data retransmission. In normal data transmission no PMAOVR Interrupt. Since all failed transfers will be retransmitted by the host initiating the retransmission, the application program can then accelerate the rest of the device's operation and prepare for the retransmission in the service program for this interrupt. However, this interrupt will not be generated in synchronous transfers (synchronous transfers do not support retransmission) so data may be lost. This bit can be read and written by the application program, but only write 0 is valid, write 1 is invalid.



13	ERR	<p><b>ERR:</b> Error. The hardware sets this bit on the occurrence of the following errors. NANS: No Answer. The host's answer timed out. CRC: Cyclic Redundancy Checksum Error. Error in CRC checksum in data or token packet. BST: Bit Stuffing Error A bit stuffing error was detected in the PID, data or CRC. FVIO: Frame format error. A non-standard frame is received (e.g., EOP appears at the wrong moment, wrong token, etc.). USB applications can usually ignore these errors as both the USB module and the host initiate a retransmission mechanism in the event of an error. The interrupt generated by this bit can be used in the development phase of the application program and can be used to monitor the transmission quality of the USB bus, identifying possible user errors (loose connection cable, severe environmental interference, damaged USB cable, etc.). This bit can be read and written by the application program, but only write 0 is valid, write 1 is invalid.</p>
12	WKUP	<p><b>WKUP:</b> Wakeup request (Wakeup) This bit will be set by hardware if a wake-up signal is detected when the USB module is in the suspend state. At this point the LP_MODE bit of the CTRLR register will be cleared to zero and USB_WAKEUP will be activated to notify the rest of the device (e.g., the wake-up unit) that the wake-up process will begin. This bit can be read and written by the application program, but only write 0 is valid, write 1 is invalid.</p>
11	SUSP	<p><b>SUSP :</b> Suspend mode request This bit is set by the hardware when there is no signal transmission on the USB line for more than 3ms to indicate a hang request from the USB bus. the hardware enables the detection of the hang signal immediately after a USB reset, but in hang mode (FSUSP = 1) the hardware will not detect the hang signal until the wake-up process is complete. This bit can be read and written by the application program, but only write 0 is valid, write 1 is invalid.</p>
10	RESET	<p><b>RESET :</b> USB reset request This bit is set by hardware when the USB module detects a USB reset signal input. At this point the USB module will reset the internal protocol state machine and trigger a reset interrupt in response to the reset signal if the interrupt is enabled. The transmit and receive portions of the USB module will be disabled until this bit is cleared. All configuration registers will not be reset unless the application program clears them. This is used to ensure that USB transfers can still be executed correctly immediately after a reset. However, the device's address and endpoint registers are reset by a USB reset. This bit can be read and written by the application program, but only write 0 is valid, write 1 is invalid.</p>
9	SOF	<p><b>SOF :</b> Start of frame flag (SOF) This bit is set by hardware when the USB module detects a SOF packet on the bus and marks the start of a new USB frame. The interrupt service program can complete the 1ms synchronization with the host by detecting the SOF event and correctly reading the registers as they are updated upon receipt of the SOF packet (this feature is very relevant when synchronizing transfers). This bit can be read and written by the application program, but only write 0 is valid, write 1 is invalid.</p>
8	ESOF	<p><b>ESOF :</b> Expected start of frame (Expected start of frame) This bit is set by hardware if the USB module does not receive the expected SOF packet. The host should send the SOF packet every millisecond, but if the USB module does not receive it, the hang-up timer will trigger this interrupt. If 3 consecutive ESOF interrupts occur, i.e. 3 consecutive SOF packets are not received, a SUSP interrupt will be generated. This bit will be set even if a SOF packet loss occurs while the hang-up timer is not latched. This bit can be read and written by the application program, but only write 0 is valid, write 1 is invalid.</p>
7:5	Reserved	Reserved bits, always 0
4	DIR	<p><b>DIR :</b> Direction of transaction This bit is written by the hardware according to the direction of transmission after an interrupt is generated by the completion of data transmission. If DIR = 0, the CTR_TX bit of the corresponding endpoint is set, signaling the completion of the transmission of an IN packet (data transfer from the USB module to the PC host). If DIR = 1, the CTR_RX bit of the corresponding endpoint is set, signaling the completion of the transmission of an OUT packet (data transfer from the PC host to the USB module). If the CTR_TX bit is also set, it signals the presence of both a pending OUT packet and an IN packet. The application program can use this information to access the operation corresponding to the USB_EPnR bit, which indicates information about the direction of the pending interrupt transmission. This bit is read-only</p>

3:0	EP_ID[3:0]	<b>EP_ID[3:0]:</b> End point ID (End point Identifier) This bit is written by the hardware according to the endpoint number of the requested interrupt after the USB module completes the data transfer and generates an interrupt. If there are multiple endpoints requesting interrupts at the same time, the hardware writes the endpoint number with the highest priority. The priority of the endpoints is defined in the following way: synchronous endpoints and double-buffered bulk endpoints have high priority, and other endpoints have low priority. If multiple endpoints of the same priority request an interrupt, the priority is determined based on the endpoint number, i.e., endpoint 0 has the highest priority, and the smaller the endpoint number, the higher the priority. Application program Interrupt requests from endpoints can be processed sequentially with the priority strategy described above. This bit is read-only.
-----	------------	--

### USB Frame Number Register (USB\_FNR)

Address offset: 0x48

Reset value: 0x0XXX , X represents undefined value

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXDP	RXDM	LCK	LSOF[1:0]	FN [10:0]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit	notation	clarification
15	RXDP	<b>RXDP:</b> D + status bit (Receive data + line status) This bit is used to observe the status of the USB <sup>D+</sup> data line and can be used to detect the occurrence of a wake-up condition in a hung state.
14	RXDM	<b>RXDM:</b> D-status bit (Receive data-line status) This bit is used to observe the status of the USB <sup>D-</sup> data line and can be used to detect the occurrence of a wake-up condition in a hung state.
13	LCK	<b>LCK:</b> Locked bit The USB module detects SOF packets at the end of a reset or wakeup sequence, and if at least 2 SOF packets are detected consecutively, the hardware will set this bit. Once this bit is latched, the frame counter stops counting and waits until the USB module resets or the bus hangs before resuming counting.
12:11	LSOF[1:0]	<b>LSOF[1:0]:</b> Loss of head of frame flag bit (Lost SOF) When an ESOF event occurs, the hardware writes the number of lost SOF packets to this bit. The pin clears this bit if the SOF packet is received again.
10:0	FN [10:0]	<b>FN[10:0]:</b> Frame number This section records the 11-bit frame number from the latest received SOF packet. The frame number is self-added for each frame sent by the host, which is significant for synchronized transmissions. This section is updated when a SOF interrupt occurs.

### USB Device Address Register (USB\_DADDR)

Address offset: 0x4C

Reset value: 0x0000

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

Bit	notation	clarification
15:8	Reserved	Reserved bits, always 0
7	EF	<b>EF:</b> USB module enable bit (Enable function) This bit is set by the application program when the USB module needs to be enabled. If this bit is 0, the USB module stops working, ignores all register settings, and does not respond to any USB communication.
6:0	ADD[6:0]	<b>ADD[6:0]:</b> device address (device address) This bit records the address value assigned to the USB device by the USB host during the enumeration process. This address value and endpoint address (EA) must match the address information in the USB token grouping in order for proper USB transfers to occur at the specified endpoint.

## USB Packet Buffer Description Table Address Register (USB\_BTABLE)

Address offset: 0x50

Reset value: 0x0000

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

Bit	notation	clarification
15	BTABLE[15:3]	<b>BTABLE[15:3]:</b> Buffer table This bit records the starting address of the packet buffer description table. The packet buffer description table is used to indicate the packet buffer address and size for each endpoint, aligned by 8 bytes (i.e., the lowest 3 bits are 000). At the beginning of each transmission, the USB module reads the packet buffer description table corresponding to the corresponding endpoint to obtain the buffer address and size information.
14	Reserved	Reserved bit, set to 0 by hardware

## 24.5.2 Endpoint Registers

The number of endpoint registers is determined by the number of endpoints supported by the USB module, which supports up to 8 bidirectional endpoints. Each USB device must support one control endpoint, and the address (EA bit) of the control endpoint must be 0. Different endpoints must use different endpoint numbers, otherwise the state of the endpoints is variable. Each endpoint has a corresponding USB\_EPnR register, which is used to store various status information of the endpoint.

### USB endpoint n register (USB\_EPnR), n=[0..7]

Address Offset: 0x00 to 0x1C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTR_RX	DTOG_RX	STAT_RX	SETUP	EPTYPE[1:0]	EP_KIND	CTR_TX	DTOG_TX	STAT_TX[1:0]	EA[3:0]						
rc_w0	t	t	t	r	rw	rw	rw	rc_w0	t	t	t	rw	rw	rw	rw

The USB module will reset when it receives the USB bus reset signal, or the FRES position bit in the CTRLR register. This register is reset except for the CTR\_RX and CTR\_TX bits which remain unchanged to handle the immediately following USB transfer. Each endpoint corresponds to a USB\_EPnR register, where n is the endpoint address, i.e., the endpoint ID number.

Performing read-modify-write operations should be avoided for such registers because between read and write operations, the hardware may set certain bits that are modified on write, causing the application program to miss the corresponding operation. Therefore, these bits have a write-invalid value, and it is recommended to modify these registers with the Load instruction so that the application does not modify bits that do not need to be modified

Bit	notation	clarification
15	CTR_RX	<b>CTR_RX:</b> Correct Transfer for reception flag bit (Correct Transfer for reception) This bit is set by hardware when an OUT or SETUP packet is received correctly, and the application program can only clear this bit. If the CTRM bit is set, the corresponding interrupt is generated. Whether an OUT packet or a SETUP packet is received can be determined by the following description of the SETUP bit determines. Packets ending in NAK or STALL and transmissions with errors do not cause this bit to be set because no data is actually transmitted. This bit can be read and written by the application program, but only write 0 is valid, write 1 is invalid.
14	DTOG_RX	<b>DTOG_RX:</b> Data Toggle, for reception transfers For non-synchronized endpoints, this bit is set by hardware and is used to mark the Toggle bit for the next data packet you wish to receive (0=DATA0, 1=DATA1). After receiving a data packet with the correct PID (Packet ID), the USB module sends the ACK handshake packet and flips this bit. For control endpoints, hardware clears this bit upon receipt of the SETUP packet. For double-buffered endpoints, this bit is also used to support double-buffer swapping (see 21.4.3 Double-Buffered Endpoints). For synchronous endpoints, since only DATA0 is sent, this bit is only used to support double-buffer swapping (see 21.4.4 Synchronous Transmission) without flip-flopping. Synchronous transfers do not require handshake packets, so the hardware sets this

		<p>bit as soon as it receives the data packet.</p> <p>The application can initialize this bit (which is required for non-control endpoints) or flip this bit for special purposes.</p> <p>This bit is readable and writable by the application program, but a write of 0 is invalid and a write of 1 flips this bit.</p>
13:12	STAT_RX[1:0]	<p><b>STAT_RX[1:0]:</b> Status bits, for reception transfers</p> <p>This bit is used to indicate the current state of the endpoint.</p> <p>Table129 lists all the states of the endpoint. When a proper OUT or SETUP data transfer is completed (CTR_RX=1), the hardware automatically sets this bit to the NAK state, allowing the application program enough time to respond to the next data packet after processing the currently transferred data.</p> <p>For double-buffered bulk endpoints, the transmission state (refer to 21.4.3 Double-Buffered Endpoints) is controlled according to the state of the buffer used, since a special transmission flow control strategy is used).</p> <p>For synchronized endpoints, the hardware does not set this bit after a proper transfer since the endpoint state can only be valid or disabled. If the application program sets this bit to STALL or NAK, the USB module responds with an undefined operation.</p> <p>This bit is readable and writable by the application program, but a write of 0 is invalid and a write of 1 flips this bit.</p>
11	SETUP	<p><b>SETUP:</b> SETUP packet transmission completed flag bit (Setup transaction completed)</p> <p>This bit is set by hardware after the USB module receives a correct SETUP packet and is only used by the control endpoint. After reception is complete (CTR_RX=1), the application program needs to test this bit to determine if the completed transmission was a SETUP packet. To prevent the interrupt service program from modifying this bit with the next token grouping while the SETUP grouping is being processed, only if CTR_RX is 0, this bit</p> <p>It can only be modified; it cannot be modified when CTR_RX is 1. This bit is read-only by the application program.</p>
10:9	EP_TPYE[1:0]	<p><b>EP_TPYE[1:0]:</b> End point type bits (End point type)</p> <p>This bit is used to indicate the current type of the endpoint, all endpoint types are listed in</p> <p>Table130 . All USB devices are required to contain a control endpoint with address 0, and can have control endpoints with other addresses if required. Only the control endpoint will have</p> <p>SETUP transmissions are ignored by other types of endpoints.SETUP transmissions cannot be responded to with a NAK or STALL packet.If the control endpoint is in the NAK state when it receives a SETUP packet, the USB module will not respond to the packet and a receive error will occur.</p> <p>Error. If the control endpoint is in the STALL state, the SETUP packet is received correctly, the data is transmitted correctly, and a correct transmission completion interrupt is generated. The OUT packet of the control endpoint is installed to be handled in the same manner as a normal endpoint.</p> <p>Batch endpoints and interrupt endpoints are handled in a very similar manner, differing only in the handling of the EP_KIND bit. Refer to 21.4.4 Synchronous Transmission for the usage of synchronous endpoints.</p>
8	EP_KIND	<p><b>EP_KIND:</b> End point special type bit (End point kind)</p> <p>This bit needs to be used in conjunction with the EP_TYPE bit as defined in Table131 .</p> <p><b>DBL_BUF:</b> Setting this bit by the application enables double buffering of bulk endpoints. See 21.4.3 Double Buffered Endpoints for details.</p> <p><b>STATUS_OUT:</b> The application program sets this bit to indicate that the USB device expects the host to send a status packet, at which point the device responds with a STALL packet for any packet that is not 0 in length. This feature is used only for control endpoints and is useful for providing application program detection of protocol layer errors. If the STATUS_OUT bit is cleared, the OUT packet can contain data of any length.</p>
7	CTR_TX	<p><b>CTR_TX:</b> Correct transfer for transmission flag bit (Correct transfer for transmission)</p> <p>This bit is set by hardware after a correct IN packet transmission has been completed. If the CTRM bit has been set, a corresponding center</p> <p>Disconnect. The application program needs to clear this bit after processing the event. At the end of an IN packet, if the host responds to NAK or STALL then this bit is not set because the data transfer was not successful.</p> <p>This bit can be read and written by the application program, but write 0 is valid and write 1 is invalid.</p>
6	DTOG_RX	<p><b>DTOG_RX:</b> Transmission Data Toggle bit (Data Toggle, for transmission transfers)</p> <p>For non-synchronized endpoints, this bit is used to indicate the Toggle bit for the next data packet to be transmitted (0 = DATA0.</p> <p>1=DATA1). After a successfully transmitted data packet, the USB module flips this bit if it receives an ACK packet from the host. For control endpoints, the USB module will set this bit after receiving the correct SETUPPID.</p> <p>For double-buffered endpoints, this bit can also be used to support packet buffer switching (see 21.4.3 Double-Buffered Endpoints).</p> <p>For synchronous endpoints, this bit is only used to support packet buffer swapping since only DATA0 is transmitted (see 21.4.4 Synchronous Transmission). Since</p>

		<p>synchronous transfers do not require handshaking packets, the hardware sets this bit as soon as a data packet is received.</p> <p>The application program can initialize this bit (necessary to initialize this bit for non-control endpoints), or it can set this bit for special purposes.</p> <p>This bit is readable and writable by the application program, but a write of 0 is invalid and a write of 1 flips this bit.</p>
5:4	STAT_TX[1:0]	<p><b>STAT_TX[1:0]:</b> status bits, for transmission transfers</p> <p>This bit identifies the current state of the endpoint, and Table132 lists all states. The application program can flip these bits to initialize the status information. After a correctly completed IN packet transmission (CTR_TX=1), the hardware automatically sets this bit to the NAK state, ensuring that the application program has enough time to prepare the data to respond to subsequent data transmissions.</p> <p>For double-buffered bulk endpoints, the state of the transmission is controlled based on the state of the buffer, due to the use of a special transmission flow control strategy (refer to 21.4.3 Double-Buffered Endpoints).</p> <p>For synchronized endpoints, the hardware does not change the state of the endpoint at the end of a data transfer since the state of the endpoint can only be valid or disabled. If the application program sets this bit to STALL or NAK, subsequent operation of the USB module is undefined.</p> <p>This bit is readable and writable by the application program, but a write of 0 is invalid and a write of 1 flips this bit.</p>
3:0	EA[3:0]	<p><b>EA[3:0]:</b> End point address (End point address)</p> <p>The application program must set these 4 bits to define an address for an endpoint before enabling it.</p>

Table129 Receive Status Codes

STAT_RX[1:0]	descriptive
00	DISABLED: The endpoint ignores all incoming requests.
01	STALL: The endpoint responds to all incoming requests with the STALL grouping.
10	NAK: The endpoint responds to all incoming requests with the NAK packet.
11	VALID: The endpoint is available for reception.

Table130 Endpoint Type Codes

EP_TYPE[1:0]	descriptive
00	BULK: Bulk Endpoint
01	CONTROL: Control Endpoints
10	ISO: Synchronized Endpoints
11	INTERRUPT: Interrupt Endpoint

Table131 Endpoint Special Type Definitions

EP_TYPE[1:0]	EP_KIND meaning
00	BULK DBL_BUF: double buffered endpoints
01	CONTROL STATUS_OUT
10	ISO unused
11	INTERRUPT unused

Table132 Transmission Status Codes

STAT_RX[1:0]	descriptive
00	DISABLED: The endpoint ignores all send requests.
01	STALL: The endpoint responds to all send requests with the STALL packet.
10	NAK: The endpoint responds to all send requests with NAK packets.
11	VALID: The endpoint can be used for sending.

## 24.5.3 Buffer Description Table

Although the buffer description table is located within the packet buffer, it can still be thought of as special registers to configure the address and size of the packet buffer shared by the USB module and the microcontroller core. Since the APB1 bus is addressed in 32 bits, all packet buffer addresses use 32-bit aligned addresses instead of the addresses used by the USB\_BTABLER register and the buffer description table. Two address representations are described below: one used by the application program to access the packet buffer, and the other relative to the USB module's local address. The packet buffer address for use by the application program needs to be multiplied by two to get the true address of the buffer in the microcontroller. The first address of the grouping buffer is 0x4000 6000. The following describes the buffer description table associated with the USB\_EPnR register. Refer to the 24.4.1 section for a complete description of the packet buffer and the usage of the buffer description table.

### Transmit Buffer Address Register n (USB\_ADDRn\_TX)

Address offset: [USB\_BTABLER]+n× 16

USB local address: [USB\_BTABLER]+n 8×

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRn_TX[15:1]															-
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit	notation	clarification													
15:1	ADDRn_TX [15:1]	ADDRn_TX[15:1]: transmission buffer address (Transmission buffer address) This bit records the start address of the buffer where the data to be sent is located when the next IN packet is received.													
0	-	Because the address of the packet buffer must be word-aligned, this bit must be '0'.													

### Transmit data byte count register n (USB\_COUNTn\_TX)

Address offset: [USB\_BTABLER]+n× 16

USB local address: [USB\_BTABLER]+n× 8+2

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						COUNTn_TX[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit	notation	clarification													
15:10	Reserved	Since the maximum data grouping supported by the USB module is 1023 bytes, the USB module ignores these bits.													
9:0	COUNTn_TX [9:0]	COUNTn_TX[9:0] : Transmission byte count This bit records the number of data bytes to be transmitted when the next IN packet is received.													

Notes: There are two USB\_COUNTn\_TX registers for the double buffer and synchronized IN endpoints: USB\_COUNTn\_TX\_1 and USB\_COUNTn\_TX\_0, respectively, with the following contents:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
-						COUNTn_TX_1[9:0]									
						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
-						COUNTn_TX_0[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

### Receive Buffer Address Register n (USB\_ADDRn\_RX)

Address offset: [USB\_BTABLE]+n× 16+8

USB local address: [USB\_BTABLE]+n× 8+4

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

Bit	notation	clarification
15:1	ADDRn_RX [15:1]	ADDRn_RX[15:1]: reception buffer address (Reception buffer address) This bit records the start address of the buffer used to hold data when the next OUT or SETUP packet is received.
0	-	Because the address of the packet buffer must be word-aligned, this bit must be '0'.

### Receive data byte count register n (USB\_COUNTn\_RX)

Address offset: [USB\_BTABLE]+n× 16+12

USB local address: [USB\_BTABLE]+n× 8+6

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

This register is used to hold the two parameters to be used when receiving packets. The high 6 bits define the size of the receive packet buffer so that the USB module can detect buffer overflow. The low 10 bits are used for the USB module to record the actual number of bytes received. Due to the limitation of the number of valid bits, the size of the buffer is represented by the number of storage blocks allocated to it, and the size of the storage blocks is determined by the required buffer size. The size of the buffer is defined during device enumeration and is expressed by the parameter maxPacketSize of the endpoint descriptor. (Please refer to “USB2.0 Protocol Specification” for more information.)

Bit	notation	clarification
15	BL_SIZE	<b>BL_SIZE</b> : storage block size (Block size) This bit is used to define the size of the storage block that determines the buffer size. If BL_SIZE=0, the size of the storage block is 2 bytes, so the size range of the packet buffer that can be allocated is 2-62 bytes. If BL_SIZE=1, the size of the storage block is 32 bytes, so the size range of the packet buffer that can be allocated is 32-512 bytes, which meets the maximum packet length limit defined by the USB protocol.
14:10	NUM_BLOCK[4:0]	<b>NUM_BLOCK[4:0]</b> : the number of blocks stored (Number of blocks) This bit is used to record the number of storage blocks allocated, which determines the size of the final packet buffer used. Refer to Table133 for details.
9:0	COUNTn_RX[9:0]	<b>COUNTn_RX[9:0]</b> : received byte count (Reception byte count) This bit is written by the USB module to record the actual byte count of the latest OUT or SETUP packet received by the endpoint.

Notes: There are two USB\_COUNTn\_RX registers for the double buffer and synchronized IN endpoints: USB\_COUNTn\_RX\_1 and USB\_COUNTn\_TX\_0, respectively, with the following contents:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BLSIZE_1	NUM_BLOCK_1[4:0]					COUNTn_RX_1[9:0]									
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
BLSIZE_0	num_block_0[4:0]					COUNTn_RX_0[9:0]									
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Table133 Definition of Grouping Buffer Sizes

The value of NUM_BLOCK[4:0].	Packet buffer size with BL_SIZE=0	Packet buffer size when BL_SIZE=1
00000	Not allowed to use	32 bytes
00001	2 bytes	64 bytes
00010	4 bytes	96 bytes



00011	6 bytes	128 bytes
...	...	...
01111	30 bytes	512 bytes
10000	32 bytes	Reserved
10001	34 bytes	Reserved
10010	36 bytes	Reserved
...	...	...
11110	60 bytes	Reserved
11111	62 bytes	Reserved

## 24.5.4 USB Register Image

Table134 USB Register Image and Reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
00h	USB_EP0R	Reserved																CTR_RX	DTOG_RX	STAT_RX [1:0]	SETUP	EP_TYPE [1:0]	EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]	EA[3:0]									
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
04h	USB_EP1R	Reserved																CTR_RX	DTOG_RX	STAT_RX [1:0]	SETUP	EP_TYPE [1:0]	EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]	EA[3:0]									
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
08h	USB_EP2R	Reserved																CTR_RX	DTOG_RX	STAT_RX [1:0]	SETUP	EP_TYPE [1:0]	EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]	EA[3:0]									
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0Ch	USB_EP3R	Reserved																CTR_RX	DTOG_RX	STAT_RX [1:0]	SETUP	EP_TYPE [1:0]	EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]	EA[3:0]									
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10h	USB_EP4R	Reserved																CTR_RX	DTOG_RX	STAT_RX [1:0]	SETUP	EP_TYPE [1:0]	EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]	EA[3:0]									
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14h	USB_EP5R	Reserved																CTR_RX	DTOG_RX	STAT_RX [1:0]	SETUP	EP_TYPE [1:0]	EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]	EA[3:0]									
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18h	USB_EP6R	Reserved																CTR_RX	DTOG_RX	STAT_RX [1:0]	SETUP	EP_TYPE [1:0]	EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]	EA[3:0]									
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1Ch	USB_EP7R	Reserved																CTR_RX	DTOG_RX	STAT_RX [1:0]	SETUP	EP_TYPE [1:0]	EP_KIND	CTR_TX	DTOG_TX	STAT_TX [1:0]	EA[3:0]									
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
20h~3Fh	Reserved																																
40h	USB_CNTR	Reserved																CTRM	PMAOVRM	ERRM	WKUPM	SUSPM	RESETM	SOFM	ESOFM	Reserved	RESUME	FSUSP	LPMODE	PDWN	FRES		
	Reset value																	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0
44h	USB_ISTR	Reserved																CTR	PMAOVR	ERR	WKUP	SUSP	RESET	SOF	ESOF	Reserved	DIR	EP_ID[3:0]					
	Reset value																	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0
48h	USB_FNR	Reserved																RXDP	RXDM	LCK	LSOF [1:0]	FN [10:0]											
	Reset value																	0	0	0	0	0	x	x	x	x	x	x	x	x	x	x	x
4Ch	USB_DADDR	Reserved																Reserved						EF	ADD[6:0]								
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
50h	USB_BTABLER	Reserved																BTABLER[15:3]										Reserved					
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

See Table 1 for register start addresses.

## 25 Controller Local Area Network (bxCAN)

### 25.1 Introduction to bxCAN

bxCAN stands for Basic Extended and supports CAN protocols 2.0A and 2.0B. It is designed to efficiently process large numbers of incoming messages with minimal CPU load. It also supports prioritization requirements for message sending (the priority feature is software configurable). For safety-critical applications, bxCAN provides all the hardware features required to support time-triggered communication modes.

### 25.2 bxCAN Key Features

- Supports CAN protocol 2.0A and 2.0B active modes
- Baud rates up to 1 Mbit/s
- Supports time-triggered communication function
- dispatch
  - 3 sending e-mail addresses
  - Priority characteristics of sent messages are software configurable
  - Record the timestamp of the moment when the SOF was sent
- reception (of transmitted signal)
  - 2 receive FIFOs at 3 levels of depth
  - Variable filter sets: W55MH32 has 14 filter sets
  - identifier list
  - FIFO overflow handling is configurable
  - Record the timestamp of the moment of receiving the SOF
- Time-triggered communication mode
  - Disable automatic retransmission mode
  - 16-bit free-running timer
  - Timestamps can be sent in the last 2 data bytes
- managerial
  - Interrupts can be masked
  - Mailboxes take up a separate address space, facilitating software efficiency

*Notes: USB and CAN share a dedicated 512-byte SRAM memory for data transmission.  
The USB and CAN can be used together in an application but not at the same time.*

### 25.3 General Description of bxCAN

In today's CAN applications, where the number of nodes in the CAN network is increasing and multiple CANs are often connected via gateways, the number of messages (which each node needs to process) in the entire CAN network has increased dramatically. In addition to application layer messages, network management and diagnostic messages have been introduced.

- Need an enhanced filtering mechanism to handle various types of messages
- In addition, application layer tasks require more CPU time, so the level of real-time response required for message reception needs to be mitigated.

- The scheme of receiving FIFOs allows that the CPU spends a long time processing application layer tasks without losing messages.

High-level protocol software built on top of the underlying CAN driver, requiring efficient interfacing with the CAN controller.

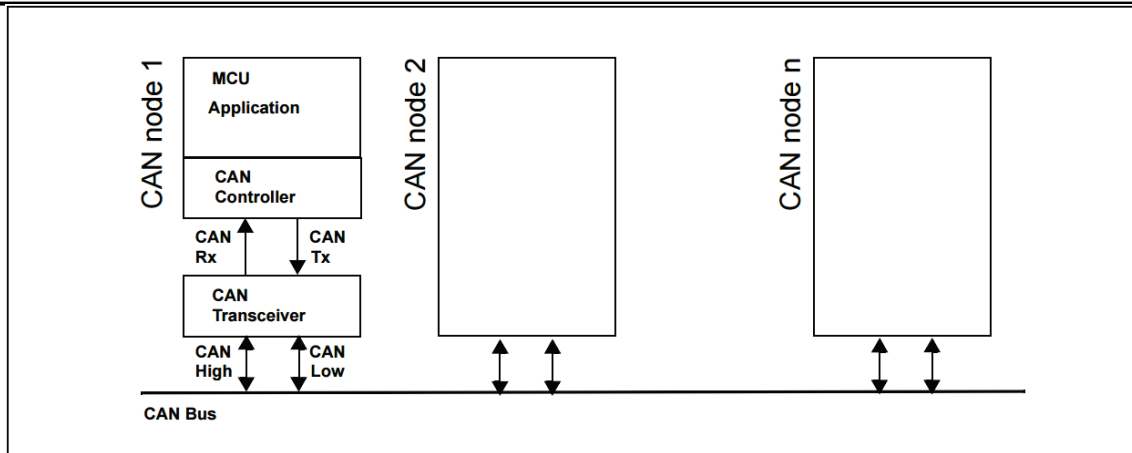


Figure196 CAN network topology

### 25.3.1 CAN2.0B Active Kernel

The bxCAN module receives and sends CAN messages fully automatically; both standard (11-bit) and extended (29-bit) identifiers are fully supported.

### 25.3.2 Control, Status and Configuration Registers

The application program, through these registers, can:

- Configure CAN parameters such as baud rate
- Request to send message
- Processing message reception
- Managing interruptions
- Getting Diagnostic Information

### 25.3.3 Send Email

There are a total of 3 sending mailboxes for the software to send messages. The send scheduler decides which mailbox is sent first based on priority.

### 25.3.4 Receiving Filter

The W55MH32 has 14 bit-width variable/configurable identifier filter sets.

#### Receive FIFO

There are 2 receive FIFOs, each of which can hold 3 full messages. They are managed entirely by hardware.

## 25.4 bxCAN operating mode

The bxCAN has 3 main operating modes: initialization, normal and sleep mode. After a hardware reset, bxCAN operates in sleep mode to save power while the internal pull-up resistor on the CANTX pin is activated. The software can request the bxCAN to enter the initialization or sleep modes by setting the INRQ or SLEEP position '1' to the CAN\_MCR register. Once in initialization or sleep mode, the bxCAN acknowledges the INAK or SLAK bit '1' of the CAN\_MSR register while the internal pull-up resistor is disabled. When both the INAK and SLAK bits are '0', the bxCAN is in normal mode. Before entering normal mode, the bxCAN must get in sync with the CAN bus; to get synchronized, the bxCAN waits for the CAN bus to reach an idle state, i.e., 11 consecutive implicit bits are monitored on the CANRX pin.

### 25.4.1 Initialization Mode

Software initialization should be performed while the hardware is in initialization mode. Set the INRQ bit of the CAN\_MCR register to '1' to request the bxCAN to enter the initialization mode, and then wait for the hardware to acknowledge the INAK position '1' of the CAN\_MSR register.

---

Clearing the INRQ bit of the CAN\_MCR register to '0' requests the bxCAN to exit the initialization mode, and the exit from the initialization mode is confirmed when the hardware clears '0' to the INAK bit of the CAN\_MSR register.

When bxCAN is in initialization mode, reception and transmission of messages are disabled and the CANTX pin outputs a recessive bit (high). The entry of the initialization mode does not change the configuration registers.

The software initialization of the bxCAN consists of at least 2 registers, the bit time characteristic (CAN\_BTR) and the control (CAN\_MCR).

Before initializing the bxCAN's filter set (mode, bit width, FIFO association, activation and filter value), the software sets '1' to the FINIT bit of the CAN\_FMR register. Initialization of the filters can be done in uninitialized mode.

*Notes: When FINIT=1, reception of the message is disabled.*

*It is possible to clear '0' to the filter activation bit (in CAN\_FA1R) and then modify the value of the corresponding filter.*

*If the filter group is not in use, then it should be left inactive (keeping its FACT bit in the clear '0' state).*

## 25.4.2 Normal Mode

After the initialization is complete, the software should allow the hardware to enter normal mode in order to receive and send messages properly. The software can request to enter the normal mode from the initialization mode by clearing '0' to the INRQ bit of the CAN\_MCR register, and then it has to wait for the hardware's acknowledgement of the INAK position '1' of the CAN\_MSR register. After synchronization with the CAN bus, i.e., monitoring 11 consecutive implicit bits (equivalent to bus idle) on the CANRX pin, the bxCAN can receive and send messages normally.

The setting of the initial value of the filter does not need to be done in initialization mode, but must be done while it is in the inactive state (the corresponding FACT bit is 0). The setting of the filter's bit width and mode, on the other hand, must be done before it enters normal mode in initialization mode.

## 25.4.3 Sleep Mode (Low Power Consumption)

The bxCAN can operate in a low power sleep mode. Software requests entry into this mode by applying a SLEEP position '1' to the CAN\_MCR register. In this mode, the bxCAN clock is stopped, but software can still access the mailbox registers.

When the bxCAN is in sleep mode, software must clear the INRQ bit of the CAN\_MCR register by '1' and also clear the SLEEP bit by '0' in order to enter initialization mode.

There are 2 ways to wake up (exit sleep mode) bxCAN: by clearing the SLEEP bit '1' by software, or by hardware detecting activity on the CAN bus.

If the AWUM bit of the CAN\_MCR register is '1', the hardware automatically clears the SLEEP bit to '0' to wake up the bxCAN as soon as CAN bus activity is detected, and if the AWUM bit of the CAN\_MCR register is '0', the software has to clear the SLEEP bit in the wake-up interrupt in order to exit from the sleep state.

*Notes: If the wakeup interrupt is allowed (WKUIE bit of the CAN\_IER register is '1'), then a wakeup interrupt is generated as soon as CAN bus activity is detected, regardless of whether the hardware automatically wakes up bxCAN.*

After clearing '0' to the SLEEP bit, the exit from sleep mode must be synchronized with the CAN bus, refer to Figure 197 : bxCAN operating modes. The exit from sleep mode is confirmed when the hardware clears '0' to the SLAK bit.

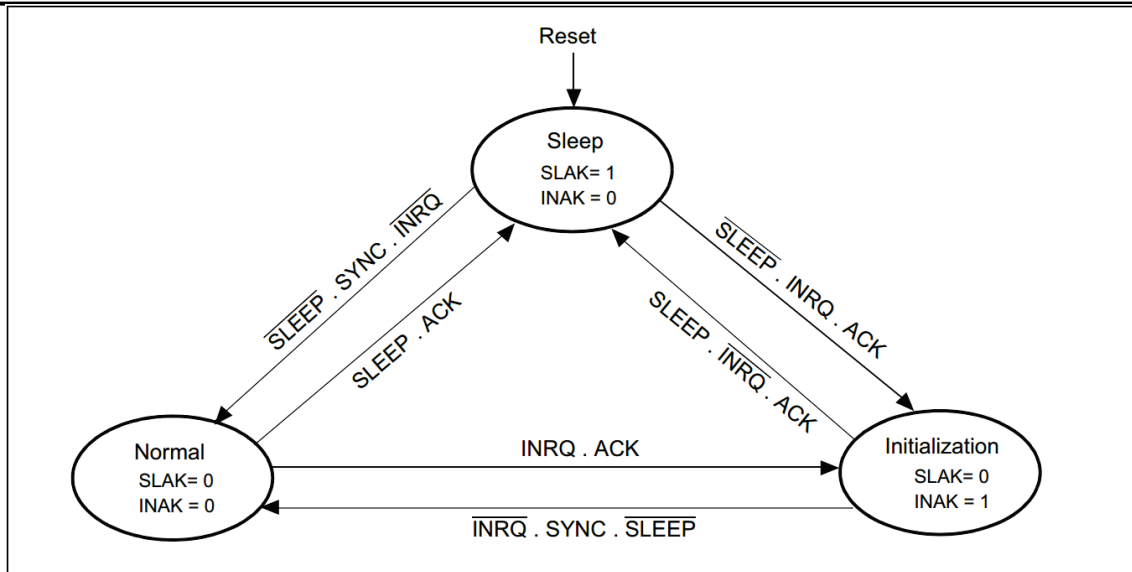


Figure197 bxCAN operating mode

Notes: 1 ACK = hardware responds to a sleep or initialization request with a status of INAK or SLAK position 1 of the CAN\_MSR register.  
2 SYNC=bxCAN waits for the CAN bus to become idle, i.e., 11 consecutive implicit bits are detected on the CANRX pin

## 25.5 Test Pattern

A test mode is selected by setting the SILM and/or LBKM position '1' of the CAN\_BTR register. These 2 bits can only be modified in the initialization mode. After selecting a test mode, the software needs to clear '0' to the INRQ bit of the CAN\_MCR register to actually enter the test mode.

### 25.5.1 Silent Mode

The silent mode is selected by SILM position '1' to the CAN\_BTR register. In silent mode, bxCAN can receive data frames and remote frames normally, but can only issue implicit bits and cannot actually send messages. If bxCAN needs to issue explicit bits (acknowledgement bits, overload flags, active error flags), such explicit bits are internally picked up and can be detected by the CAN core, while the CAN bus remains unaffected and remains in the implicit bit state. Therefore, the silent mode is usually used to analyze the activity of the CAN bus without affecting the bus - explicit bits (acknowledgement bits, error frames) are not actually sent to the bus.

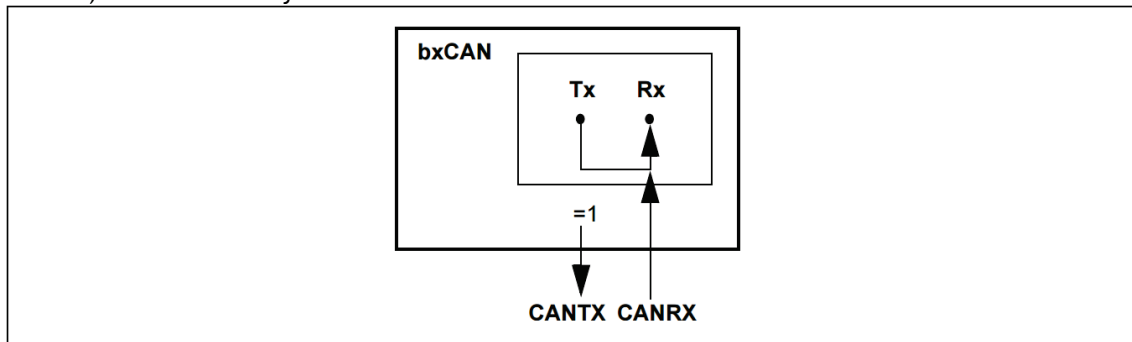


Figure198 bxCAN operating in silent mode

## 25.5.2 Loopback Mode

The loopback mode is selected by setting the LBKM position '1' of the CAN\_BTR register. In loopback mode, bxCAN treats the transmitted message as received and saves (if it can be filtered by reception) it in the receive mailbox.

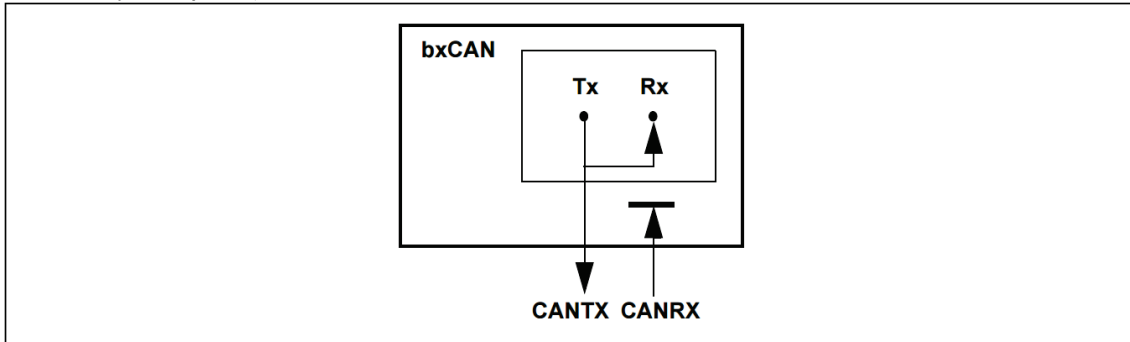


Figure199 bxCAN operating in loopback mode

The loopback mode can be used for self-testing. To avoid external influences, the CAN kernel ignores acknowledgement errors in loopback mode (in the data/far The acknowledge bit moment of the program frame is not tested for dominant bits). In loopback mode, bxCAN internally feeds the Tx output back to the Rx input, completely ignoring the actual state of the CANRX pin. Transmitted messages can be detected on the CANTX pin.

## 25.5.3 Loopback Silent Mode

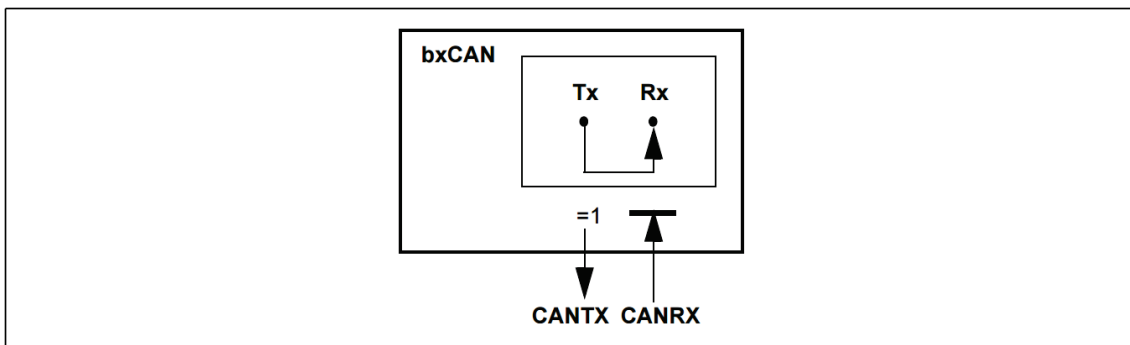


Figure200 bxCAN operating in loopback silent mode

The loopback silent mode can be selected by setting both the LBKM and SILM bits of the CAN\_BTR register to '1'. This mode can be used for "hot self-testing", i.e. the bxCAN can be tested as in the loopback mode without affecting the entire CAN system connected to CANTX and CANRX. In loopback silent mode, the CANRX pin is disconnected from the CAN bus and the CANTX pin is driven to a hidden bit state.

## 25.6 When the W55MH32 is in Debug Mode

When the microcontroller is in debug mode, the Cortex-M3 core is in a suspended state and the bxCAN can continue to operate normally or stop depending on the state of the configuration bits described below:

- DBG\_CAN1\_STOP bit for CAN1 or DBG\_CAN2\_STOP bit for CAN2 in the Debug (DBG) module. See Section31.16.2 : Timer, Watchdog, bxCAN, and Support for I2C Debugging details.
- DBF bit in CAN\_MCR. See Section25.9.2 : CAN Control and Status Registers for details.

## 25.7 bxCAN Functional Description

### 25.7.1 Send Processing

The process of sending a message is as follows: the application program selects a vacant sending mailbox; sets the identifier, the data length and the data to be sent; and then requests the sending of the CAN\_TlXR register with the TXRQ position '1'. after the TXRQ position '1', the mailbox is no longer a vacant mailbox; and once the mailbox is no longer a vacant mailbox, the software no longer has the write permission for the mailbox registers. after the TXRQ position 1 , the mailbox immediately enters the pending state and waits to become the highest priority

---

mailbox, see Sending Priority. Once the mailbox becomes the highest priority mailbox, its state changes to the Scheduled to Send state. As soon as the CAN bus enters the idle state, the message in the scheduled transmission mailbox is sent (enters the transmission state). Once the message in the mailbox has been successfully sent, it immediately becomes a vacant mailbox; the hardware accordingly indicates a successful send by placing RQCP and TXOK in the CAN\_TSR register at position 1.

If the transmission fails, it is '1' to the ALST location of the CAN\_TSR register if caused by arbitration, and '1' to the TERR location if caused by a transmission error.

### **Send Priority**

#### **Determined by Identifier**

When more than 1 sending mailbox is pending, the sending order is determined by the identifier of the message in the mailbox. According to the CAN protocol, the message with the lowest identifier value has the highest priority. If the values of the identifiers are equal, then the message with the smaller mailbox number is sent first.

#### **Determined by the Order in Which Requests are Sent**

The transmit mailbox can be configured to transmit FIFO by setting TXFP position '1' of the CAN\_MCR register. In this mode, the priority of transmission is determined by the transmit request order. This mode is useful for segmented sending.

#### **Discontinue**

The send request can be aborted by setting the ABRQ position '1' to the CAN\_TSR register. The send request is aborted immediately if the mailbox is in the pending or scheduled state. If the mailbox is in the sending state, then aborting the request can lead to 2 results. If the message in the mailbox is successfully sent, then the mailbox becomes a vacant mailbox and the TXOK bit of the CAN\_TSR register is set '1' by hardware. If the message in the mailbox fails to be sent, then the mailbox becomes scheduled and then the send request is aborted, the mailbox becomes vacant and the TXOK bit is cleared '0' by hardware. Therefore, if a mailbox is in the send state, it becomes a vacant mailbox at the end of the send operation.

#### **Disable Automatic Retransmission Mode**

This mode is primarily used to fulfill the need for a time-triggered communication option in the CAN standard. The hardware is made to operate in this mode by applying a NART position '1' to the CAN\_MCR register.

In this mode, the send operation will only be performed once. If the send operation fails, either due to lost arbitration or an error, the hardware will not automatically send the message again. At the end of a transmit operation, the hardware considers the transmit request to be complete, thus applying a '1' to the RQCP location of the CAN\_TSR register, while the result of the transmission is reflected in the TXOK, ALST and TERR bits.

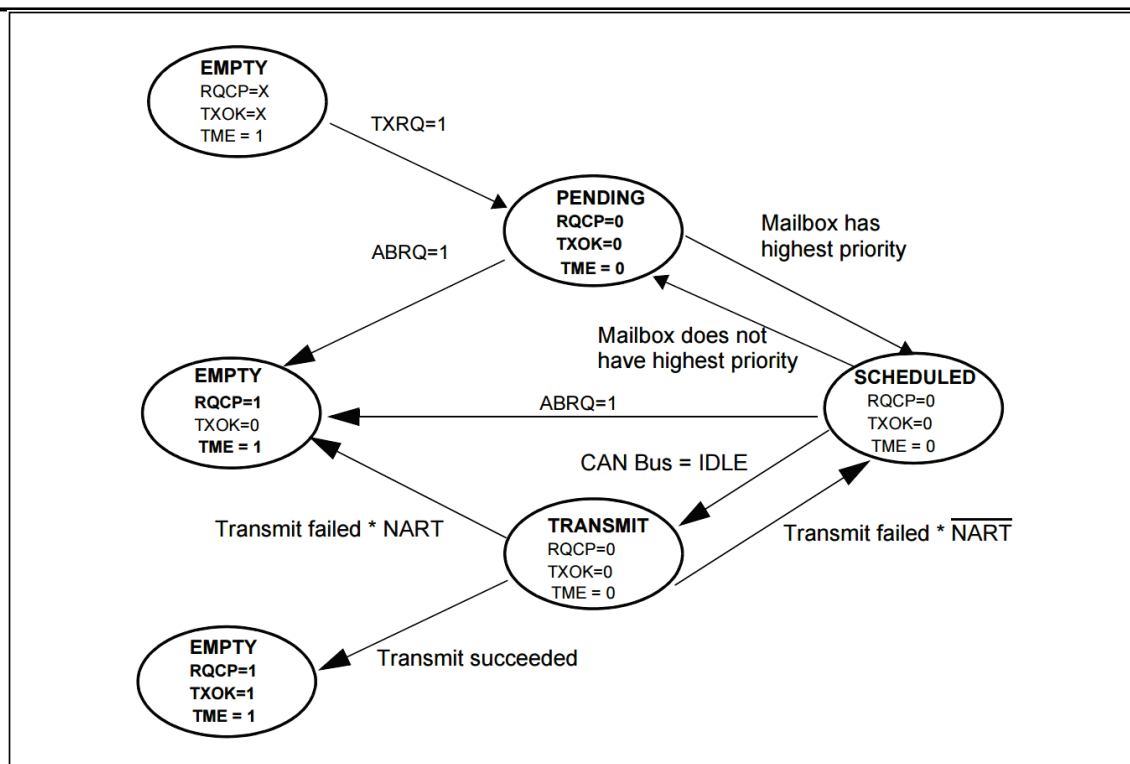


Figure201 Send Mailbox Status

## 25.7.2 Time-triggered Communication Mode

In this mode, the internal timer of the CAN hardware is activated and is used to generate timestamps (of the sending and receiving mailboxes), which are stored in the CAN\_RDTxR/CAN\_TDTxR registers, respectively. The internal timer is accumulated at each CAN bit time (see section 25.7.7). The internal timer is sampled at the sample point position of the received and transmitted frame start bits and generates the timestamp.

## 25.7.3 Reception Management

The received messages are stored in FIFOs with 3 levels of mailbox depth. the FIFOs are managed entirely by hardware, which saves CPU processing load, simplifies the software and guarantees data consistency. The application program can only read the first received message in the FIFO by reading the FIFO output mailbox.

### Valid Message

According to the CAN protocol, a message is considered valid when it is received correctly (no errors until the last bit of the EOF field) and passes the identifier filter. Please refer to section 25.7.4 : Identifier Filtering.



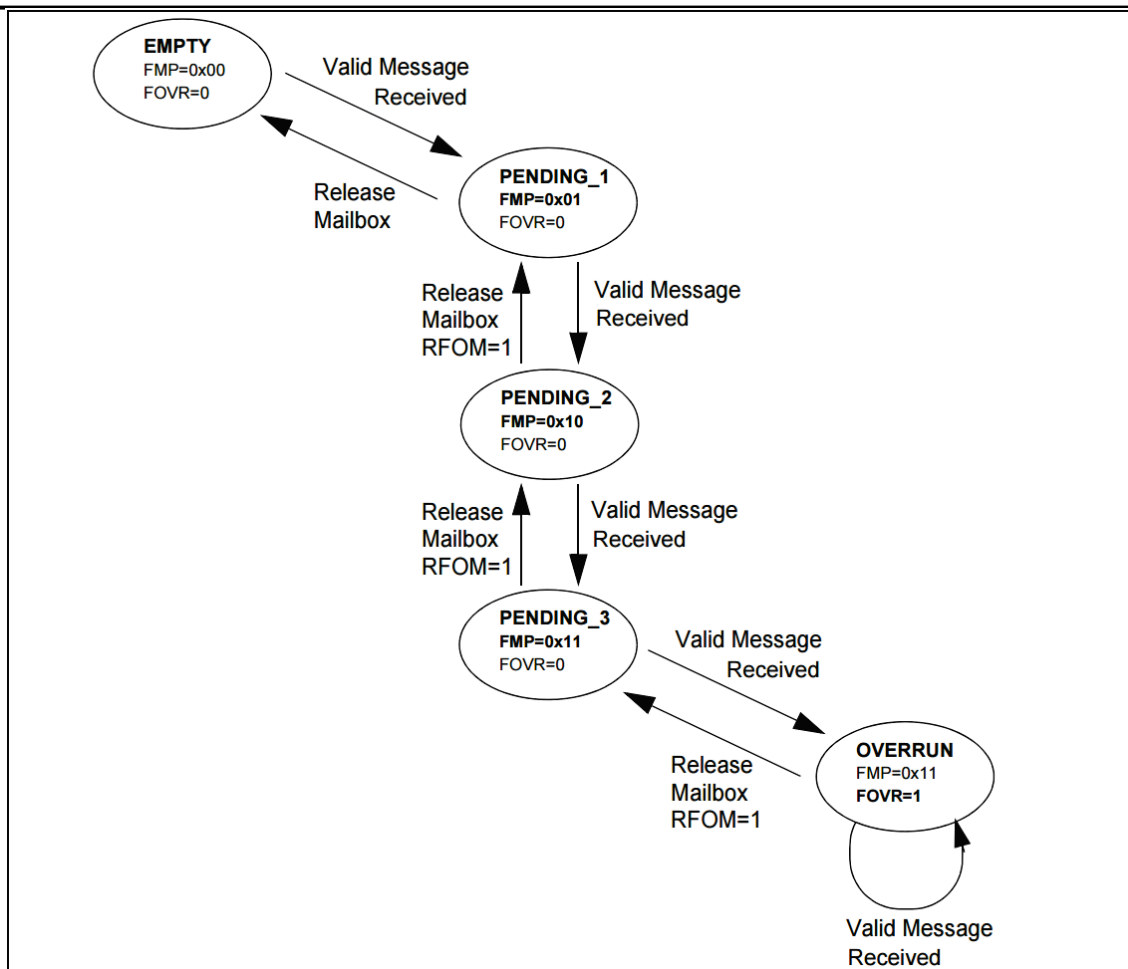


Figure 202 Receive FIFO Status

### FIFO Management

The FIFO starts from an empty state, and after the first valid message is received, the FIFO state changes to pending\_1 (pending\_1), and the hardware sets FMP[1:0] of the CAN\_RFR register to '01' (binary 01b) accordingly. The software can read the FIFO output mailbox to read the message in the mailbox, and then release the mailbox by setting '1' to the RFOM bit of the CAN\_RFR register, so that the FIFO is again becomes empty. If a valid message is received at the same time as the mailbox is released, the FIFO remains in the pending 1 state and the software can read the FIFO output mailbox to read out the newly received message.

If the application does not release the mailbox, after the next valid message is received, the FIFO state changes to pending\_2 (pending\_2) and the hardware sets FMP[1:0] to '10' (binary 10b) accordingly. Repeating the above process, the third valid message changes the FIFO into a for the pending\_3 state (FMP[1:0] = 11b). At this point, software must set 1 to the RFOM bit to free the mailbox so that the FIFO can have room for the next valid message; otherwise, the arrival of the next valid message will result in the loss of a message.

See 25.7.5 section: Message Storage

### Overflows

When the FIFO is in the pending\_3 state (i.e., all 3 mailboxes of the FIFO are full), the next valid message causes an overflow and a message is lost. At this point, the hardware sets a '1' to the FOVR bit of the CAN\_RFR register to indicate the overflow condition. As to which message will be discarded depends on the setting of the FIFO:

- If the FIFO locking feature is disabled (the RFLM bit of the CAN\_MCR register is cleared '0'), the last received message in the FIFO is overwritten by the new message. In this way, the latest received message is not discarded.

- If FIFO locking is enabled (the RFLM bit of the CAN\_MCR register is set to '1'), then newly received messages are discarded and the software can read the 3 earliest received messages in the FIFO.

#### Receive Related Interrupts

Once a message is deposited into the FIFO, the hardware updates the FMP[1:0] bits and generates an interrupt request if the FMPIE bit in the CAN\_IER register is '1'.

When the FIFO becomes full (i.e., the 3rd message is deposited), the FULL bit of the CAN\_RFR register is set to '1' and a full interrupt request is generated if the FFIE bit of the CAN\_IER register is '1'.

In the case of an overflow, the FOVR bit is set to '1' and an overflow interrupt request is generated if the FOVIE bit in the CAN\_IER register is '1'.

## 25.7.4 Identifier Filtering

In CAN protocol, the identifier of the message does not represent the address of the node but is related to the content of the message. Therefore, the sender sends the message to all receivers in the form of a broadcast. When a node receives a message - based on the value of the identifier - it decides whether the message is needed by the software; if so, it is copied into SRAM; if not, the message is discarded without software intervention.

To meet this need, the bxCAN controller provides the application program with a set of 14 variable bit-width, configurable filters (13 to 0) in order to receive only those messages required by the software. The hardware filtering saves CPU overhead that would otherwise have to be filtered by the software thus taking up some CPU overhead. Each filter group x consists of two 32-bit registers, CAN\_FxR0 and CAN\_FxR1.

#### Variable Bit Width

The bit width of each filter group can be independently configured to meet the different needs of an application. Depending on the bit width, each filter group can provide:

- 1 32-bit filter including: STDID[10:0], EXTID[17:0], IDE and RTR bits
- 2 16-bit filters including: STDID[10:0], IDE, RTR and EXTID[17:15] bits are available at Figure 203.

Additionally the filter can be configured for, Masked Bit Mode and Identifier List Mode.

#### Masked Bit Mode

In Masked Bit Mode, the Identifier Register, together with the Mask Register, specifies that any bit of the message identifier should be treated as either a "must match" or a "don't care".

#### Identifier List Mode

In identifier list mode, the mask register is also used as an identifier register. Therefore, instead of using one identifier plus one mask bit, 2 identifier registers are used. Each bit of the receive message identifier must be the same as the filter identifier.

#### Filter Group Bit Width and Mode Settings

Filter groups can be configured via the corresponding CAN\_FMR register. Before configuring a filter group, it must be set to the disabled state by clearing the FACT bit of the CAN\_FAR register. By setting the corresponding FSCx bit of CAN\_FS1R, the to configure the bit width of a filter group, see Figure 203. With the FBMx bit of CAN\_FMR, the identifier list mode or the mask bit mode of the corresponding mask/identifier register can be configured.

In order to filter out a set of identifiers, the filter group should be set to work in mask bit mode. In order to filter out an identifier, the filter group should be set to work in identifier list mode. Filter groups that are not used by the application should remain disabled.

Each filter in a filter group is numbered (called a filter number) from 0 to some maximum value - depending on the mode of the filter group and the setting of the bit width.

For filter configuration, see the following figure.

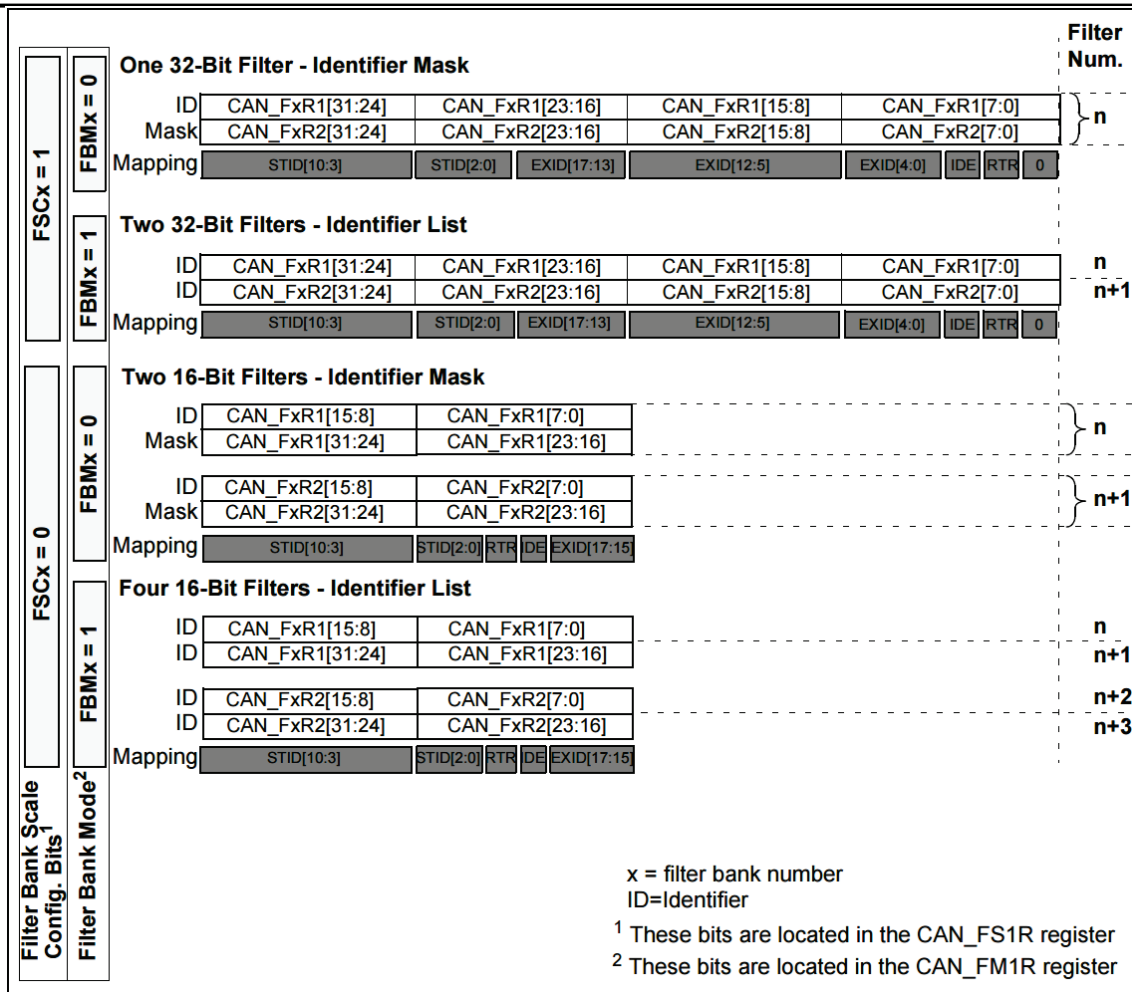


Figure203 Filter Group Bit Width Setting-Register Organization

### Filter Match Serial Number

Once a received message has been stored in the FIFO, it can be accessed by the application program. Typically, the data in the message is copied into SRAM; in order to copy the data to the proper location, the application needs to recognize the different data based on the identifier of the message. bxCAN provides filter match numbers to simplify this recognition process.

According to the filter prioritization rules, the filter match serial number is stored in the mailbox along with the message. Thus each received message has a filter match number associated with it.

Filter match serial numbers can be used in the following two ways:

- Compare the filter match number to a set of desired values.
- Use the filter match number as an index to access the destination address.

For filters in identifier list mode (non-masked filters), the software does not need to compare directly with the identifiers. For filters in masked bit mode, the software only needs to compare those masked bits (bits that must match) that are needed.

When numbering filters, no consideration is given to whether the filter group is active or not. In addition, each FIFO individually numbers its associated filters. Refer to the following figure for an example.

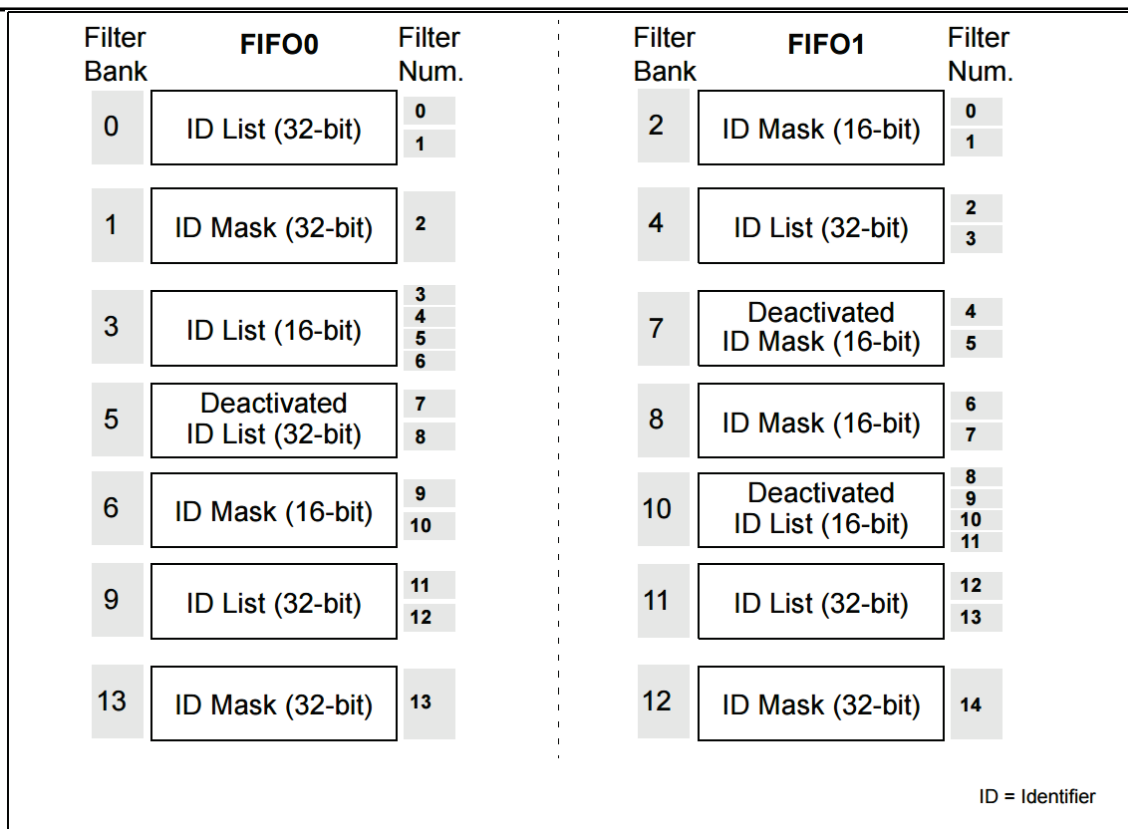


Figure204 Example of filter numbering

### Filter Prioritization Rules

Depending on the different configurations of the filters, it is possible that a message identifier can pass through more than one filter; in this case, the filter match sequence number stored in the receiving mailbox is determined according to the following prioritization rules:

- Filters with a bit width of 32 bits have higher priority than filters with a bit width of 16 bits
- For filters with the same bit width, the identifier list mode has higher priority than the masked bit mode
- Filters with the same bit width and mode, the priority is determined by the filter number, the smaller filter number has a higher priority

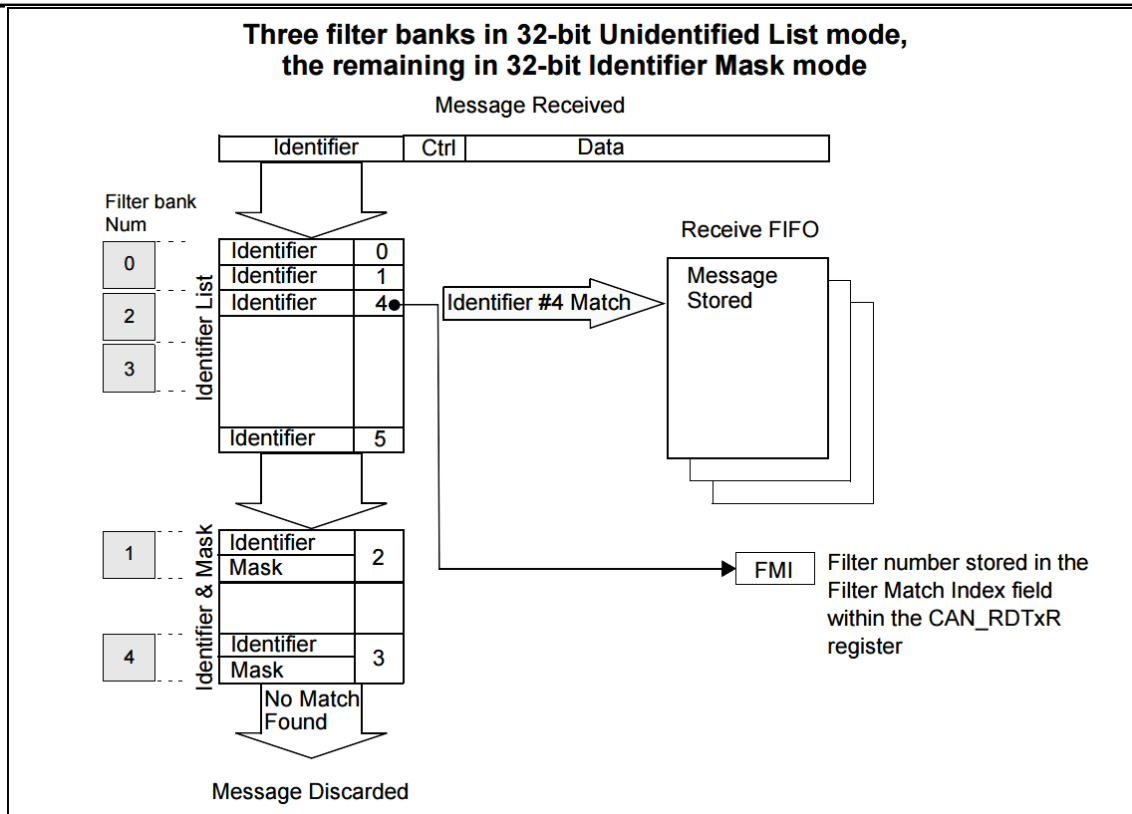


Figure205 Examples of filter mechanisms

The above example illustrates the filter rules of bxCAN: when a message is received, its identifier is first compared with the filters configured in identifier list mode; if a match is made, the message is deposited into the associated FIFO and the serial number of the matched filter is stored in the Filter Match Sequence Number. As shown in the example, the message identifier matches the #4 identifier, so the message contents and FMI4 are deposited into the FIFO. If there is no match, the message identifier is then compared with the filter configured in mask bit mode.

If the message identifier does not match any of the identifiers in the filter, then the hardware discards the message and does not bother the software in any way.

## 25.7.5 Message Storage

The mailbox is the interface between the software and the hardware for passing messages. The mailbox contains all the information related to the message: identifier, data, control, status and timestamp information.

### Send Email

The software needs to set up the various information about the message to be sent (and then send the request to send) in an empty send mailbox. The status of the send can be known by querying the CAN\_TSR register.

Table135 Send Mailbox Register List

Offset relative to the base address of the sending mailbox	register name
0	CAN_TlRxR
4	CAN_TDTxR
8	CAN_TDLxR
12	CAN_TDHxR

### Receiving Mailbox (FIFO)

After receiving a message, the software can access the output mailbox of the receive FIFO to read it. Once the software has processed the message (e.g., read it out), the software should set '1' to the RFOM bit of the CAN\_RfRxR register to release the message to allow storage space for later received messages. The filter match sequence number is stored in the FMI field of the

CAN\_RDTxR register. the 16-bit timestamp is stored in the TIME[15:0] field of the CAN\_RDTxR register.

Table136 Receive Mailbox Register List

Offset relative to the base address of the receiving mailbox	register name
0	CAN_RlRxR
4	CAN_RDTxR
8	CAN_RDLxR
12	CAN_RDHxR

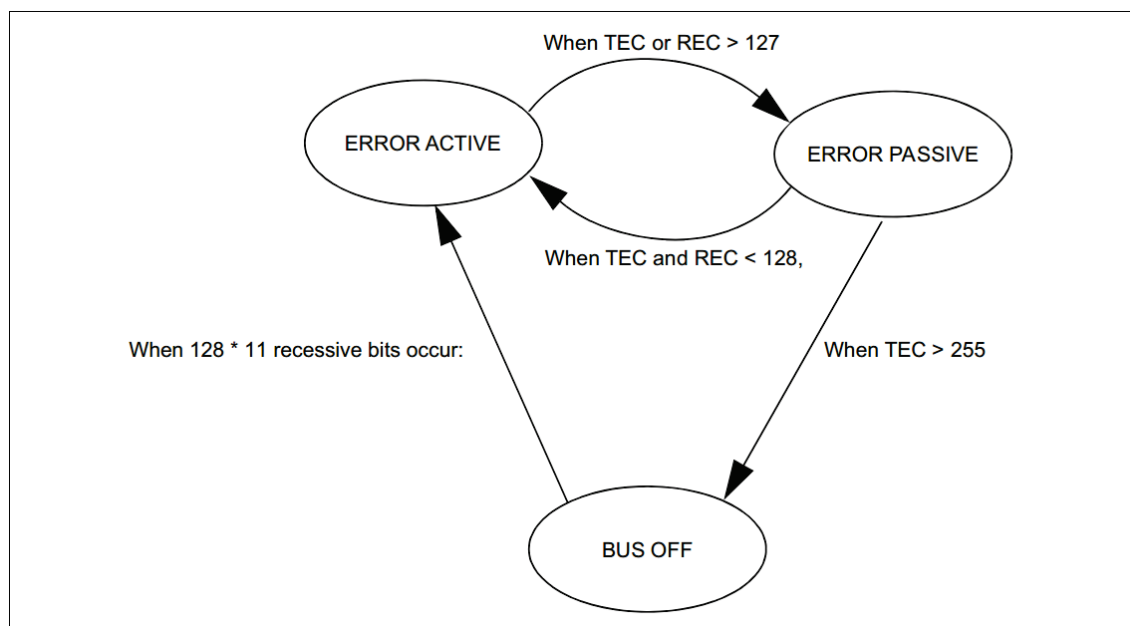


Figure206 CAN Error Status Diagram

## 25.7.6 Error Management

Error management, as described by the CAN protocol, is implemented entirely by hardware through a transmit error counter (TEC field in the CAN\_ESR register), and a receive error counter (REC field in the CAN\_ESR register), whose value is increased or decreased depending on the error. For more information on TEC and REC management, please refer to the CAN standard.

The software can read out their values to determine the stability of the CAN network.

In addition, the CAN\_ESR register provides detailed information about the current error status. By setting the CAN\_IER register (e.g., the ERRIE bit), software has the flexibility to control interrupt generation when an error is detected.

### Offline Recovery

When TEC is greater than 255, the bxCAN enters the offline state and the BOFF bit of the CAN\_ESR register is set to '1'. In the offline state, bxCAN cannot receive and send messages. Depending on the setting of the ABOM bit in the CAN\_MCR register, the bxCAN can be recovered from the offline state (changed to an error-active state) either automatically or at the request of software. In both cases, the bxCAN must wait for a recovery process as described by the CAN standard (128 times 11 consecutive implicit bits are detected on the CANRX pin).

If the ABOM bit is '1', the recovery process is automatically started once the bxCAN goes offline. If the ABOM bit is '0', the software must request bxCAN to enter and then exit the initialization mode before the recovery process is turned on.

*Notes: In initialization mode, bxCAN does not monitor the status of the CANRX pin, which prevents the recovery process from being completed. In order to complete the recovery process, the bxCAN must operate in normal mode.*

## 25.7.7 Bit-Time Characterization

The Bit-Time Characterization Logic monitors the serial CAN bus by sampling and adjusting its sampling point by synchronizing with the edge of the start of the frame and re-synchronizing with the following edge.

Its operation can be simply explained by dividing the notional per-bit time into 3 segments as described below:

- Synchronization Segment (SYNC\_SEG): It is normally expected that bit changes occur within this time period. Its value is fixed to 1 time unit (1xtCAN).
- Time segment 1 (BS1): defines the position of the sampling point. It contains PROP\_SEG and PHASE\_SEG1 from the CAN standard, and its value can be programmed from 1 to 16 time units, but it can also be automatically extended to compensate for the positive phase drift due to the difference in frequency of the different nodes in the network.
- Time segment 2 (BS2): defines the position of the transmit point. It represents PHASE\_SEG2 in the CAN standard. its value can be programmed from 1 to 8 time units, but it can also be automatically shortened to compensate for negative phase drift.

The resynchronization jump width (SJW) defines the upper limit of how many time units can be lengthened or shortened in each bit. Its value can be programmed from 1 to 4 time units.

An effective hop is defined as the 1st transition from a dominant bit to a recessive bit when bxCAN does not send the recessive bit itself.

If a valid jump is detected in time segment 1 (BS1) instead of in the synchronization segment (SYNC\_SEG), then the time of BS1 is extended up to as long as SJW, and thus the sampling point is delayed.

Conversely if a valid jump is detected at time period 2 (BS2) instead of at SYNC\_SEG, then the time at BS2 is shortened by up to as long as the SJW and thus the sampling point is advanced.

To avoid programming errors in the software, the setting of the bit time characteristic register (CAN\_BTR) can only be done when the bxCAN is in the initialized state.

*Notes: For detailed information on CAN bit timing characteristics and resynchronization mechanisms, refer to the ISO 11898 standard.*

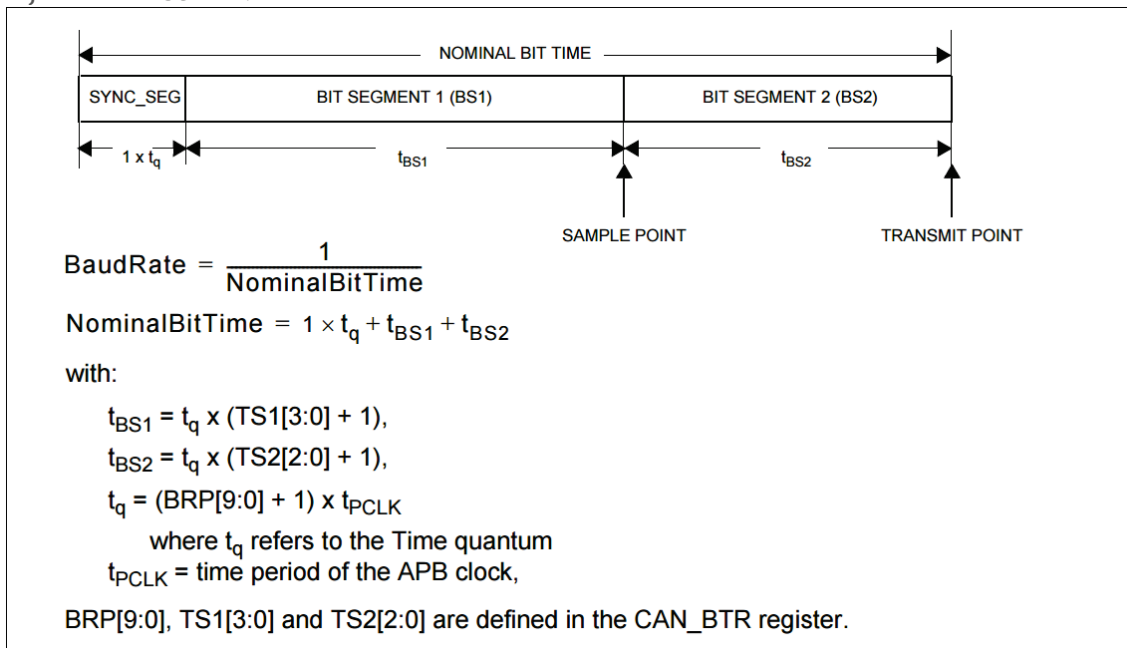


Figure207 Bit Timing

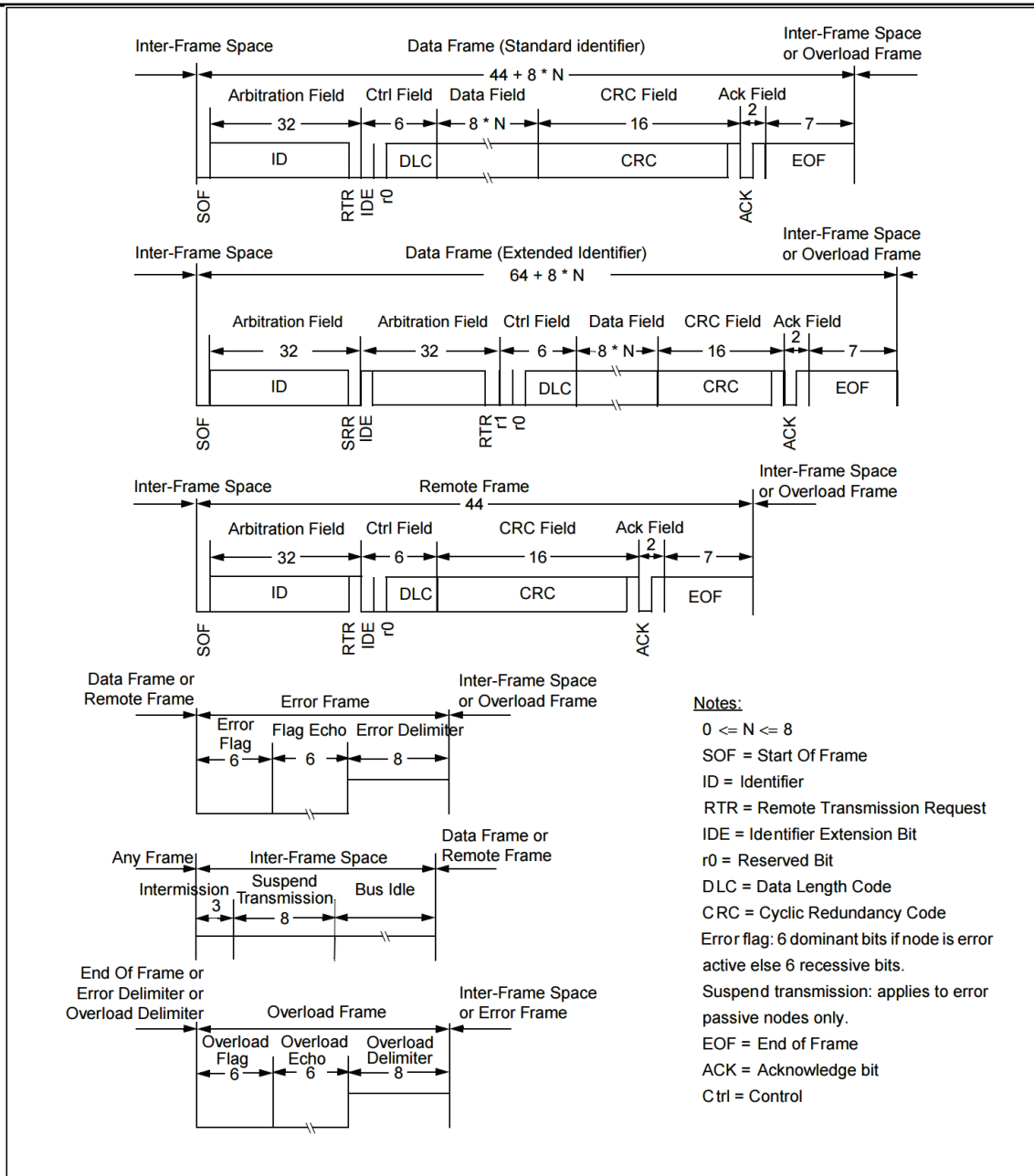


Figure208 Various CAN frames



## 25.8 bxCAN Interrupt

The bxCAN occupies 4 dedicated interrupt vectors. Each interrupt source can be individually allowed and disabled by setting the CAN interrupt allow register (CAN\_IER).

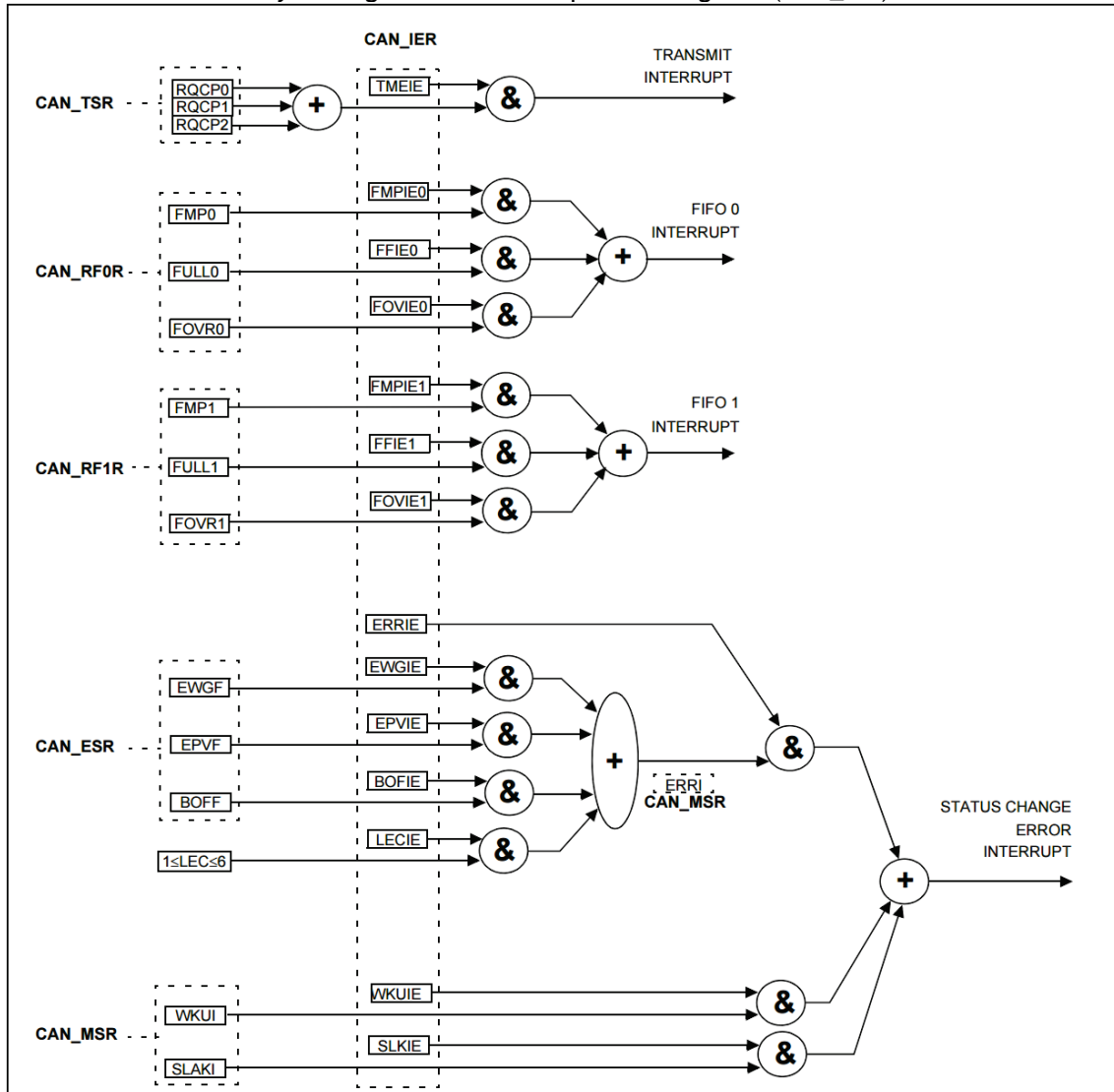


Figure 209 Event Flags and Interrupt Generation

- A transmit interrupt can be generated by the following events:
  - Transmit mailbox 0 becomes empty and the RQCP0 bit of the CAN\_TSR register is set to '1'.
  - Transmit mailbox 1 becomes empty and the RQCP1 bit of the CAN\_TSR register is set to '1'.
  - Transmit mailbox 2 becomes empty and the RQCP2 bit of the CAN\_TSR register is set to '1'.
- The FIFO0 interrupt can be generated by the following events:
  - FIFO0 receives a new message and the FMP0 bit of the CAN\_RF0R register is no longer '00'.
  - In the case that FIFO0 becomes full, the FULL0 bit of the CAN\_RF0R register is set to '1'.
  - In the case of an overflow of FIFO0, the FOVR0 bit of the CAN\_RF0R register is set to '1'.
- The FIFO1 interrupt can be generated by the following events:
  - FIFO1 receives a new message and the FMP1 bit of the CAN\_RF1R register is no longer '00'.
  - In the case that FIFO1 becomes full, the FULL1 bit of the CAN\_RF1R register is set to '1'.
  - In the case of an overflow of FIFO1, the FOVR1 bit of the CAN\_RF1R register is set to '1'.
- Error and state change interrupts can be generated by the following events:
  - error conditions, refer to the CAN Error Status Register (CAN\_ESR) for more information on error conditions.

- Wake-up condition to monitor to the Start of Frame (SOF) bit on the CAN receive pin.
- CAN enters sleep mode.

## 25.9 CAN Register Description

Abbreviations used in register descriptions can be found in Section1 . These peripheral registers must be operated in word (32-bit) format.

### 25.9.1 Register Access Protection

Incorrect access to certain registers can lead to temporary interference of one CAN node with the entire CAN network. Therefore, software can only modify the CAN\_BTR register when the CAN is in initialization mode.

While the sending of error data does not pose a problem for the network layer of the CAN network, it can have a serious impact on the application program. Therefore, the software can only change it in a state where the sending mailbox is empty, seeFigure201 .

The values of the filters can only be modified in the state where the corresponding filter group is turned off, or after setting the FINIT bit to '1'. In addition, the filter settings can only be modified if the entire filter is set to initialization mode (i.e. FINIT=1), i.e. the CAN\_FmXR, CAN\_FSxR and CAN\_FFAR registers can be modified.

### 25.9.2 CAN Control and Status Registers

For abbreviations in register descriptions, see section1 .

#### CAN Master Control Register (CAN\_MCR)

Address Offset:0x00

Reset value:0x0001 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															DBF
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESET	Reserved							TTCM	ABOM	AWUM	NART	RFLM	TXFP	SLEEP	INRQ
rs								rw	rw	rw	rw	rw	rw	rw	rw

Bit	notation	clarification
31:17	Reserved	Reserved, always reads 0.
16	DBF	<b>DBF:</b> Debug freeze 0: CAN works as usual during commissioning 1: Freeze CAN receive/transmit while debugging. Can still read/write and control the receive FIFO normally.
15	RESET	<b>RESET:</b> bxCAN software master reset 0: This peripheral works normally; 1: A forced reset of the bxCAN is performed, after which the bxCAN enters sleep mode (the FMP bit and the CAN_MCR register are initialized to their Reset values). The hardware automatically clears '0' to this bit thereafter.
14:8	Reserved	Reserved, hardware forced to 0.
7	TTCM	<b>TTCM:</b> Time triggered communication mode (Time triggered communication mode) 0: Disable time-triggered communication mode; 1: Allow time-triggered communication mode. Note:To learn more about time-triggered communication modes, refer to 22.7.2: Time-Triggered Communication Modes.
6	ABOM	<b>ABOM:</b> Automatic bus-off management (Automatic bus-off management) This bit determines under what conditions the CAN hardware can exit the offline state. 0: The exit process of the offline state is that after the software sets '1' and subsequently clears '0' to the INRQ bit of the CAN_MCR register, the offline state is exited once the hardware detects 128 consecutive 11-bit recessive bits; 1: Once the hardware detects 128 times 11 consecutive hidden bits, it automatically exits the offline state. Note:For more information on offline status, see 22.7.6: Error Management.
5	AWUM	<b>AWUM:</b> Automatic wakeup mode (Automatic wakeup mode) This bit determines whether the CAN will be woken up by hardware or software when it is in sleep mode. 0: Sleep mode is awakened by software by clearing the SLEEP bit in the CAN_MCR register; 1: Sleep mode is automatically woken up by hardware by detecting CAN

		messages. While waking up, the hardware automatically clears '0' to the SLEEP and SLAK bits of the CAN_MSR register.
4	NART	<b>NART:</b> No automatic retransmission. 0: According to the CAN standard, the CAN hardware will keep retransmitting the message automatically when it fails to send it until it is sent successfully; 1: CAN messages are sent only 1 time, regardless of the result of the sending (success, error or arbitration loss).
3	RFLM	<b>RFLM:</b> Receive FIFO locked mode (Receive FIFO locked mode) 0: The FIFO is not locked at the time of receive overflow, when the message in the receive FIFO is not read out, the next received message will overwrite the original message; 1: The FIFO is locked during receive overflow, and when a message in the receive FIFO is not read, the next received message is discarded.
2	TXFP	<b>TXFP:</b> Transmit FIFO priority When there are multiple messages waiting to be sent at the same time, this bit determines the order in which these messages are sent 0: The priority is determined by the identifier of the message; 1: The priority is determined by the order in which requests are sent.
1	SLEEP	<b>SLEEP:</b> Sleep mode request Software for this position '1' can request the CAN to go into sleep mode, and once the current CAN activity (sending or receiving messages) is over, the CAN goes to sleep. Software clears '0' to this bit to enable the CAN to exit sleep mode. When the AWUM bit is set and the SOF bit is detected in the CANRx signal, the hardware clears '0' to this bit. This bit is set '1' after a reset, i.e. the CAN is in sleep mode after a reset.
0	INRQ	<b>INRQ:</b> Initialization request Software clearing this bit to '0' enables the CAN to move from initialization mode to normal operation mode: when the CAN detects 11 consecutive implicit bits on the receive pin, the CAN is synchronized and ready to receive and send data. For this purpose, the hardware clears '0' to the INAK bit of the CAN_MSR register accordingly. A software response to this bit 1 enables the CAN to enter initialization mode from normal operating mode: once the current CAN activity (sending or receiving) is over, the CAN enters initialization mode. Correspondingly, hardware pairs the INAK position '1' of the CAN_MSR register.

### CAN Master Status Register (CAN\_MSR)

Address offset:0x04

Reset value:0x0000 0C02

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				RX	SAMP	RXM	TXM	Reserved			SLAKI	WKUI	ERRI	SLAK	INAK
				r	r	r	r				rc_w1	rc_w1	rc_w1	r	r

Bit	notation	clarification
31:12	Reserved	Reserved, always reads 0.
11	RX	<b>RX:</b> CAN receive level (CANRx signal) This bit reflects the actual level of the CAN receive pin (CAN_RX).
10	SAMP	<b>SAMP:</b> the last sample point. The last sampled value of the CAN receive pin (corresponding to the value of the current receive bit).
9	RXM	<b>RXM:</b> Receive mode A '1' in this bit indicates that the CAN is currently a receiver.
8	TXM	<b>TXM:</b> Transmit mode A '1' in this bit indicates that the CAN is currently a transmitter.
7:5	Reserved	Reserved, always reads 0.
4	SLAKI	<b>SLAKI:</b> Sleep acknowledge interrupt When SLKIE=1, once the CAN enters sleep mode hardware '1's to this bit and the corresponding interrupt is triggered immediately afterwards. When this bit is set to '1', a change of state interrupt is generated if the SLKIE bit in the CAN_IER register is set. Software can clear '0' to this bit, and hardware also clears '0' to the SLAK bit when it is cleared '0'. Note:When SLKIE=0, this bit should not be queried, but the SLAK bit should be queried to know the sleep state.
3	WKUI	<b>WKUI:</b> Wakeup interrupt (Wakeup interrupt) When the CAN is in sleep state, the hardware sets the Start of Frame (SOF) bit to '1' as soon as it is detected; and generates a state change interrupt if the WKUIE bit in the CAN_IER register is '1'.

		This bit is cleared '0' by software.
2	ERRI	<b>ERRI:</b> Error interrupt When an error is detected, a bit in the CAN_ESR register is set '1' and hardware '1' to that bit if the corresponding interrupt enable bit in the CAN_IER register is also set '1'; if the ERRIE bit in the CAN_IER register is '1', a state change interrupt is generated. This bit is cleared '0' by software.
1	SLAK	<b>SLAK:</b> Sleep mode confirmation This bit is set '1' by hardware to indicate that the software CAN module is in sleep mode. This bit is an acknowledgement of the software request to enter sleep mode (to the SLEEP position '1' of the CAN_MCR register). Hardware clears '0' to this bit when the CAN exits sleep mode (synchronization with the CAN bus is required). Synchronization with the CAN bus means that the hardware needs to detect 11 consecutive implicit bits on the RX pin of the CAN. Note: Clearing '0' to the SLEEP bit of the CAN_MCR, either through software or hardware, will initiate the process of exiting sleep mode. For more information on clearing the SLEEP bit, see the description of the AWUM bit of the CAN_MCR register.
0	INAK	<b>INAK:</b> Initialization Acknowledgement This bit is set '1' by hardware to indicate that the software CAN module is in initialization mode. This bit is an acknowledgement of the software request to enter the initialization mode (INRQ location '1' to the CAN_MCR register). The hardware clears '0' to this bit when the CAN exits initialization mode (synchronization with the CAN bus is required). Synchronization with the CAN bus means that the hardware needs to detect 11 consecutive implicit bits on the RX pin of the CAN.

### CAN Transmit Status Register (CAN\_TSR)

Address offset: 0x08

Reset value: 0x1C00 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOW2	LOW1	LOW0	TME2	TME1	TME0	CODE[1:0]	ABRQ2		Reserved			TERR2	ALST2	TXOK2	RQCP2
r	r	r	r	r	r	r	rs					rc_w1	rc_w1	rc_w1	rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABRQ1		Reserved		TERR1	ALST1	TXOK1	RQCP1	ABRQ0		Reserved		TERR0	ALST0	TXOK0	RQCP0
rs				rc_w1	rc_w1	rc_w1	rc_w1	rs				rc_w1	rc_w1	rc_w1	rc_w1

Bit	notation	clarification
31	LOW2	<b>LOW2:</b> Lowest priority flag for mailbox2 When multiple mailboxes are waiting for a message to be sent and mailbox 2 has the lowest priority, the hardware assigns a '1' to that position.
30	LOW1	<b>LOW1:</b> Lowest priority flag for mailbox1 When more than one mailbox is waiting to send a message and mailbox 1 has the lowest priority, the hardware assigns a '1' to that position.
29	LOW0	<b>LOW0:</b> Lowest priority flag for mailbox0 When more than one mailbox is waiting to send a message and mailbox 0 has the lowest priority, the hardware assigns a '1' for that position. Note: If only 1 mailbox is waiting, LOW[2:0] is cleared '0'.
28	TME2	<b>TME2:</b> Transmit mailbox2 empty When there are no messages waiting to be sent in mailbox 2, the hardware assigns a '1' to this position.
27	TME1	<b>TME1:</b> Transmit mailbox1 empty (Transmit mailbox1 empty) When there are no messages waiting to be sent in mailbox 1, the hardware assigns a '1' to that position.
26	TME0	<b>TME0:</b> Send mailbox0 empty (Transmit mailbox0 empty) When there are no messages waiting to be sent in mailbox 0, the hardware assigns a '1' to this position.
25:24	CODE[1:0]	<b>CODE[1:0]:</b> Mailbox code When at least 1 sending mailbox is empty, these 2 bits indicate the next empty sending mailbox number. When all sending mailboxes are empty, these 2 bits indicate the sending mailbox number with the lowest priority.
23	ABRQ2	<b>ABRQ2:</b> Abort request for mailbox2 A software '1' to this position aborts the send request for mailbox 2, and the hardware clears '0' to this bit when the send message for mailbox 2 is cleared. If there are no messages waiting to be sent in mailbox 2, a '1' to this position has no effect.
22:20	Reserved	Reserved bit, hardware forces its value to 0
19	TERR2	<b>TERR2:</b> Transmission failure of mailbox2 (Transmission error of mailbox2) When the transmission of mailbox2 fails because of an error, '1' to this position.
18	ALST2	<b>ALST2:</b> Arbitration lost for mailbox2 (Arbitration lost for mailbox2) When mailbox2

		fails to send due to arbitration loss, '1' to this position.
17	TXOK2	<b>TXOK2:</b> Transmission success of mailbox2 (Transmission OK of mailbox2) The hardware updates this bit every time after mailbox2 makes a transmission attempt: 0: The last send attempt failed; 1: Last send attempt was successful. When the send request for mailbox 2 has been successfully completed, the hardware is '1' for that position. SeeFigure201 .
16	RQCP2	<b>RQCP2:</b> Request completed mailbox2 When the last request (send or abort) to mailbox 2 is completed, the hardware is '1' for that location. Software writing a '1' to this bit can clear a '0' to it; the bit is also cleared '0' when the hardware receives a transmit request (the TXRQ bit of the CAN_TI2R register is set to '1'). When this bit is cleared '0', the other transmit status bits of Mailbox 2 (TXOK2,ALST2 and TERR2) are also cleared '0'.
15	ABRQ1	<b>ABRQ1:</b> Abort request for mailbox1 A software '1' to this position aborts the send request for mailbox 1, and the hardware clears '0' to this bit when the send message for mailbox 1 is cleared. If there are no messages waiting to be sent in Mailbox 1, a '1' to this position has no effect.
14:12	Reserved	Reserved bit, hardware forces its value to 0
11	TERR1	<b>TERR1:</b> Transmission failure of mailbox1 (Transmission error of mailbox1) When the transmission of mailbox1 fails because of an error, '1' to this position.
10	ALST1	<b>ALST1:</b> Arbitration lost for mailbox1 (Arbitration lost for mailbox1) When mailbox1 fails to send due to arbitration loss, '1' for this position.
9	TXOK1	<b>TXOK1:</b> Transmission success of mailbox1 (Transmission OK of mailbox1) The hardware updates this bit every time after mailbox1 makes a transmission attempt: 0: The last send attempt failed; 1: Last send attempt was successful. When the send request for mailbox 1 has been successfully completed, the hardware '1' to that location. SeeFigure201 .
8	RQCP1	<b>RQCP1:</b> Request completed mailbox1 When the last request (send or abort) to mailbox 1 is completed, the hardware is '1' for that location. Software writing a '1' to this bit can clear a '0' to it; the bit is also cleared '0' when the hardware receives a transmit request (the TXRQ bit of the CAN_TI1R register is set to '1'). When this bit is cleared '0', the other transmit status bits of Mailbox 1 (TXOK1, ALST1 and TERR1) are also cleared '0'.
7	ABRQ0	<b>ABRQ0:</b> Abort request for mailbox0 (Abort request for mailbox0) Software can abort a send request from mailbox 0 for this position '1', and hardware clears '0' to this bit when a send message from mailbox 0 is cleared. If there are no messages waiting to be sent in mailbox 0, there is no effect on this position 1.
6:4	Reserved	Reserved bit, hardware forces its value to 0
3	TERR0	<b>TERR0:</b> Transmission failure of mailbox0 (Transmission error of mailbox0) When the transmission of mailbox0 fails because of an error, '1' to this position.
2	ALST0	<b>ALST0:</b> Arbitration lost for mailbox0 (Arbitration lost for mailbox0) When mailbox0 fails to send due to arbitration loss, '1' to this position.
1	TXOK0	<b>TXOK0:</b> Transmission success of mailbox0 (Transmission OK of mailbox0) The hardware updates this bit every time after a transmission attempt is made by mailbox0: 0: The last send attempt failed; 1: Last send attempt was successful. When the send request for mailbox 0 has been successfully completed, the hardware '1' to that position. SeeFigure201 .
0	RQCP1	<b>RQCP1:</b> Request completed mailbox0 When the last request (send or abort) to mailbox 0 is completed, the hardware '1' to that location. Software writing a '1' to this bit can clear a '0' to it; the bit is also cleared '0' when the hardware receives a transmit request (the TXRQ bit of the CAN_TI0R register is set to '1'). When this bit is cleared '0', the other transmit status bits of Mailbox 0 (TXOK0, ALST0 and TERR0) are also cleared '0'.

### CAN receive FIFO0 register (CAN\_RF0R)

Address Offset:0x0C

Reset value:0x00

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										RFOM0	FOVR0	FULL0	Reserved	FMP0[1:0]	
										rs	rc_w1	rc_w1		r	r

Bit	notation	clarification
31:6	Reserved	Reserved bit, hardware forced to 0
5	RFOM0	<b>RFOM0:</b> Release receive FIFO0 output mailbox (Release FIFO0 output mailbox) The software releases the output mailbox of the receive FIFO by applying a '1' to this location. If the receive FIFO is empty, then a '1' to this location has no effect, i.e. a '1' to this location only makes sense if there are messages in the FIFO. If there are more than 2 messages in the FIFO, since the FIFO is characterized by the fact that the software needs to release the output mailbox to access the 2nd message. When the output mailbox is released, hardware clears '0' to this bit.
4	FOVR0	<b>FOVR0:</b> FIFO0 overrun When FIFO0 is full and a new message is received and the message meets the filtering criteria, hardware '1' to this bit. This bit is cleared '0' by software.
3	FULL0	<b>FULL0:</b> FIFO0 full (FIFO0 full) When there are 3 messages in FIFO0, hardware '1' to this bit. This bit is cleared '0' by software.
2	Reserved	Reserved bit, hardware forces its value to 0
1:0	FMP0[1:0]	<b>FMP0[1:0]:</b> FIFO0 message number (FIFO0 messagepending) FIFO0 Message Number These 2 bits reflect the number of messages currently stored in receive FIFO0. The hardware adds 1 to FMP0 every time 1 new message is deposited into receive FIFO0. Whenever software writes a '1' to the RFOM0 bit to release the output mailbox, FMP0 is decremented by one until it is zero.

### CAN receive FIFO1 register (CAN\_RF1R)

Address offset:0x10

Reset value:0x00

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										RFOM1	FOVR1	FULL1	Reserved	FMP1[1:0]	
										rs	rc_w1	rc_w1		r	r

Bit	notation	clarification
31:6	Reserved	Reserved bit, hardware forced to 0
5	RFOM1	<b>RFOM1:</b> Release receive FIFO1 output mailbox (Release FIFO1 output mailbox) The software releases the output mailbox of the receive FIFO by applying a '1' to this location. If the receive FIFO is empty, then a '1' to this location has no effect, i.e. a '1' to this location only makes sense if there are messages in the FIFO. If there are more than 2 messages in the FIFO, since the FIFO is characterized by the fact that the software needs to release the output mailbox to access the 2nd message. When the output mailbox is released, hardware clears '0' to this bit.
4	FOVR1	<b>FOVR1:</b> FIFO1 overrun When FIFO1 is full and a new message is received and the message meets the filtering criteria, hardware '1' to this bit. This bit is cleared '0' by software.
3	FULL1	<b>FULL1:</b> FIFO1 full (FIFO1 full) Hardware '1' to this bit when there are 3 messages in FIFO1. This bit is cleared '0' by software.
2	Reserved	Reserved bit, hardware forces its value to 0
1:0	FMP1[1:0]	<b>FMP1[1:0]:</b> FIFO1 message number (FIFO1 messagepending) FIFO1 Message Number These 2 bits reflect the number of messages currently stored in receive FIFO1. The hardware adds 1 to FMP1 every time 1 new message is deposited into receive FIFO1. Whenever software writes a 1 to the RFOM1 bit to release the output mailbox,

FMP1 is decremented by 1 until it is 0.

### CAN Interrupt Enable Register (CAN\_IER)

Address Offset:0x14

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved														SLKIE	WKUIE
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERRIE	Reserved			LECIE	BOFIE	EPVIE	EWGIE	Reserved	FOVIE1	FFIE1	FMPIE1	FOVIE0	FFIE0	FMPIE0	TMEIE
rw				rw	rw	rw	rw	res	rw	rw	rw	rw	rw	rw	rw

Bit	notation	clarification
31:18	Reserved	Reserved bit, hardware forced to 0
17	SLKIE	<b>SLKIE:</b> Sleep interrupt enable 0: When the SLAKI bit is set to '1', no interrupt is generated; 1: When the SLAKI bit is set to '1', an interrupt is generated.
16	WKUIE	<b>WKUIE:</b> Wakeup interrupt enable 0: When the WKUI bit is set to '1', no interrupt is generated; 1: An interrupt is generated when the WKUI bit is set to '1'.
15	ERRIE	<b>ERRIE:</b> Error interrupt enable 0: No interrupt is generated when there is an error pending in the CAN_ESR register; 1: Generate an interrupt when there is an error pending in the CAN_ESR register.
14:12	Reserved	Reserved bit, hardware forced to 0.
11	LECIE	<b>LECIE:</b> Last error code interrupt enable 0: The ERRI bit is not set when an error is detected and the hardware sets LEC[2:0]; 1: Set the ERRI bit to '1' when an error is detected and the hardware sets LEC[2:0].
10	BOFIE	<b>BOFIE:</b> offline interrupt enable (Bus-off interrupt enable) 0: When the BOFF bit is set to '1', the ERRI bit is not set; 1: Set the ERRI bit to '1' when the BOFF bit is set to '1'.
9	EPVIE	<b>EPVIE:</b> Error Passive Interrupt Enable (Error Passive Interrupt Enable) 0: When the EPVF bit is set to '1', the ERRI bit is not set; 1: Set the ERRI bit to '1' when the EPVF bit is set to '1'.
8	EWGIE	<b>EWGIE:</b> Error warning interrupt enable 0: When the EWGF bit is set to '1', the ERRI bit is not set; 1: Set the ERRI bit to '1' when the EWGF bit is set to '1'.
7	Reserved	Reserved bit, hardware forced to 0
6	FOVIE1	<b>FOVIE1:</b> FIFO1 overrun interrupt enable 0: When the FOVR bit of FIFO1 is set to '1', no interrupt is generated; 1: An interrupt is generated when the FOVR bit of FIFO1 is set to '1'.
5	FFIE1	<b>FFIE1:</b> FIFO1 full interrupt enable 0: When the FULL bit of FIFO1 is set to '1', no interrupt is generated; 1: An interrupt is generated when the FULL bit of FIFO1 is set to '1'.
4	FMPIE1	<b>FMPIE1:</b> FIFO1 messagepending interrupt enable (FIFO messagepending interrupt enable) 0: When the FMP[1:0] bits of FIFO1 are non-zero, no interrupt is generated; 1: An interrupt is generated when the FMP[1:0] bits of FIFO1 are non-zero.
3	FOVIE0	<b>FOVIE0:</b> FIFO0 overrun interrupt enable 0: When the FOVR bit of FIFO0 is set to '1', no interrupt is generated; 1: An interrupt is generated when the FOVR bit of FIFO0 is set to '1'.
2	FFIE0	<b>FFIE0:</b> FIFO0 full interrupt enable 0: When the FULL bit of FIFO0 is set to '1', no interrupt is generated; 1: An interrupt is generated when the FULL bit of FIFO0 is set to '1'.
1	FMPIE0	<b>FMPIE0:</b> FIFO0 messagepending interrupt enable (FIFO messagepending interrupt enable) 0: When the FMP[1:0] bits of FIFO0 are non-zero, no interrupt is generated; 1: An interrupt is generated when the FMP[1:0] bits of FIFO0 are non-zero.
0	TMEIE	<b>TMEIE:</b> Transmit mailbox empty interrupt enable 0: When the RQCPx bit is set to '1', no interrupt is generated; 1: An interrupt is generated when the RQCPx bit is set to '1'. Note:Refer to25.8 section bxCAN Interrupts.



## CAN Error Status Register (CAN\_ESR)

Address offset:0x18

Reset value:0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REC[7:0]								TEC[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								LEC[2:0]		Reserv ed	BOFF	EPVF	WEGF		
								rw	rw	rw	res	r	r	r	r

Bit	notation	clarification
31:24	REC[7:0]	<b>REC[7:0]:</b> Receive error counter This counter is implemented in accordance with the reception part of the fault definition mechanism of the CAN protocol. According to the CAN standard, when a reception error occurs, this counter is either increased by 1 or increased by 8, depending on the conditions of the error; and after each successful reception, this counter is decreased by 1, or when the value of this counter is greater than 127, it is set to a value of 120. when the value of this counter is greater than 127, CAN enters an error-passive state.
23:16	TEC[7:0]	<b>TEC[7:0]:</b> Low 8 bits of the 9-bit transmit error counter (Least significant byte of the 9-bit transmit error counter) Similar to the above, this counter is implemented according to the sending part of the fault definition mechanism of the CAN protocol.
15:17	Reserved	Reserved bit, hardware forced to 0.
6:4	LEC[2:0]	<b>LEC[2:0]:</b> Last error code When an error is detected on the CAN bus, the hardware sets it according to the error. The hardware clears its value to '0' when the message has been correctly sent or received. The hardware does not use error code 7. the software can set this value so that updates to the code can be detected.000: No error; 001:Bit Filling Error; 010:Format (Form) error; 011:Acknowledgement (ACK) error; 100:recessive bit error; 101:dominant bit error; 110:CRC error; 111:Set by software.
3	Reserved	Reserved bit, hardware forced to 0.
2	BOFF	<b>BOFF:</b> Bus-off flag When entering the offline state, the hardware '1' to this position. When the transmit error counter TEC overflows, i.e. is greater than 255, the CAN enters the offline state. Please refer to 22.7.6.
1	EPVF	<b>EPVF:</b> Error passive flag When the number of errors reaches the threshold for error passive, the hardware '1' for that position. (Receive Error Counter or Transmit Error Counter value >127).
0	EWGF	<b>EWGF:</b> Error warning flag When the number of errors reaches the warning threshold, the hardware '1' for the position. (Receive Error Counter or Transmit Error Counter value ≥ 96).

## CAN Bit Timing Register (CAN\_BTR)

Address Offset:0x1C

Reset value:0x0123 0000

Notes: This register can only be accessed by software when the CAN is in initialization mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SILM	LBKM	Reserved				SJW[1:0]		Reserv ed	TS2[2:0]			TS1[3:0]			
rw	rw					rw			rw			rw			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						BRP [9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	notation	clarification
31	SILM	<b>SILM:</b> Silent mode (for debugging) (Silent mode (debug)) 0:Normal state; 1:Silent mode.
30	LBKM	<b>LBKM:</b> Loopback mode (debug) 0:Disable loopback mode;



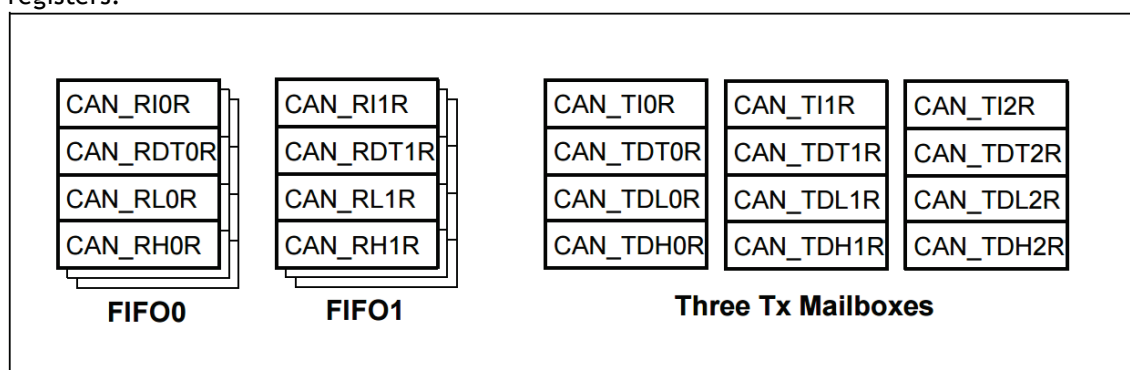
		1:Allow loopback mode.
29:16	Reserved	Reserved bit, hardware forced to 0.
25:24	SJW[1:0]	SJW[1:0]:Resynchronization jump width For resynchronization purposes. this bit field defines an upper limit on how many time cells the CAN hardware can extend or shorten in each bit. $t_{RJW} = t_{CANx}(SJW[1:0]+1)$ .
23	Reserved	Reserved bit, hardware forced to 0.
22:20	TS2[2:0]	TS2[2:0]:Time segment 2 (Timesegment2) This bit field defines how many time cells $t_{BS2} = t_{CANx}(TS2[2:0]+1)$ are occupied by time period 2.
19:16	TS1[3:0]	TS1[3:0]:Time segment1 (Time segment1) This bit field defines how many time cells are occupied by time period 1 $t_{BS1}=t_{CANx}(TS1[3:0]+1)$ For more information on bit time characteristics, refer to the25.7.7 section Bit Time Characteristics.
15:10	Reserved	Reserved bit, hardware forces its value to 0.
9:0	BRP [9:0]	BRP[9:0]:Baud rate prescaler (Baud rate prescaler) This bit field defines the time length of the time unit (tq) $tq=(BRP[9:0]+1)xtPCLK$

## 25.9.3 CAN Mailbox Register

This section describes the transmit and receive mailbox registers. For more information on register images, refer to the25.7.5 section Message Storage. The send and receive mailboxes are virtually identical, with the following exceptions:

- FMI field of the CAN\_RDTxR register;
- The receiving mailbox is read-only;
- The transmit mailbox is writable only when it is empty, and the corresponding TME bit in the CAN\_TSR register is '1', indicating that the transmit mailbox is empty.

There are 3 sending mailboxes and 2 receiving mailboxes. Each receive mailbox is a 3-level deep FIFO and can only access the first received message in the FIFO. Each mailbox contains 4 registers.



RX and TX

Transmit mailbox identifier register (CAN\_TlXR) (x=0..2)

Address Offset:0x180, 0x190, 0x1A0

Reset value:0xFFFF XXXX, X=undefined bit (except bit 0, TXRQ=0 on reset)

Notes:1 This register is write-protected when the mailbox it belongs to is in the wait-to-send state.

2 This register implements the transmit request control function (bit 0) - Reset value is 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
stid[10:0]/exid[28:18]											EXID[17:13]				
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
EXID[12:0]													IDE	RTR	TXRQ
													rw	rw	rw

Bit	notation	clarification
31:21	STID[10:0]/EXID [28:18]	STID[10:0]/EXID[28:18]: Standard identifier or extended identifier (Standard identifier or extended identifier) Depending on the contents of the IDE bits, these are either standard identifiers or the high byte of the extended identity.
20:3	EXID[17:0]	EXID[17:0]:Extended identifier (Extended identifier) The low byte of the extended identity.

2	IDE	<b>IDE:</b> Identifier extension This bit determines the type of identifier used for messages in the sending mailbox 0: Use the standard identifier; 1: Use extended identifiers.
1	RTR	<b>RTR:</b> Remote transmission request 0: Data frame; 1: Remote frames.
0	TXRQ	<b>TXRQ:</b> Transmit mailbox request It is set '1' by the software to request data to be sent to the mailbox. When the data is sent and the mailbox is empty, hardware clears '0' to it.

### Transmit mailbox data length and timestamp register (CAN\_TDTxR) (x=0..2)

When the mailbox is not in the vacant state, all bits of this register are write-protected.

Address Offset: 0x184, 0x194, 0x1A4

Reset value: undefined

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

Bit	notation	clarification
31:16	TIME[15:0]	<b>TIME[15:0]</b> :Message time stamp (Message time stamp) This field contains, at the moment of sending this message SOF, the value of the 16-bit timer.
15:9	Reserved	reserved bit
8	TGT	<b>TGT</b> :Transmit timestamp (Transmit global time) This bit is only valid if the CAN is in time-triggered communication mode, i.e. the TTCM bit of the CAN_MCR register is '1'. 0: No timestamp TIME[15:0] is sent; 1: Send timestamp TIME[15:0]. In a message of length 8, the timestamp TIME[15:0] is the last 2 bytes sent: TIME[7:0] as the 7th byte and TIME[15:8] as the 8th byte, which replace the data written to CAN_TDHxR[31:16] (DATA6[7:0] and DATA7[7:0]). In order to send the 2 bytes of the timestamp, the DLC must be programmed to 8.
7:4	Reserved	Reserved Bits.
3:0	DLC [15:0]	<b>DLC[15:0]</b> :Send data length (Data length code) This field specifies the data length of the data message or the data length of the remote frame request.1 message contains 0 to 8 bytes of data. which is determined by the DLC.

### Transmit mailbox low byte data register (CAN\_TDLxR) (x=0..2)

When the mailbox is not in the vacant state, all bits of this register are write-protected.

Address Offset: 0x188, 0x198, 0x1A8

Reset value: undefined bit

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA3[7:0]								DATA2[7:0]							
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
DATA1[7:0]								DATA0[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	notation	clarification
31:24	DATA3[7:0]	<b>DATA3[7:0]</b> :Data byte3 (Data byte3) Data byte 3 of the message.
23:16	DATA2[7:0]	<b>DATA2[7:0]</b> :Data byte2 (Data byte2) Data byte 2 of the message.
15:8	DATA1[7:0]	<b>DATA1[7:0]</b> :Data byte1 (Data byte1) Data byte 1 of the message.
7:0	DATA0[7:0]	<b>DATA0[7:0]</b> :Data byte0 (Data byte0) Data byte 0 of the message. The message contains 0 to 8 bytes of data and starts at byte 0.

### Transmit mailbox high byte data register (CAN\_TDHxR) (x=0..2)

When the mailbox is not in the vacant state, all bits of this register are write-protected.

Address offset: 0x18C, 0x19C, 0x1AC

Reset value: undefined bit

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA7[7:0]								DATA6[7:0]							
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
DATA5[7:0]								DATA4[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	notation	clarification
31:24	DATA7[7:0]	DATA7[7:0]:Data byte7 (Data byte7) Data byte 7 of the message Note:If the TTCM bit of the CAN_MCR register is '1' and the TGT bit of this mailbox is also '1', then DATA7 and DATA6 will be replaced by the TIME timestamp.
23:16	DATA6[7:0]	DATA6[7:0]:Data byte6 (Data byte6) Data byte 6 of the message.
15:8	DATA5[7:0]	DATA5[7:0]:Data byte5 (Data byte5) Data byte 5 of the message.
7:0	DATA4[7:0]	DATA4[7:0]:Data byte4 (Data byte4) Data byte 4 of the message.

### Receive FIFO mailbox identifier register (CAN\_RlRxR) (x=0..1)

Address offset: 0x1B0, 0x1C0

Reset value: undefined bit

Notes: All receive mailbox registers are read-only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
stid[10:0]/exid[28:18]											EXID[17:13]				
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
EXID[12:0]												IDE	RTR	Reserved	
r	r	r	r	r	r	r	r	r	r	r	r	r	r		

Bit	notation	clarification
32:21	STID[10:0]/EXID [28:18]	STID[10:0]/EXID[28:18]: Standard identifier or extended identifier (Standard identifier or extended identifier) Depending on the contents of the IDE bits, these are either standard identifiers or the high byte of the extended identity.
20:3	EXID[17:0]	EXID[17:0]:Extended identifier (Extended identifier) The low byte of the extended identifier.
2	IDE	IDE: Identifier extension This bit determines the type of identifier used for messages in the receiving mailbox 0: Use standard identifiers; 1: Use extended identifiers.
1	RTR	RTR: Remote transmission request 0: Data frame; 1: Teleframe.
0	Reserved	Reserved Bits.

### Receive FIFO mailbox data length and timestamp register (CAN\_RDTxR) (x=0..1)

Address offset: 0x1B4, 0x1C4

Reset value: undefined

Notes: All receive mailbox registers are read-only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIME[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
FMI [7:0]								Reserved				DLC[3:0]			
r	r	r	r	r	r	r	r					r	r	r	r

Bit	notation	clarification
31:16	TIME[15:0]	TIME[15:0]:Message time stamp This field contains, at the moment of reception of this message SOF, the value of the 16-bit timer.
15:8	FMI [15:0]	FMI[15:0]:Filter match index Here are the filter serial numbers for messaging that exists in the mailbox. For details on identifier filtering, see the 25.7.4 section on filter match numbers.
7:4	Reserved	Reserved bit, hardware forced to 0.
3:0	DLC [15:0]	DLC[15:0]:Receive data length (Data length code) This field indicates the data length of the received data frame (0 ~ 8). For remote frame requests, the data length DLC is constant 0.

### Receive FIFO mailbox low byte data register (CAN\_RDLxR) (x=0..1)

Address offset: 0x1B8, 0x1C8

Reset value: undefined bit

Notes: All receive mailbox registers are read-only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA3[7:0]								DATA2[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
DATA1[7:0]								DATA0[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
Bit	notation	clarification													
31:24	DATA3[7:0]	DATA3[7:0]:Data byte3 (Data byte3) Data byte 3 of the message.													
23:16	DATA2[7:0]	DATA2[7:0]:Data byte2 (Data byte2) Data byte 2 of the message.													
15:8	DATA1[7:0]	DATA1[7:0]:Data byte1 (Data byte1) Data byte 1 of the message.													
7:0	DATA0[7:0]	DATA0[7:0]:Data byte0 (Data byte0) Data byte 0 of the message. The message contains 0 to 8 bytes of data and starts at byte 0.													

### Receive FIFO mailbox high byte data register (CAN\_RDHxR) (x=0..1)

Address offset: 0x1BC, 0x1CC

Reset value: undefined bit

Notes: All receive mailbox registers are read-only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA7[7:0]								DATA6[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
DATA5[7:0]								DATA4[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
Bit	notation	clarification													
31:24	DATA7[7:0]	DATA7[7:0]:Data byte7 (Data byte7) Data byte 7 of the message													
23:16	DATA6[7:0]	DATA6[7:0]:Data byte6 (Data byte6) Data byte 6 of the message.													
15:8	DATA5[7:0]	DATA5[7:0]:Data byte5 (Data byte5) Data byte 5 of the message.													
7:0	DATA4[7:0]	DATA4[7:0]:Data byte4 (Data byte4) Data byte 4 of the message.													

## 25.9.4 CAN Filter Register

### CAN Filter Master Register (CAN\_FMR)

Address Offset:0x200

Reset value:0x2A1C 0E01

Notes: The non-reserved bits of this register are completely controlled by software.

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

Bit	notation	show
31:1	Reserved	Reserved bit, forced to a Reset value.
0	FINIT	FINIT: Filter init mode (Filter init mode) for all the filter group initialization mode settings. 0:Filter set is working in normal mode; 1:The filter group is working in initialization mode.

### CAN Filter Mode Register (CAN\_FM1R)

Address Offset:0x204

Reset value:0x0000 0000

Notes: This register can only be written to if CAN\_FMR (FINIT=1) is set so that the filter is in initialization mode.

Notes: Refer to Figure 203 Filter Group Bit Width Setting-Register Organization

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	FBM13	FBM12	FBM11	FBM10	FBM9	FBM8	FBM7	FBM6	FBM5	FBM4	FBM3	FBM2	FBM1	FBM0	
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	notation	clarification
31:14	Reserved	Reserved bit, forced to a Reset value.
13:0	FBMx	FBMx: Filter mode (Filter mode) The working mode of filter group x. 0:The two 32-bit registers of filter group x operate in identifier mask bit mode; 1:The two 32-bit registers of filter group x operate in identifier list mode.

### CAN Filter Bit Width Register (CAN\_FS1R)

Address offset:0x20C

Reset value:0x0000 0000

Notes: This register can only be written to if CAN\_FMR (FINIT=1) is set so that the filter is in initialization mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	FSC13	FSC12	FSC11	FSC10	FSC9	FSC8	FSC7	FSC6	FSC5	FSC4	FSC3	FSC2	FSC1	FSC0	
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Notes: Refer to Figure 203 Filter Group Bit Width Setting-Register Organization

Bit	notation	clarification
31:14	Reserved	Reserved bit, forced to a Reset value.
13:0	FSCx	FSCx: Filter scale configuration (Filter scale configuration) Bit width of filter group x (13 ~ 0).

		0: The filter bit width is two 16-bit bits; 1: The filter bit width is a single 32 bits.
--	--	---

### CAN Filter FIFO Association Register (CAN\_FFA1R)

Address Offset:0x214

Reset value:0x0000 0000

Notes: This register can only be written to if CAN\_FMR (FINIT=1) is set so that the filter is in initialization mode.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	FFA13	FFA12	FFA11	FFA10	FFA9	FFA8	FFA7	FFA6	FFA5	FFA4	FFA3	FFA2	FFA1	FFA0	
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	notation	clarification
31:14	Reserved	Reserved bit, forced to a Reset value.
13:0	FFAx	FFAx:Filter FIFO assignment for filterx After a message has passed a certain filter, it will be stored in its associated FIFO. 0: Filter is associated to FIFO0; 1: Filter is associated to FIFO1.

### CAN Filter Activation Register (CAN\_FA1R)

Address Offset:0x21C

Reset value:0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	FACT13	FACT12	FACT11	FACT10	FACT9	FACT8	FACT7	FACT6	FACT5	FACT4	FACT3	FACT2	FACT1	FACT0	
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	notation	clarification
31:14	Reserved	Reserved bit, forced to a Reset value.
13:0	FFACTx	FFACTx:Filter active The software sets '1' to a bit to activate the corresponding filter. The corresponding filter register x (CAN_FxR[0:1]) can only be modified after clearing '0' to the FACTx bit or setting '1' to the FINI bit of the CAN_FMR register. 0: Filter is disabled. 1: The filter is activated.

### Register x (CAN\_FiRx) for CAN filter group i (i=0..27,x=1,2)

Address offset: 0x240h...0x31Ch

Reset value: undefined bit

Notes: The W55MH32 has a total of 14 sets of filters: i=0..13. Each set of filters consists of two 32-bit registers, CAN\_FiR[2:1].

The corresponding filter registers can only be modified if the corresponding FACTx bit of the CAN\_FAxR register is clear '0' or the FINIT bit of the CAN\_FMR register is '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FB31	FB30	FB29	FB28	FB27	FB26	FB25	FB24	FB23	FB22	FB21	FB20	FB19	FB18	FB17	FB16
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
FB15	FB14	FB13	FB12	FB11	FB10	FB9	FB8	FB7	FB6	FB5	FB4	FB3	FB2	FB1	FB0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

In all configuration cases:

Bit	notation	clarification
31:0	FB [31:0]	FB[31:0]: Filter bits identifier pattern Each bit of the register corresponds to the level of the corresponding bit of the desired identifier.

		0: expects the corresponding bit to be dominant; 1: expects the corresponding bit to be recessive. Masked Bit Mode Each bit of the register indicates whether or not the corresponding identifier register bit must match the corresponding bit of the desired identifier. 0: Don't care, this bit is not used for comparison; 1: Must match, the arriving identifier bits must match the identifier register bits corresponding to the filter.
--	--	--

Notes: Depending on the different settings of the filter bitwidth and mode, the two registers in the filter group have different functions. See the 25.7.4 section Identifier Filtering mappings, functional descriptions, and mask register associations for filter

The Mask/Identifier registers in Mask Bit mode are the same as the register bit definitions in Identifier List mode. See Table 137 for addresses of the filter group registers.

## 25.9.5 bxCAN Register List

Table 137 List of bxCAN-registers and their Reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0													
000h	CAN_MCR	Reserved																DBF	RESET	Reserved								TTCM	ABOM	AWUM	NART	RFLM	TXFP	SLEEP	INRQ											
	Reset value																	1	0									0	0	0	0	0	0	1	0											
004h	CAN_MSR	Reserved																				RX	SAMP	RXM	TXM	Reserved				SLAKI	WKUI	ERRI	SLAK	INAK												
	Reset value																					1	1	0	0					0	0	0	1	0												
008h	CAN_TSR	LOW [2:0]		TME [2:0]			CODE [1:0]		ABRQ2	Reserved				TERR2	ALST2	TXOK2	RQCP2	ABRQ1	Reserved				TERR1	ALST1	TXOK1	RQCP1	ABRQ0	Reserved				TERR0	ALST0	TXOK0	RQCP0											
	Reset value	0	0	0	1	1	1	0	0	0					0	0	0	0	0					0	0	0	0	0					0	0	0	0										
00Ch	CAN_RFOR	Reserved																								RFOV0	FOVR0	FULL0					Reserved		FMP0 [1:0]											
	Reset value																									0	0	0							0	0										
010h	CAN_RF1F	Reserved																								RFOV1	FOVR1	FULL1					Reserved		FMP1 [1:0]											
	Reset value																									0	0	0							0	0										
014h	CAN_IER	Reserved														SLKIE	WKUE	ERRIE	Reserved				LECIE	BOFIE	EPVIE	EWGIE	Reserved		FOVIE1	FFIE1	FMPIE1	FOVIE0	FFIE0	FMPIE0	TMEIE											
	Reset value															0	0	0					0	0	0	0			0	0	0	0	0	0	0											
018h	CAN_ESR	REC[7:0]								TEC[7:0]								Reserved								LEC[2:0]			Reserved		BOFF	EPVF	EWGF													
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									0	0			0			0	0	0												
01Ch	CAN_BTR	SILW	LBKM	Reserved				SJW [1:0]		Reserved		TS2[2:0]		TS1[3:0]		Reserved									BRP [9:0]																					
	Reset value	0	0					0	0			0	1	0	0	0	1	1							0	0	0	0	0	0	0	0	0	0												
020h~17Fh	Reserved																																													
180h	CAN_TIOR	stid[10:0]/exid[28:18]												EXID[17:0]																IDE		RTR	TXRQ													
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0														

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
184h	CAN_TDTOR	TIME[15:0]																Reserved				TGT	Reserved				DLC[3:0]						
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x									x					x	x	x
188h	CAN_TDLOR	DATA3[7:0]								DATA2[7:0]								DATA1[7:0]								DATA0[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
18Ch	CAN_TDHOR	DATA7[7:0]								DATA6[7:0]								DATA5[7:0]								DATA4[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
190h	CAN_TI1R	stid[10:0]/exid[28:18]												EXID[17:0]														IDE	RTR	TXRQ			
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	
194h	CAN_TDT1R	TIME[15:0]																Reserved				TGT	Reserved				DLC[3:0]						
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x									x					x	x	x
198h	CAN_TDL1R	DATA3[7:0]								DATA2[7:0]								DATA1[7:0]								DATA0[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
19Ch	CAN_TDH1R	DATA7[7:0]								DATA6[7:0]								DATA5[7:0]								DATA4[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
1A0h	CAN_TI2R	stid[10:0]/exid[28:18]												EXID[17:0]														IDE	RTR	TXRQ			
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	
1A4h	CAN_TDT2R	TIME[15:0]																Reserved				TGT	Reserved				DLC[3:0]						
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x									x					x	x	x
1A8h	CAN_TDL2R	DATA3[7:0]								DATA2[7:0]								DATA1[7:0]								DATA0[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
1ACh	CAN_TDH2R	DATA7[7:0]								DATA6[7:0]								DATA5[7:0]								DATA4[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
1B0h	CAN_RIOR	stid[10:0]/exid[28:18]												EXID[17:0]														IDE	RTR	Reserved			
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		
1B4h	CAN_RDTOR	TIME[15:0]																FMI [7:0]				Reserved				DLC[3:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x					x	x	x	x	
1B8h	CAN_RDLOR	DATA3[7:0]								DATA2[7:0]								DATA1[7:0]								DATA0[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
1BCh	CAN_RDHOR	DATA7[7:0]								DATA6[7:0]								DATA5[7:0]								DATA4[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
1C0h	CAN_RI1R	stid[10:0]/exid[28:18]												EXID[17:0]														IDE	RTR	Reserved			
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		
1C4h	CAN_RDT1R	TIME[15:0]																FMI [7:0]				Reserved				DLC[3:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x					x	x	x	x	
1C8h	CAN_RDL1R	DATA3[7:0]								DATA2[7:0]								DATA1[7:0]								DATA0[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
1CCh	CAN_RDH1R	DATA7[7:0]								DATA6[7:0]								DATA5[7:0]								DATA4[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x



SeeTable1 for register start addresses.

## 26 Serial Peripheral Interface (SPI)

### 26.1 SPI Introduction

The SPI interface can be configured to support either the SPI protocol or the I2S audio protocol. the SPI interface operates in SPI mode by default, and the functionality 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 in master mode and provides a communications clock (SCK) for external slave devices. The interface can also operate in a multi-master configuration.

It can be used for a variety of purposes, including two-wire simplex synchronous transmission using one bi-directional data line and also reliable communication using CRC checksums.

I2S is also a 3-pin synchronous serial interface communication protocol. It supports four audio standards, including the Philips I2S standard, the MSB and LSB alignment standards, and the PCM standard. It can operate in 2 modes, master and slave, in half-duplex communication. When it acts as a master device, it provides clock signals to external slave devices through the interface.

**WARNING:** Since some of the SPI3/I2S3 pins are shared with JTAG pins (SPI3\_NSS/I2S3\_WS with JTDI, the

SPI3\_SCK/I2S3\_CK with JTD0), so these pins are not controlled by the IO controller and they are (after each reset) reserved for JTAG use by default. If the user wants to configure the pins for SPI3/I2S3, the JTAG must be turned off (for debugging) and switched to the SWD interface, or (for standard applications) both the JTAG and SWD interfaces must be turned off. See Section 7.3.4 : JTAG/SWD Multiplexing Function Remapping for details.

### 26.2 SPI and I<sup>2</sup>S Key Features

#### 26.2.1 SPI Features

- 3-wire full-duplex synchronous transmission
- Duplex simplex synchronous transmission with or without a third bi-directional data line
- 8 or 16 bit transmission frame format selection
- master or slave operation
- Supports multi-master mode
- 8 master mode baud rate prescaling factors (up to fPCLK/2)
- Slave mode frequency (max. fPCLK/2)
- Fast communication in master and slave mode
- NSS management by software or hardware in both master and slave modes: dynamic change of master/slave operation mode
- Programmable clock polarity and phase
- Programmable data order, MSB first or LSB first
- Dedicated transmit and receive flags that trigger interrupts
- SPI bus busy status flag
- Hardware CRC to support reliable communication
  - In transmit mode, the CRC value can be sent as the last byte
  - Automatic CRC checking of the last byte received in full duplex mode
- Interruptible main mode fault, overload and CRC error flags
- 1-byte transmit and receive buffer with DMA support: generates transmit and receive requests

#### 26.2.2 I<sup>2</sup>S-Function

- Simplex communication (send or receive only)
- master or slave operation
- 8-bit linear programmable prescaler for accurate audio sampling frequency (8KHz to 96kHz)
- The data format can be 16-bit, 24-bit or 32-bit.
- Audio channel fixed packet frames are 16-bit (16-bit data frames) or 32-bit (16-, 24-, or 32-bit data frames)
- Programmable clock polarity (steady state)

- Underflow flag bit in slave transmit mode and overflow flag bit in master/slave receive mode
- 16-bit data registers are used for transmit and receive, one register at each end of the channel
- Supported I2S protocols:
  - I2S Philips Standard
  - MSB alignment standard (left justified)
  - LSB alignment standard (right justified)
  - PCM standard (16-bit channel frames with long or short frame synchronization or 16-bit data frames extended to 32-bit channel frames)
- Data direction is always MSB first
- DMA capability for both transmit and receive
- The master clock can be output to an external audio device with a fixed ratio of 256x $F_s$  ( $F_s$  is the audio sampling frequency)

## 26.3 SPI Functional Description

### 26.3.1 Summarize

The block diagram of SPI is shown below.

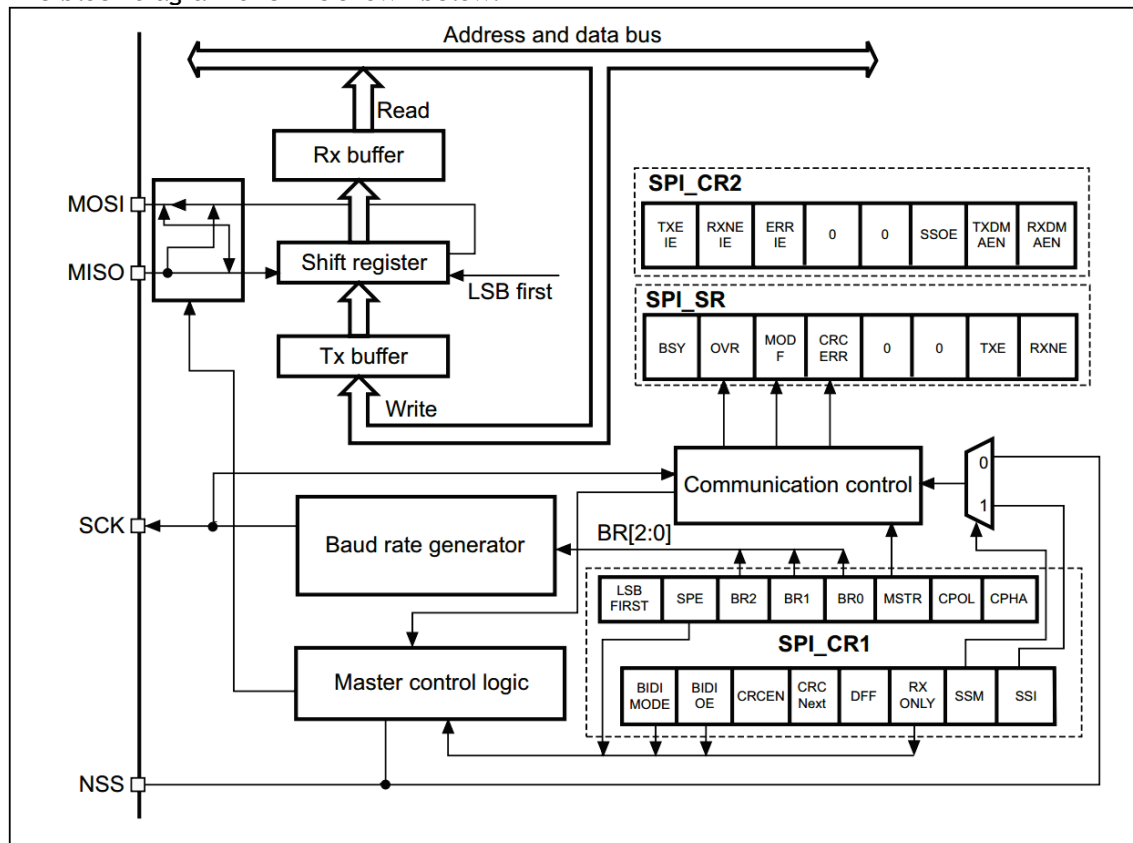


Figure210 SPI Block Diagram

Typically SPI is connected to external devices via 4 pins:

- **MISO**: Master device input/slave device output pin. This pin sends data in slave mode and receives data in master mode.
- **MOSI**: Master device output/slave device input pin. This pin sends data in master mode and receives data in slave mode.
- **SCK**: Serial port clock, as output from master device, input from slave device
- **NSS**: Slave device selection. This is an optional pin to select the master/slave device. It is intended to be used as a "chip select pin", allowing the master device to communicate with specific slave devices individually, avoiding conflicts on the data lines. The NSS pin of the slave device can be driven by a standard I/O pin of the master device. Once enabled (the SSOE bit), the NSS pin can also be used as an output pin and pulled low when the SPI is in master mode; at this point, all SPI devices whose NSS pins are connected to the master's NSS pin will detect a low level and automatically enter the slave device state if they are set to NSS hardware mode.

When configured as a master device and NSS is configured as an input pin (MSTR=1, SSOE=0), if NSS is pulled low, this SPI device enters the master mode failure state: i.e., the MSTR bit is automatically cleared and this device enters the slave mode (see)

The following figure shows an example of a single master and single slave device interconnection.

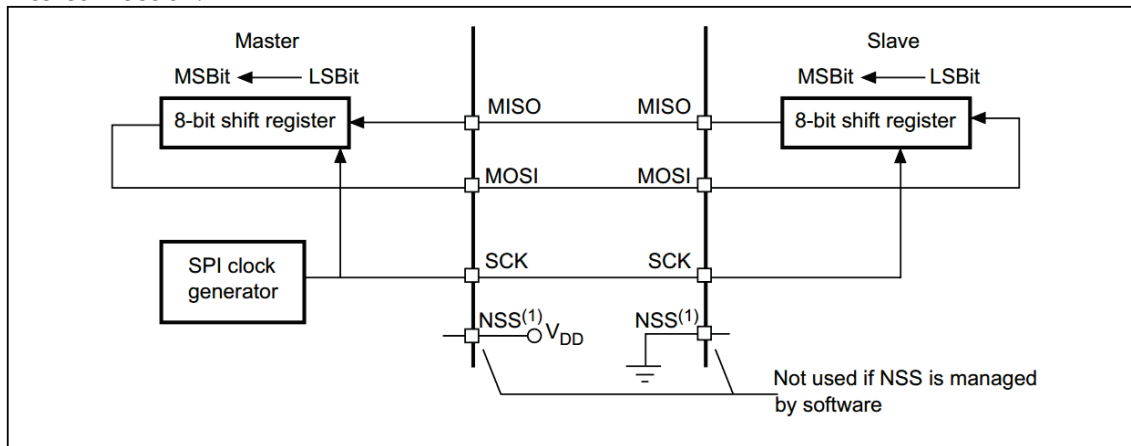


Figure 211 Single Master and Single Slave Applications

1. Here the NSS pin is set as the input

The MOSI pins are connected to each other and the MISO pins are connected to each other. In this way, data is transmitted serially between master and slave (MSB bit comes first).

Communication is always initiated by the master device. The master device sends data to the slave device via the MOSI pin, and the slave device sends data back via the MISO pin. This means that the data output and data input of full-duplex communication are synchronized with the same clock signal; the clock signal is provided by the master device via the SCK pin.

### Slave select (NSS) pin management

Hardware or software slave select management can be set using the SSM bit in the SPI\_CR1 register.

- Software NSS management (SSM = 1): The slave select information is driven internally by the value of the SSI bit in the SPI\_CR1 register. The external NSS pin remains free for other application uses.
- Hardware NSS management (SSM = 0): Two configurations are possible depending on the NSS output configuration (SSOE bit in register SPI\_CR2).
  - NSS output enabled (SSM = 0, SSOE = 1): This configuration is used only when the device operates in master mode. The NSS signal is driven low when the master starts the communication and is kept low until the SPI is disabled.
  - NSS output disabled (SSM = 0, SSOE = 0): This configuration allows multimaster capability for devices operating in master mode. For devices set as slave, the NSS pin acts as a classical NSS input: the slave is selected when NSS is low and deselected when NSS is high.

### Phase and polarity of clock signals

The CPOL and CPHA bits of the SPI\_CR register are able to be combined to form four possible timing relationships. The CPOL (clock polarity) bit controls the idle state level of the clock when there is no data being transmitted, and is valid for devices in both master and slave modes. If CPOL is cleared '0', the SCK pin remains low in the idle state; if CPOL is set '1', the SCK pin remains high in the idle state. If the CPHA (clock phase) bit is set to '1', the data bits are sampled on the second edge of the SCK clock (falling edge if the CPOL bit is 0, rising edge if the CPOL bit is '1') and the data is latched on the second clock edge. If the CPHA bit is cleared '0', the first edge of the SCK clock (which is the rising edge when the CPOL bit is '0' and the falling edge when the CPOL bit is '1') is sampled for data bits, and the data is latched at the first clock edge.

The combination of CPOL clock polarity and CPHA clock phase selects the clock edge for data capture.

Figure 212 shows the four CPHA and CPOL bit combinations for SPI transfers. This figure can be interpreted as a master or slave timing diagram with the SCK pin, MISO pin, and MOSI pin of the master and slave devices directly connected.

- Notes:
1. the SPE bit must be cleared to disable SPI before changing the CPOL/CPHA bit.
  2. The master and slave must be configured in the same timing mode.

3. The idle state of SCK must be the same as the polarity specified in the SPI\_CR1 register (when CPOL is '1', SCK should be pulled up to a high level when idle; when CPOL is '0', SCK should be pulled down to a low level when idle).
4. The data frame format (8-bit or 16-bit) is selected by the DFF bit in the SPI\_CR1 register and determines the length of data to be sent/received.

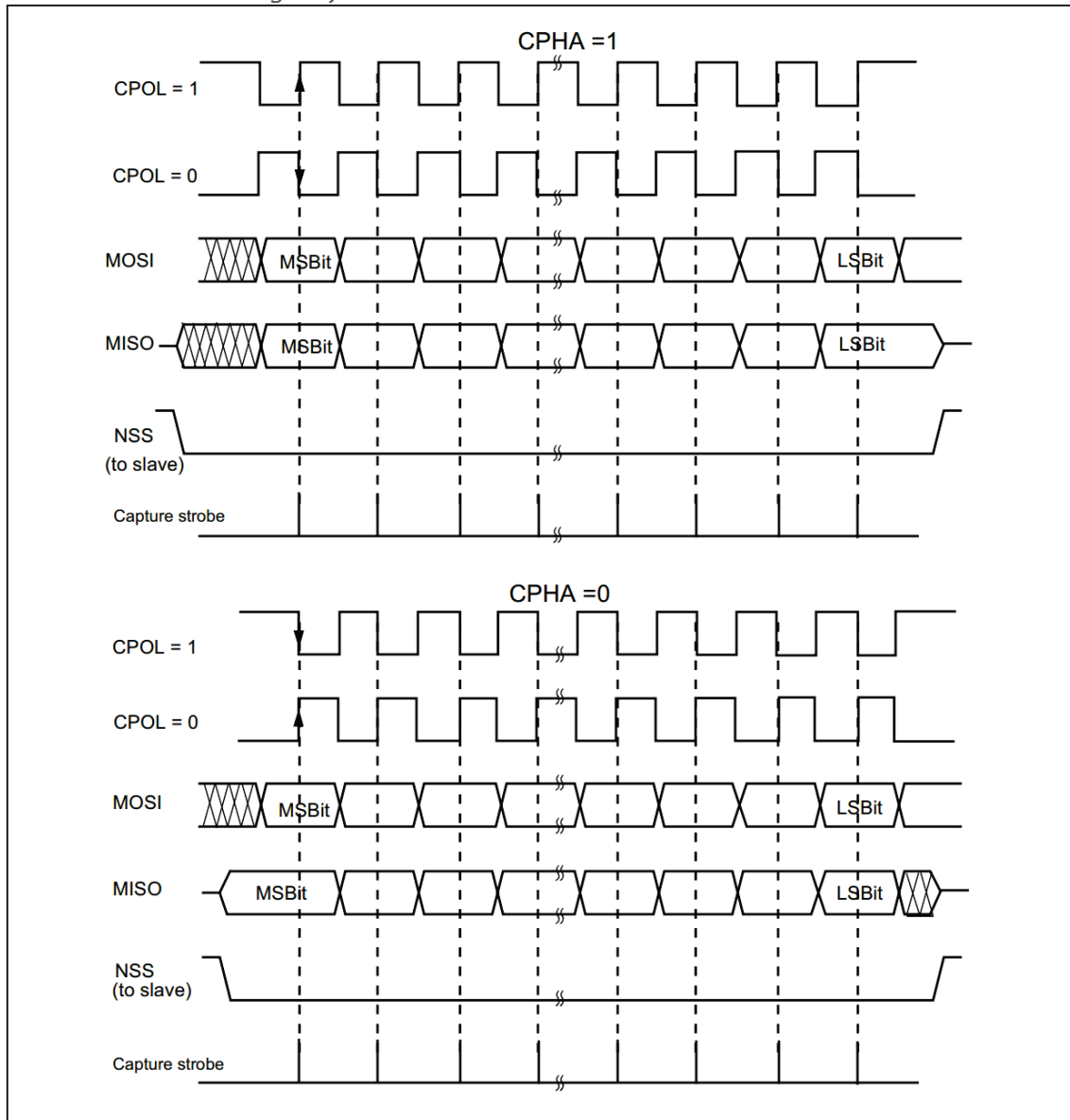


Figure 212 Data Clock Timing Diagram

1. These timings are shown with the LSBFIRST bit reset in the SPI\_CR1 register.

#### Data Frame Format

Depending on the LSBFIRST bit in the SPI\_CR1 register, the data bits can be output MSB first or LSB first.

Each data frame can be either 8-bit or 16-bit depending on the DFF bit in the SPI\_CR1 register. The selected data frame format is valid for both transmission and/or reception.

### 26.3.2 Configuring SPI for Slave Mode

In slave mode, the SCK pin is used to receive the serial clock from the master device. the setting of BR[2:0] in the SPI\_CR1 register does not affect the data transfer rate.

**Notes:** It is recommended to enable the SPI slave device before the master device sends the clock, otherwise unexpected data transfer may occur. The SPI slave device is enabled at the communication clock's first

*The data registers of the slave device must be ready before an edge arrives or before the ongoing communication ends. The polarity of the communication clock must be at a stable value before enabling the slave and master devices.*

Follow the steps below to configure the SPI for slave mode:

#### Configuration Steps

1. Set the DFF bit to define the data frame format as 8-bit or 16-bit.
2. The CPOL and CPHA bits are selected to define the phase relationship between the data transfer and the serial clock (see Figure 213). To ensure proper data transfer, the CPOL and CPHA bits must be configured the same way for both the slave and master devices.
3. The frame format ("MSB first" or "LSB first" as defined by the LSBFIRST bit in the SPI\_CR1 register) must be the same as the master device.
4. In hardware mode (refer to the Slave Select (NSS) Pin Management section), the NSS pin must be low during a complete data frame (8-bit or 16-bit) transfer. In NSS software mode, set the SSM bit in the SPI\_CR1 register and clear the SSI bit.
5. Clear the MSTR bit, set the SPE bit (SPI\_CR1 register) to make the corresponding pin work in SPI mode.

In this configuration, the MOSI pin is the data input and the MISO pin is the data output.

#### Data Transmission Process

In a write operation, data words are written to the transmit buffer in parallel.

The transmit process begins when the slave device receives the clock signal and the first data bit appears on the MOSI pin (translation: the first bit is sent out at this point). The remaining bits (7 more bits for the 8-bit data frame format and 15 more bits for the 16-bit data frame format) are loaded into the shift register. When the data in the transmit buffer is transferred to the shift register, the TXE flag of the SPI\_SP register is set, and an interrupt will be generated if the TXEIE bit of the SPI\_CR2 register is set.

#### Data Reception Process

For receivers, when data reception is complete:

- The data in the shift register is transferred to the receive buffer and the RXNE flag in the SPI\_SR register is set.
- If the RXNEIE bit in the SPI\_CR2 register is set, an interrupt is generated.

After the last sample clock edge, the RXNE bit is set to '1' and the received data byte in the shift register is transferred to the receive buffer. When the SPI\_DR register is read, the SPI device returns the value of this receive buffer.

The RXNE bit is cleared when the SPI\_DR register is read.

### 26.3.3 Configuring SPI Master Mode

In the main configuration, a serial clock is generated at the SCK pin.

#### Configuration Steps

1. The serial clock baud rate is defined via the BR[2:0] bits of the SPI\_CR1 register.
2. Select the CPOL and CPHA bits to define the phase relationship between the data transfer and the serial clock (see Figure 212).
3. Set the DFF bit to define the 8-bit or 16-bit data frame format.
4. Configure the LSBFIRST bit of the SPI\_CR1 register to define the frame format.
5. If the NSS pin is required to operate in input mode, the NSS pin should be connected high during the entire data frame transmission in hardware mode; in software mode, the SSM bit and SSI bit of the SPI\_CR1 register need to be set. If the NSS pin operates in output mode, only the SSOE bit needs to be set.
6. The MSTR bit and the SPE bit must be set (these bits remain set only when the NSS pin is connected high).

In this configuration, the MOSI pin is the data output and the MISO pin is the data input.

#### Data Transmission Process

When data is written to the transmit buffer, the transmit process begins.

On sending the first data bit, the data word is passed in parallel (over the internal bus) into the shift register and later shifted out serially onto the MOSI pin; whether MSB comes first or LSB

comes first depends on the setting of the LSBFIRST bit in the SPI\_CR1 register. The TXE flag will be set when data is transferred from the transmit buffer to the shift register, and an interrupt will be generated if the TXEIE bit in the SPI\_CR1 register is set.

#### Data Reception Process

For the receiver, when the data transfer is complete:

- The data in the shift register is transmitted to the receive buffer and the RXNE flag is set.
- If the RXNEIE bit in the SPI\_CR2 register is set, an interrupt is generated.

At the last sample clock edge, the RXNE bit is set and the data word received in the shift register is transferred to the receive buffer. On reading the SPI\_DR register, the SPI device returns the data in the receive buffer.

Reading the SPI\_DR register will clear the RXNE bit.

Once a transmission has begun, a continuous stream of transmissions can be maintained if the next data that will be sent is put into the transmit buffer. Before attempting to write the transmit buffer, it is necessary to verify that the TXE flag should be '1'.

*Notes: In NSS hardware mode, the NSS input to the slave device is controlled by the NSS pin or another software-driven GPIO pin.*

### 26.3.4 Configure SPI for Simplex Communication

The SPI module is capable of operating in simplex mode in two configurations:

- 1 clock line and 1 bidirectional data line;
- 1 clock line and 1 data line (receive only or transmit only);

#### 1 Clock Line and 1 Bidirectional Data Line (BIDIMODE=1)

This mode is enabled by setting the BIDIMODE bit in the SPI\_CR1 register. In this mode, the SCK pin is used as the clock, the master device uses the MOSI pin and the slave devices use the MISO pin for data communication. The direction of transmission is controlled by BIDIOE in the SPI\_CR1 register, when this bit is '1', the data line is an output, otherwise it is an input.

#### 1 Clock and 1 Unidirectional Data Line (BIDIMODE=0)

In this mode, the SPI module can either be used as a transmit-only or a receive-only.

- Transmit-only mode is similar to full-duplex mode (BIDIMODE=0, RXONLY=0): data is transmitted on the transmit pins (MOSI in master mode, MISO in slave mode), and the receive pins (MISO in master mode, MOSI in slave mode) can be used as general-purpose I/Os. At this point, software does not have to bother with the data in the receive buffer (if the data register is read, it does not contain any receive data).

- In receive-only mode, the SPI output function can be turned off by setting the RXONLY bit in the SPI\_CR2 register; at this point, the transmit pins (MOSI in master mode, MISO in slave mode) are freed up and can be used for other functions.

The way to configure and enable the SPI module for receive-only mode is:

- When in master mode, communication starts as soon as SPI is enabled and stops the current reception as soon as the SPE bit is cleared. In this mode, there is no need to read the BSY flag, this flag is always '1' during SPI communication.

- In slave mode, SPI is always receiving as long as NSS is pulled low (or the SSI bit is '0' in NSS software mode) while SCK has a clock pulse.

### 26.3.5 The Process of Sending and Receiving Data

#### Receive and Transmit Buffers

On receive, the received data is stored in an internal receive buffer; on transmit, the data will first be stored in an internal transmit buffer before being sent.

A read operation to the SPI\_DR register will return the contents of the receive buffer; data written to the SPI\_DR register will be written to the transmit buffer.

#### Starting Transmission in Main Mode

- Full duplex mode (BIDIMODE=0 and RXONLY=0)
  - The transmission starts when data is written to the SPI\_DR register (transmit buffer);



- While transmitting the first bit of data, the data is transferred in parallel from the transmit buffer to the 8-bit shift register and then serially shifted to the MOSI pin in sequence;
- At the same time, the data received on the MISO pin is sequentially shifted serially into the 8-bit shift register and then transferred in parallel into the SPI\_DR register (receive buffer).
- Unidirectional receive-only mode (BIDIMODE=0 and RXONLY=1)
  - Transmission starts when SPE=1;
  - Only when the receiver is activated, the data received on the MISO pin is sequentially shifted serially into the 8-bit shift register and then transferred in parallel into the SPI\_DR register (receive buffer).
- Bidirectional mode, when sending (BIDIMODE=1 and BIDIOE=1)
  - The transmission starts when data is written to the SPI\_DR register (transmit buffer);
  - While transmitting the first bit of data, the data is transferred in parallel from the transmit buffer to the 8-bit shift register and then serially shifted to the MOSI pin in sequence;
  - No data is received.
- Bidirectional mode, on receive (BIDIMODE=1 and BIDIOE=0)
  - Transmission starts when SPE=1 and BIDIOE=0;
  - The data received on the MOSI pin is sequentially shifted serially into the 8-bit shift register and then transferred in parallel into the SPI\_DR register (receive buffer).
  - No transmitter is activated and no data is sent serially to the MOSI pin.

#### Transmission From Mode

- Full duplex mode (BIDIMODE=0 and RXONLY=0)
  - When the slave device receives a clock signal and the first data bit appears in its MOSI, the data transfer begins and subsequent data bits move sequentially into the shift register;
  - Meanwhile, when the first data bit is transmitted, the data in the transmit buffer is transferred in parallel to the 8-bit shift register and subsequently sent serially to the MISO pin. The software must ensure that the data to be sent is written in the transmit register before the SPI master device starts the data transfer.
- Unidirectional receive-only mode (BIDIMODE=0 and RXONLY=1)
  - The data transfer begins when the slave device receives the clock signal and the first data bit appears in its MOSI, followed by the sequential movement of the data bits into the shift register;
  - No transmitter is activated and no data is transmitted serially to the MISO pin.
- Bidirectional mode, when sending (BIDIMODE=1 and BIDIOE=1)
  - The data transfer starts when the slave device receives the clock signal and the first data bit in the transmit buffer is transferred to the MISO pin;
  - At the same time as the first data bit is transferred to the MISO pin, the data to be sent in the transmit buffer is transferred in parallel to the 8-bit shift register and subsequently sent serially to the MISO pin. The software must ensure that the data to be sent is written in the transmit register before the SPI master device starts the data transfer;
  - No data is received.
- Bidirectional mode, on receive (BIDIMODE=1 and BIDIOE=0)
  - The data transfer starts when the slave device receives the clock signal and the first data bit appears in its MOSI;
  - The data received on the MISO pin is serially transferred into an 8-bit shift register and then transferred in parallel to the SPI\_DR register (receive buffer);
  - No transmitter is activated and no data is transmitted serially to the MISO pin.

#### Handling the Sending and Receiving of Data

When data is transferred from the transmit buffer to the shift register, the TXE flag (transmit buffer empty) is set, which indicates that the internal transmit buffer is ready to receive the next data; if the TXEIE bit is set in the SPI\_CR2 register, an interrupt is generated at this point; the TXE bit is cleared by writing to the SPI\_DR register.

*Notes: Before writing to the transmit buffer, the software must verify that the TXE flag is '1', otherwise the new data will overwrite the data already in the transmit buffer.*



On the last edge of the sampling clock, the RXNE flag is set when data is being transferred from the shift register to the receive buffer (the receive buffer is not empty); it indicates that the data is ready to be read from the SPI\_DR register; if the RXNEIE bit is set in the SPI\_CR2 register, an interrupt is generated at this point; reading the SPI\_DR register clears the RXNE flag bit. In some configurations, the BSY flag can be used when transmitting the last data to wait for the end of the data transfer.

### Full Duplex Transmit and Receive Process Mode in Master or Slave Mode (BIDIMODE=0 and RXONLY=0)

The software must follow the process described below to send and receive data (see Figure213 and Figure214 ):

1. Set the SPE bit to '1' to enable the SPI module;
2. Write the first data to be sent in the SPI\_DR register, this operation clears the TXE flag;
3. Wait for TXE=1 and then write the second data to be sent. Wait for RXNE=1, then read the SPI\_DR register and get the first data received, clearing the RXNE bit while reading SPI\_DR. Repeat these operations to send subsequent data while receiving n-1 data;
4. Wait for RXNE=1 and then receive the last data;
5. Wait for TXE=1 and turn off the SPI module after BSY=0.

It is also possible to implement this process in a handler that responds to an interrupt generated by the rising edge of the RXNE or TXE flags.

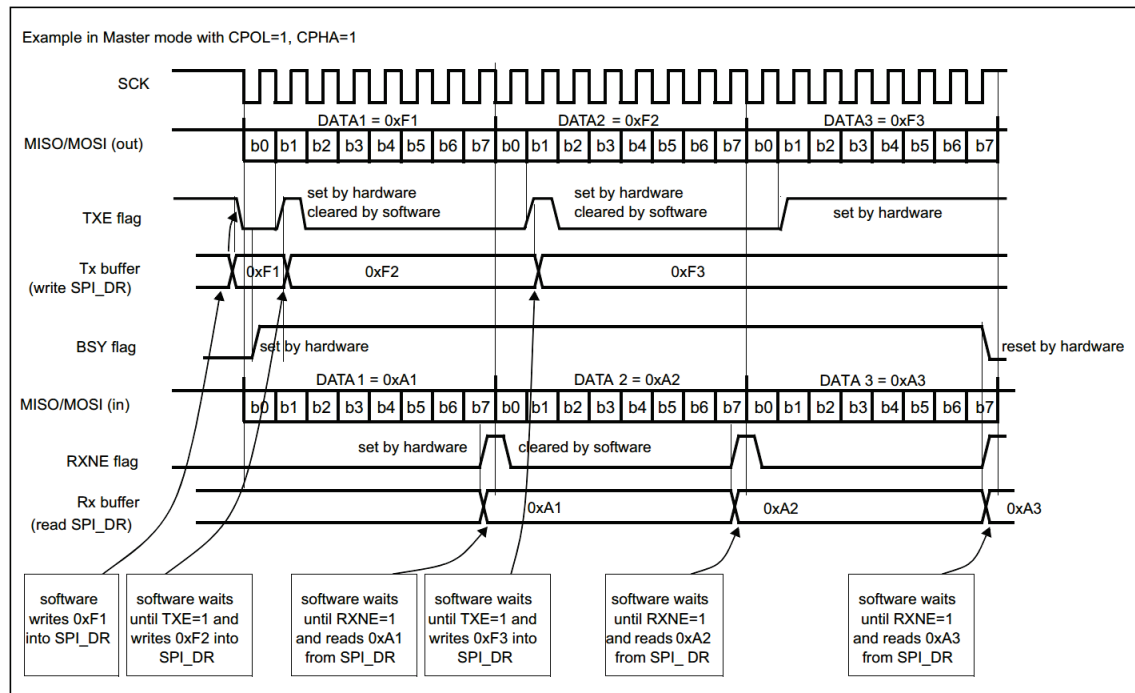


Figure213 Diagram of TXE/RXNE/BSY changes during continuous transmission in main mode, full duplex mode (BIDIMODE=0 and RXONLY=0)

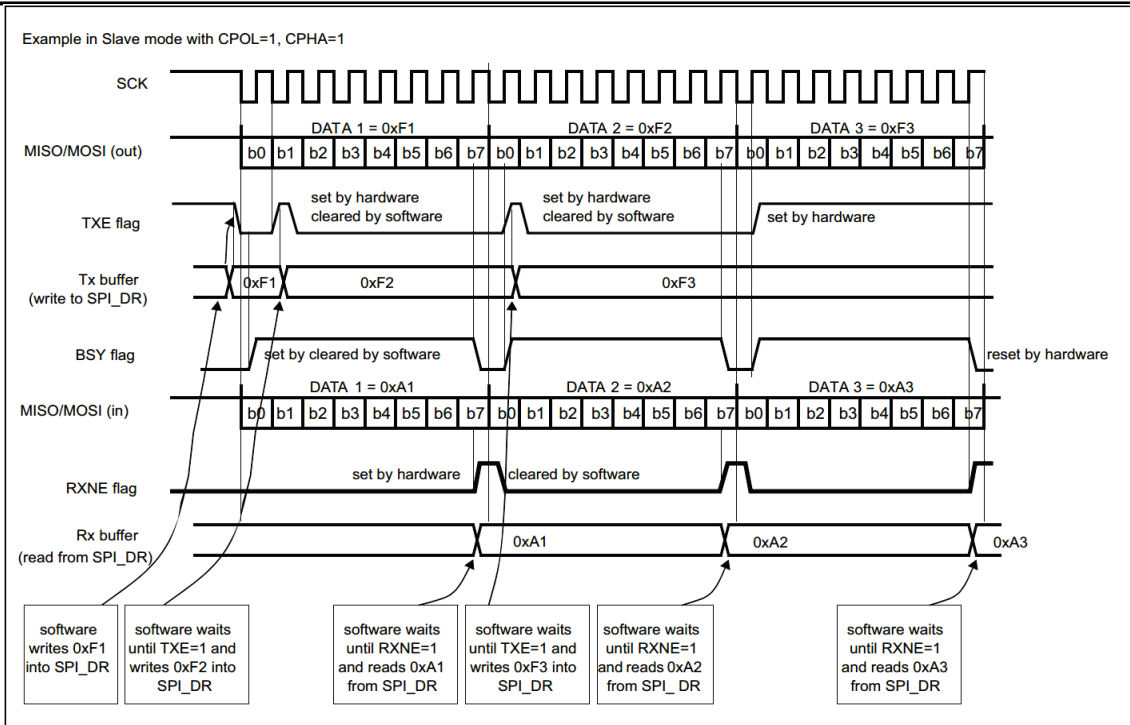


Figure214 Schematic diagram of TXE/RXNE/BSY changes during continuous transmission in slave mode, full duplex mode (BIDIMODE=0 and RXONLY=0)

### Send Only Process (BIDIMODE=0 and RXONLY=0)

In this mode, the transmission process can be briefly described as follows, using the BSY bit to wait for the end of the transmission (see Figure215 and Figure216 ):

1. Set the SPE bit to '1' to enable the SPI module;
2. Write the first data to be sent in the SPI\_DR register, this operation clears the TXE flag;
3. Wait for TXE=1 and then write the second data to be sent. Repeat this operation to send the subsequent data;
4. After writing the last data to the SPI\_DR register, wait for TXE=1; then wait for BSY=0, which indicates that the transfer of the last data has been completed.

It is also possible to implement this process in the handler of an interrupt generated in response to the rising edge of the TXE flag.

- Notes:
1. For discontinuous transmission, there is a delay of 2 APB clock cycles between the operation of writing to the SPI\_DR register and setting the BSY bit, so in transmit-only mode, after writing the last data, it is better to wait for TXE=1 and then wait for BSY=0.
  2. In transmit-only mode, after 2 data transfers, the OVR bit in the SPI\_SR register changes to '1' since the received data will not be read. The software does not have to pay attention to this OVR flag bit.

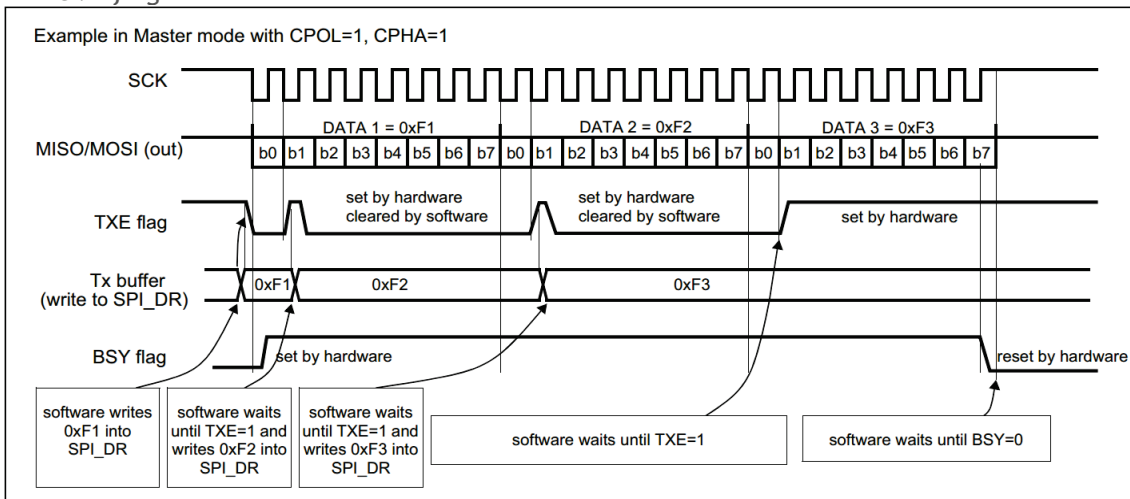


Figure215 Intention of TXE/BSY change during continuous transmission in the master device transmit-only mode (BIDIMODE=0 and RXONLY=0)

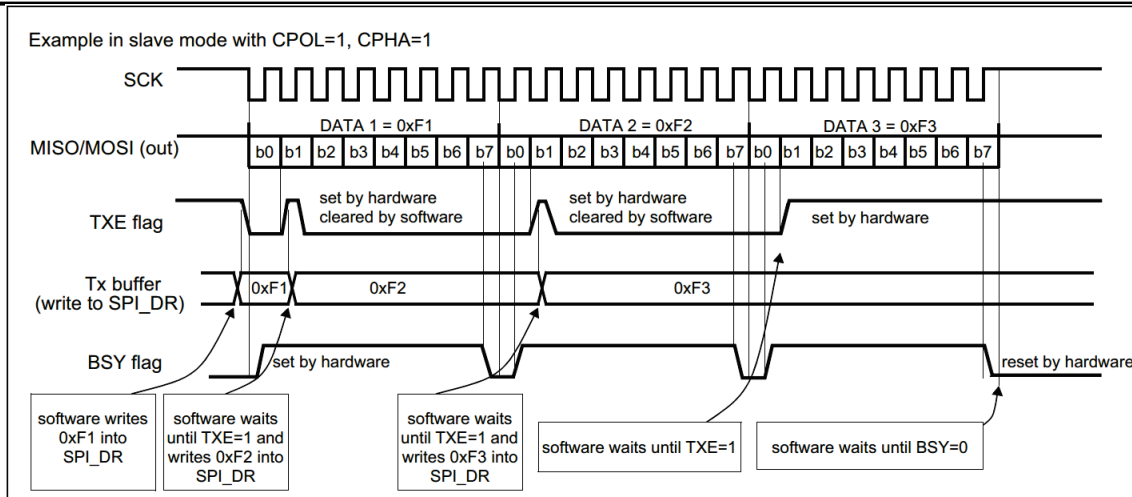


Figure216 Diagram of TXE/BSY change during continuous transmission in transmit-only mode (BIDIMODE=0 and RXONLY=0) from slave device

### Bidirectional Sending Process (BIDIMODE=1 and BIDIOE=1)

In this mode, the operation procedure is similar to the transmit-only mode, with one difference: before enabling the SPI module, it is necessary to set both the BIDIMODE and BIDIOE bits to '1' in the SPI\_CR2 register.

### Unidirectional Receive-Only Mode (BIDIMODE=0 and RXONLY=1)

In this mode, the transmission process can be briefly described as follows (see):

1. In the SPI\_CR2 register, set RXONLY=1;
2. Set SPE=1 to enable the SPI module:
  - a) In main mode, the SCK clock signal is generated immediately, and serial data is received continuously until the SPI is turned off (SPE=0);
  - b) In slave mode, serial data is received when the SPI master device pulls down the NSS signal and generates the SCK clock.
3. Wait for RXNE=1, then read the SPI\_DR register for received data (which will also clear the RXNE bit). Repeat this operation to receive all data.

It is also possible to implement this process in the handler of an interrupt generated in response to the rising edge of the RXNE flag.

**Notes:** If the SPI module, follow the recommendations in section26.3.8 .is turned off after the last data transfer

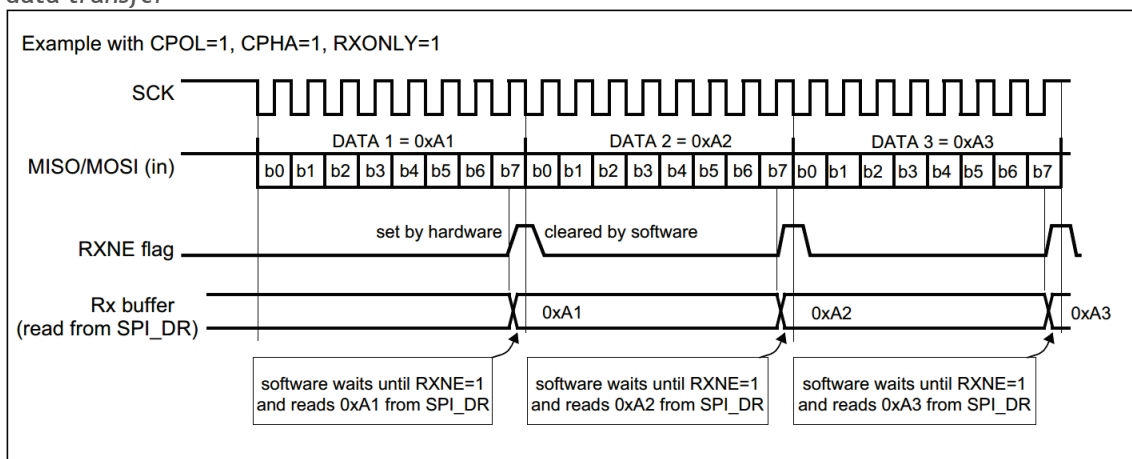


Figure217 Schematic diagram of RXNE variation during continuous transmission in receive-only mode (BIDIMODE=0 and RXONLY=1)

### One-way Receive Process (BIDIMODE=1 and BIDIOE=0)

In this mode, the operation procedure is similar to the receive-only mode, except that: before enabling the SPI module, it is necessary to set BIDIMODE to '1' and clear the BIDIOE bit to '0' in the SPI\_CR2 register.

## Continuous and Discontinuous Transmission

When sending data in master mode, continuous communication is possible if the software is fast enough to detect each rising edge of TXE (or TXE interrupt) and write to the SPI\_DR register immediately before the end of the ongoing transmission; at this point, the SPI clock is kept continuous between the transmission of each data item, while the BSY bit is not cleared.

If the software is not fast enough, this results in discontinuous communication; this is then cleared between each data transfer (see below). In the receive-only mode of the master mode (RXONLY=1), the communication is always continuous and the BSY flag is always '1'.

In slave mode, the continuity of communication is determined by the SPI master device. Regardless, even if communication is continuous, the BSY flag will be low for at least one SPI clock cycle between each data item (see Figure 216).

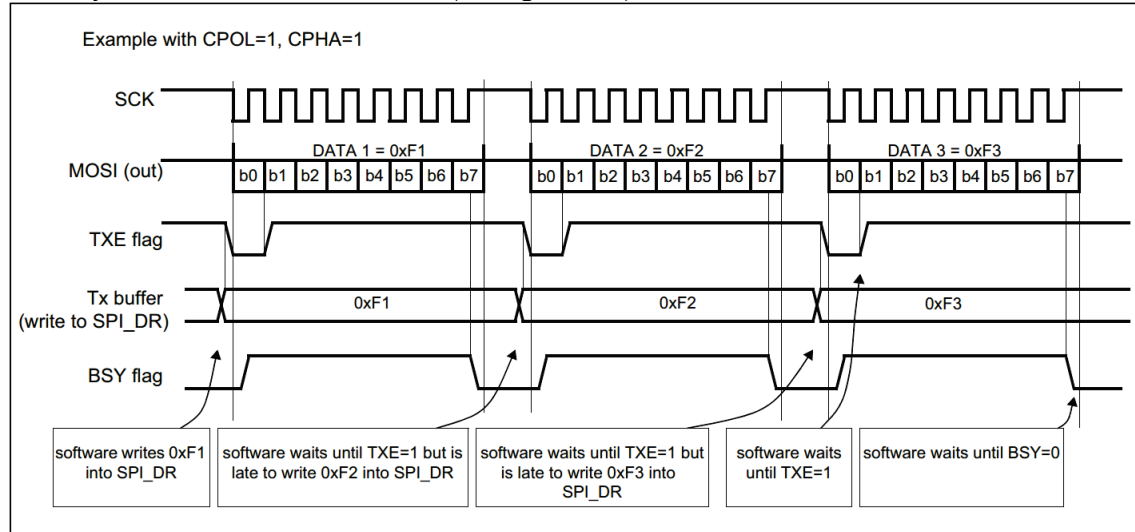


Figure 218 Schematic diagram of TXE/BSY variation when sending in discontinuous transmission (BIDIMODE=0 and RXONLY=0)

## 26.3.6 CRC Calculations

CRC checks are used to ensure the reliability of full-duplex communications. Separate CRC calculators are used for data transmission and data reception. The CRC is calculated by performing a programmable polynomial operation on each receive bit. The CRC is calculated on the sampling clock edge defined by the CPHA and CPOL bits in the SPI\_CR1 register.

**Notes:** This SPI interface provides two methods of CRC calculation, depending on the data frame format selected for sending and/or receiving: 8-bit data frames take CR8 is used; CRC16 is used for 16-bit data frames.

CRC calculation is enabled by setting the CRCEN bit in the SPI\_CR1 register. Setting the CRCEN bit also resets the CRC registers (SPI\_RXCRCR and SPI\_TXCRCR). When the CRCNEXT bit of SPI\_CR1 is set, the contents of SPI\_TXCRCR will be sent after the current byte is sent.

When transmitting the contents of SPI\_TXCRCR, if the value received in the shift register does not match the contents of SPI\_RXCRCR, the CRCERR flag bit of the SPI\_SR register is set to one. If there is still data in the TX buffer, the value of the CRC is transmitted only at the end of the data byte transmission. During the transmission of the CRC, the CRC calculator is turned off and the value of the register remains unchanged.

**Notes:** Please refer to the product manual to confirm that this feature is available (not all models have this feature).

SPI communication can use CRC by following the steps below:

- Sets the values of CPOL, CPHA, LSBFirst, BR, SSM, SSI, and MSTR;
- Enter a polynomial in the SPI\_CRCPR register;
- CRC computation is enabled by setting the SPI\_CR1 register CRCEN bit, this operation also clears the registers SPI\_RXCRCR and SPI\_TXCRCR;
- Setting the SPE bit of the SPI\_CR1 register initiates the SPI function;
- Communication is initiated and maintained until only the last byte or half word remains;
- Setting the CRCNext bit of SPI\_CR1 when writing the last byte or halfword to the transmit buffer instructs the hardware to send the value of the CRC after the last data has been sent. During the sending of the CRC value, the CRC calculation is stopped;

- When the last byte or half word is sent, the SPI sends the CRC value and the CRCNext bit is cleared. Similarly, the received CRC is compared to the SPI\_RXCRCR value, and if the comparison does not match, the CRCERR flag bit on SPI\_SR is set, and an interrupt is generated when ERRIE is set in the SPI\_CR2 register.

*Notes: When the SPI module is in slave device mode, please be careful to enable the CRC calculation after the clock has stabilized, otherwise you may get the wrong CRC calculation results. In fact, as long as the CRCEN bit is set, the CRC calculation is performed as long as there is an input clock on the SCK pin, regardless of the state of the SPE bit.*

When the SPI clock frequency is high, the user must be careful when sending CRCs. During the CRC transmission, the CPU should be used for as little time as possible; in order to avoid errors during the reception of the last data and the CRC, function calls should be disabled during the sending of the CRC. Setting the CRCNEXT bit must be done before sending/receiving the last data.

When the SPI clock frequency is high, because the CPU operation affects the SPI bandwidth, it is recommended to use DMA mode to avoid the SPI reduced speed.

When configured in slave mode and NSS hardware mode is used, the NSS pin should remain low during data transfers and CRC transfers.

When the SPI is configured in slave mode and the CRC function is used, the CRC calculation is performed even if the NSS pin is high (Note: when the NSS signal is high, the CRC calculation will continue to be performed if there is a clock pulse on the SCK pin). This will occur, for example, when the master device communicates alternately with multiple slave devices (translation: find a way to avoid CRC misoperation at this point).

When switching from not selecting a slave device (NSS signal is high) to selecting a new slave device (NSS signal is low), the CRC values of both master and slave should be cleared in order to keep the results of the next CRC calculation at the master and slave ends synchronized.

Follow the steps below to clear the CRC value:

1. Shut down the SPI module (SPE=0).
2. Clear the CRCEN bit to '0';
3. Set the CRCEN bit to '1';
4. Enable SPI module (SPE=1).

## 26.3.7 Status Flag

The application program can completely monitor the status of the SPI bus through 3 status flags.

### Transmit Buffer Idle Flag (TXE)

A '1' for this flag indicates that the transmit buffer is empty and the next data to be sent can be written into the buffer. The TXE flag is cleared when SPI\_DR is written.

### Receive Buffer Non-Null (RXNE)

A '1' for this flag indicates that valid receive data is contained in the receive buffer. Reading the SPI data register clears this flag.

### Bus Sign

The BSY flag is set and cleared by hardware (writing this bit has no effect), this flag indicates the state of the SPI communication layer.

When it is set to '1', it indicates that the SPI is busy communicating with one exception: in master mode bi-directional receive mode (MSTR=1, BDM=1 and BDOE=0), the BSY flag is kept low during reception.

Before the software has to shut down the SPI module and enter shutdown mode (or turn off the device clock), the BSY flag can be used to detect whether the transmission is

The BSY flag can also be used to avoid write conflicts in a multi-master system. end. which avoids corrupting the last transmission. so the following procedure needs to be followed closely. In addition to the main mode of bi-directional reception (MSTR=1, BDM=1 and BDOE=0), the BSY flag is set to '1' when transmission starts.

This flag is cleared to '0' in the following cases:

- When the transmission ends (in the main mode, with the exception of the case of continuous communication);
- When the SPI module is turned off;

- When a Master Mode Failure is generated (MODF=1).  
If communication is not continuous, the BSY flag is low between the transmission of each data item.

When communication is continuous:

- In master mode: the BSY flag remains high throughout the transmission;
- In slave mode: the BSY flag is low for one SPI clock cycle between the transmission of each data item.

*Notes: Do not use the BSY flag to handle the sending and receiving of each data item; it is better to use the TXE and RXNE flags.*

## 26.3.8 Shutdown SPI

When communication ends, it can be terminated by shutting down the SPI module. Clearing the SPE bit shuts down the SPI.

In some configurations, shutting down the SPI module and entering shutdown mode before the transfer is complete can result in the current transfer being corrupted and the BSY flag becoming untrustworthy.

To avoid this, it is recommended to follow the steps below when shutting down the SPI module:

### Full Duplex Mode in Master or Slave Mode (BIDIMODE=0, RXONLY=0)

1. Wait for RXNE=1 and receive the last data;
2. Wait for TXE=1;
3. Wait for BSY=0;
4. Shut down SPI (SPE=0) and finally enter shutdown mode (or turn off the clock for that module).

### Unidirectional Transmit-Only Mode in Master or Slave Mode (BIDIMODE=0, RXONLY=0) or

### Bidirectional Transmit Mode (BIDIMODE=1, BIDIOE=1)

After writing the last data in the SPI\_DR register:

1. Wait for TXE=1;
2. Wait for BSY=0;
3. Shut down SPI (SPE=0) and finally enter shutdown mode (or turn off the clock for that module).

### Unidirectional Receive-Only Mode in Master or Slave Mode (MSTR=1, BIDIMODE=0,

### RXONLY=1) or Bidirectional Receive Mode (MSTR=1, BIDIMODE=1, BIDIOE=0)

This situation needs to be handled specifically to ensure that SPI does not start a new transmission:

1. Wait for the penultimate (n-1st) RXNE=1;
2. Wait one SPI clock cycle (using a software delay) before shutting down SPI (SPE=0);
3. Wait for the last RXNE=1 before entering shutdown mode (or turning off the clock for that module).

*Notes: In unidirectional transmit-only mode (MSTR=1, BDM=1, BDOE=0) in master mode, the BSY flag is always low during transmission.*

### Receive-Only Mode in Slave Mode (MSTR=0, BIDIMODE=0, RXONLY=1) or Bidirectional

### (MSTR=0, BIDIMODE=1, BIDIOE=0)

1. SPI can be turned off at any time (SPE=0) and SPI will be turned off at the end of the current transmission;
2. If you wish to enter shutdown mode, you must first wait for BSY=0 before entering shutdown mode (or turning off the clock for that module).



## 26.3.9 SPI Communication using DMA

In order to achieve maximum communication speed, the SPI transmit buffer needs to be filled with data in a timely manner, and likewise the data in the receive buffer must be read away in a timely manner to prevent overflow. In order to facilitate high speed data transfer, SPI implements a DMA mechanism using a simple request/response.

The SPI module can issue a DMA transfer request when the corresponding enable bit on the SPI\_CR2 register is set. The transmit and receive buffers also have their own DMA requests (see ).

- When transmitting, a DMA request is issued each time TXE is set to '1', and the DMA controller then writes data to the SPI\_DR register, and the TXE flag is cleared as a result.
- On reception, a DMA request is issued each time RXNE is set to '1', and the DMA controller reads the data from the SPI\_DR register, and the RXNE flag is cleared as a result.

When only SPI is used to send data, only the SPI transmit DMA channel needs to be enabled. In this case, the OVR is set to '1' because no received data is read (translation: software does not have to bother with this flag).

When only SPI is used to receive data, only the receive DMA channel of SPI needs to be enabled. In transmit mode, after the DMA has transmitted all the data to be sent (the TCIF flag in the DMA\_ISR register changes to '1'), it is possible to confirm the end of the SPI communication by monitoring the BSY flag, which avoids corrupting the transmission of the last data when shutting down the SPI or going into stop mode. Therefore the software needs to wait for TXE=1 and then wait for BSY=0.

**Notes:** In discontinuous communication, there is a delay of 2 APB clock cycles between the operation of writing data to SPI\_DR and the BSY bit being set to '1', so it is necessary to wait for TXE=1 before waiting for BSY=0 after writing the last data.

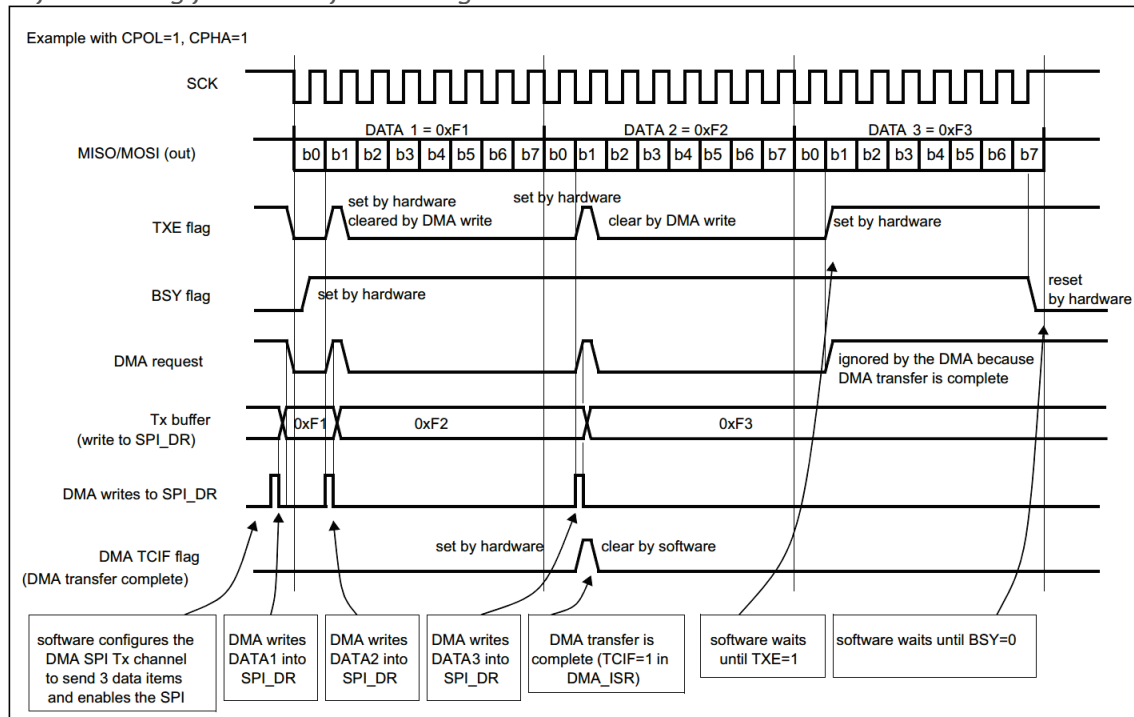


Figure219 Send using DMA

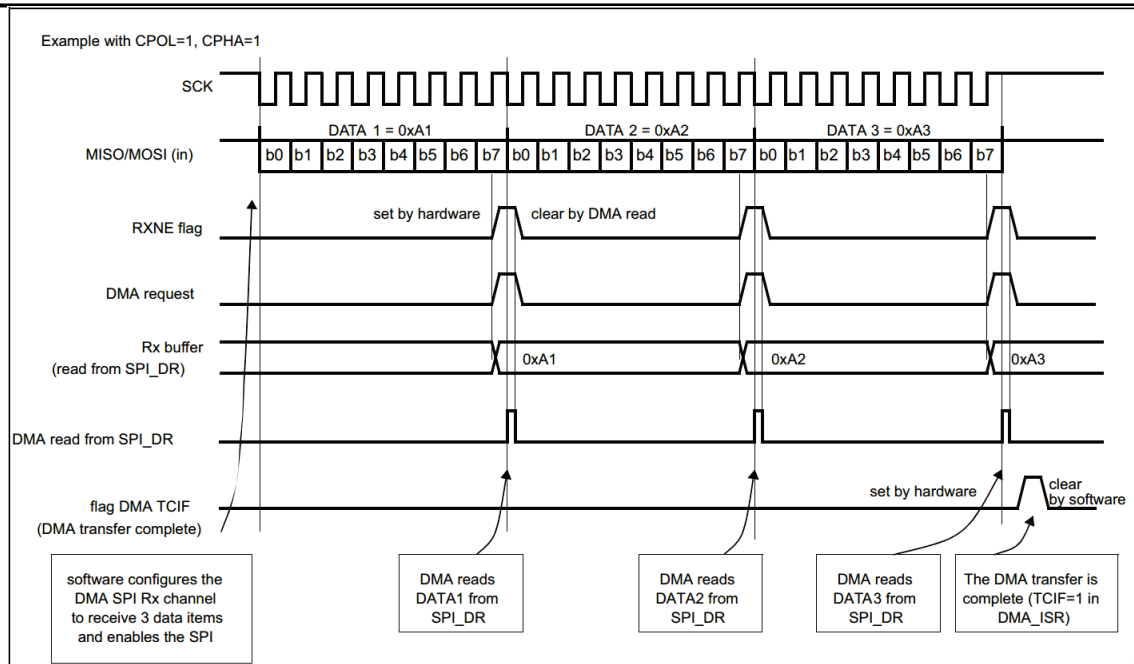


Figure220 Receiving with DMA

### DMA with CRC

When enabling SPI to use CRC checking and DMA mode is enabled, the sending and receiving of CRC bytes is done automatically at the end of communication.

At the end of the data and CRC transfer, a CRCERR flag of '1' in the SPI\_SR register indicates that an error occurred during the transfer.

## 26.3.10 Error Message

### Master Mode Failure Error (MODF)

A master mode failure occurs only when: the master device's NSS pin is pulled low under NSS pin hardware mode management; or when the SSI bit is set to '0' under NSS pin software mode management; 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 an SPI interrupt is generated if the ERRIE bit is set;
- The SPE bit is cleared to '0'. This stops all outputs and shuts down the SPI interface;
- The MSTR bit is cleared to '0', thus forcing this device into slave mode. The following steps are used to clear the MODF bit:
  1. When the MODF bit is set to '1', a read or write operation to the SPI\_SR register is performed;
  2. Then write the SPI\_CR1 register.

In systems with multiple MCUs, to avoid conflicts with multiple slave devices, the NSS pin of that master device must be pulled high before zeroing the MODF bit. After zeroing is completed, the SPE and MSTR bits can be restored to their original state.

For security reasons, the hardware does not allow the SPE and MSTR bits to be set when the MODF bit is '1'.

Normally configured, a slave device's MODF bit 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 program can perform a reset or return to the default state to recover from the error state.

### Overflow Error

An overflow error occurs when the master device has sent a data byte and the slave device has not cleared the RXNE generated by the previous data byte. When an overflow error is generated:

- The OVR bit is set to '1'; when the ERRIE bit is set, an interrupt is generated.

At this point, the data in the receiver buffer is not new data sent by the master device, and reading the SPI\_DR register returns previously unread data; all subsequently transmitted data is discarded.



---

The OVR can be cleared by reading the SPI\_DR register and SPI\_SR register in turn.

#### CRC error

The CRC error flag is used to check the validity of the received data when the CRCEN bit on the SPI\_CR register is set. If the value received in the shift register (the SPI\_TXCRCR value sent by the sender) does not match the value in the receiver's SPI\_RXCRCR register, the CRCERR flag on the SPI\_SR register is set to '1'.

### 26.3.11 SPI Interrupt

Table138 SPI Interrupt Requests

disruption event	event marker	Enable Control Bit
Send buffer empty flag	TXE	TXEIE
Receive buffer non-empty flag	RXNE	RXNEIE
Master Mode Failure Event	MODF	ERRIE
overflow error	OVR	
CRC error flag	CRCERR	

## 26.4 I2S Functional Description

### 26.4.1 I2S Functional Description

The block diagram of I2S is shown below:

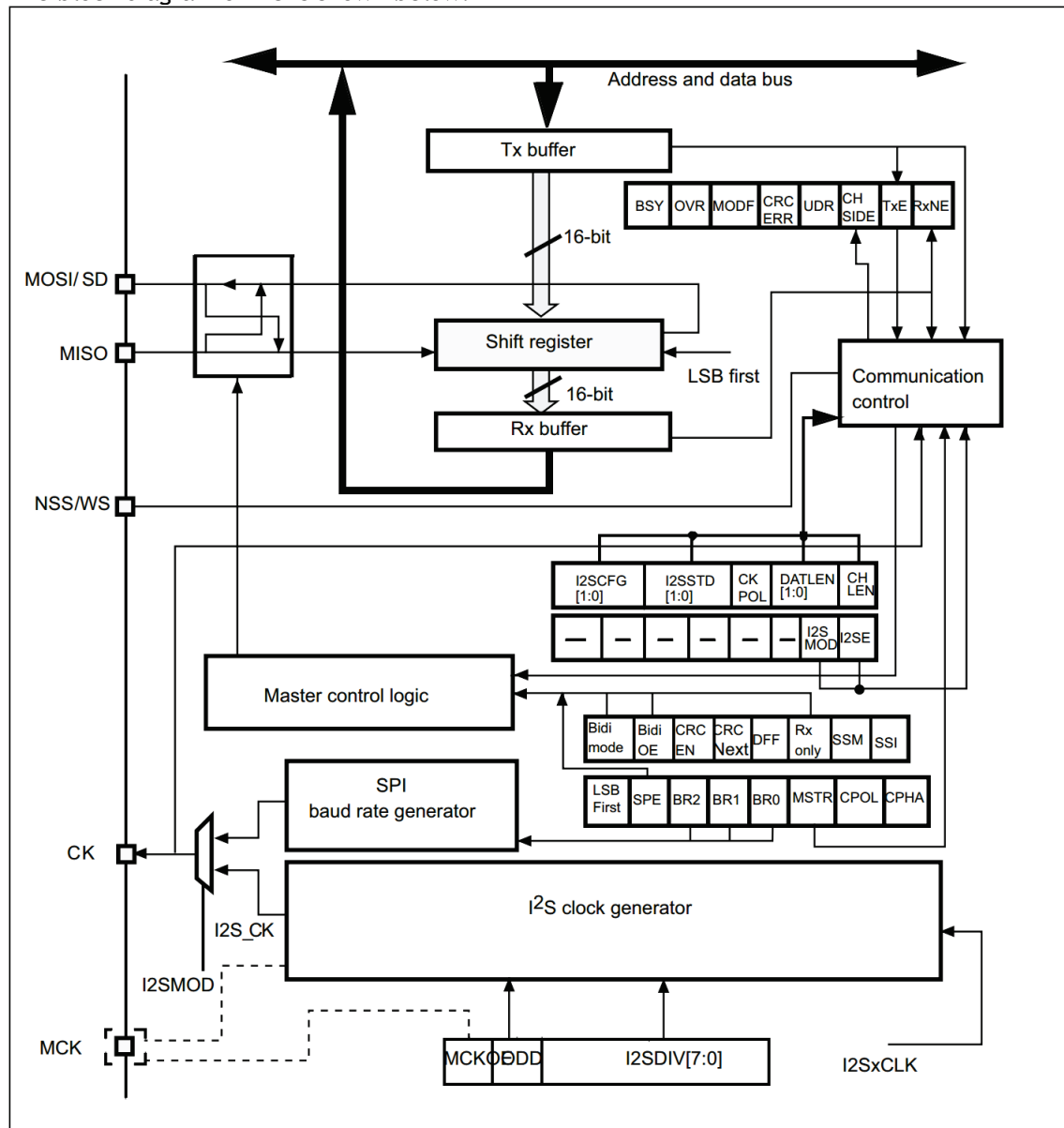


Figure 221 I2S Block Diagram

The I2S function can be enabled by setting the I2SMOD position of register SPI\_I2SCFGR to '1'. At this point, the SPI module can be used as an I2S audio interface. The I2S interface uses approximately the same pins, flags and interrupts as the SPI interface.

I2S and SPI share 3 pins:

- SD: Serial Data (mapped to MOSI pin), used to send and receive data from the 2 time division multiplexed channels;
- WS: Word Select (mapped to NSS pin), used as data control signal output in master mode and input in slave mode;
- CK: Serial clock (mapped to SCK pin), used as clock signal output in master mode and input in slave mode.

An additional pin can be used to output a clock when the master clock is required by some external audio device:

- MCK: Master Clock (independently mapped), used as an output additional clock signal pin when I2S is configured in master mode and the MCKOE bit of register SPI\_I2SPR is '1'. The frequency of the output clock signal is pre-set to  $256 \times F_s$ , where  $F_s$  is the sampling frequency of the audio signal.

When set to master mode, the I2S uses its own clock generator to generate the clock signal for communication. This clock generator is also the clock source for the master clock output. There are 2 additional registers in I2S mode, one is the register SPI\_I2SPR related to the clock generator configuration, and the other is the I2S general configuration register SPI\_I2SCFGR (which allows you to set parameters such as the audio standard, slave/master modes, data format, packet framing, clock polarity, etc.).

Register SPI\_CR1 and all CRC registers are not used in I2S mode. Similarly, the SSOE bit of register SPI\_CR2, and the MODF bit and CRCERR bit of register SPI\_SR are not used in I2S mode. I2S uses the same register SPI\_DR as SPI for 16-bit wide mode data transfer.

## 26.4.2 Supported Audio Protocols

The 3-wire bus supports time-division multiplexing of audio data on 2 channels: left and right, but only one 16-bit register is used for transmission or reception. Therefore, software must write data to the data register according to the channel currently being transmitted; similarly, when reading register data, the CHSIDE bit of SPI\_SR is examined to determine which channel the received data belongs to. The left channel always sends data before the right channel (the CHSIDE bit is meaningless under the PCM protocol).

There are four available data and packet frame combinations. Data can be sent in the following four data formats:

- 16-bit data packed into a 16-bit frame
- 16-bit data packed into 32-bit frames
- 24-bit data packed into 32-bit frames
- 32-bit data packed into 32-bit frames

When using 16-bit data to extend 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 does not require software intervention and there is no need to have a DMA request (only one read/write operation is required).

24-bit and 32-bit data frames require 2 CPU read or write operations to register SPI\_DR, and when using DMA, 2

DMA Transmission. For 24-bit data, the lowest 8 bits are set to 0 by hardware after expansion to 32 bits. For all data formats and communication standards, the highest bit (MSB) is always sent first.

The I2S interface supports four audio standards, which can be selected by setting the I2SSTD[1:0] bits and PCMSYNC bits of register SPI\_I2SCFGR.

### I2S Philips Standard

In this standard, pin WS is used to indicate which channel the data being sent belongs to. This pin is active 1 clock cycle before the first bit of data (MSB) is sent.

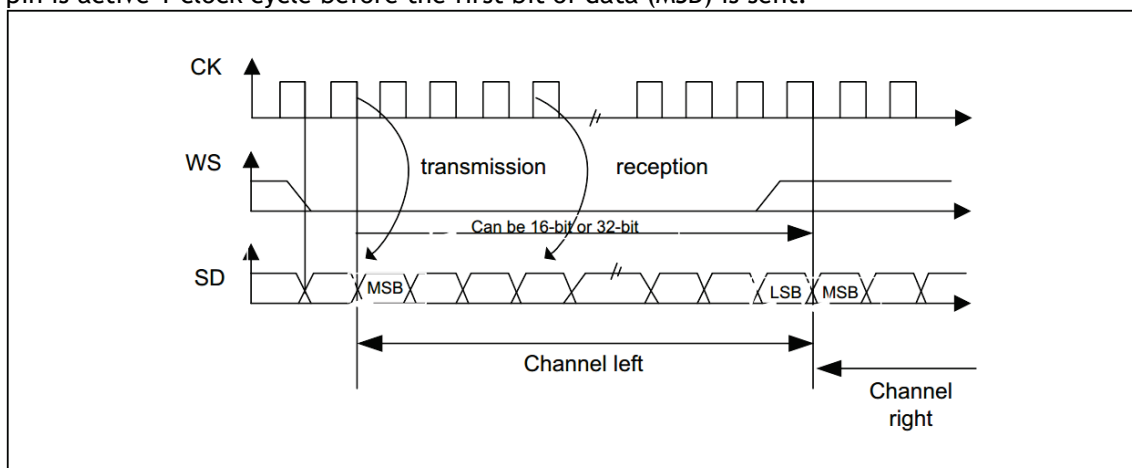


Figure 222 I2S Philips protocol waveforms (16/32 bit full precision, CPOL=0)

The sender changes the data on the falling edge of the clock signal (CK) and the receiver reads the data on the rising edge. The WS signal also changes on the falling edge of the clock signal.

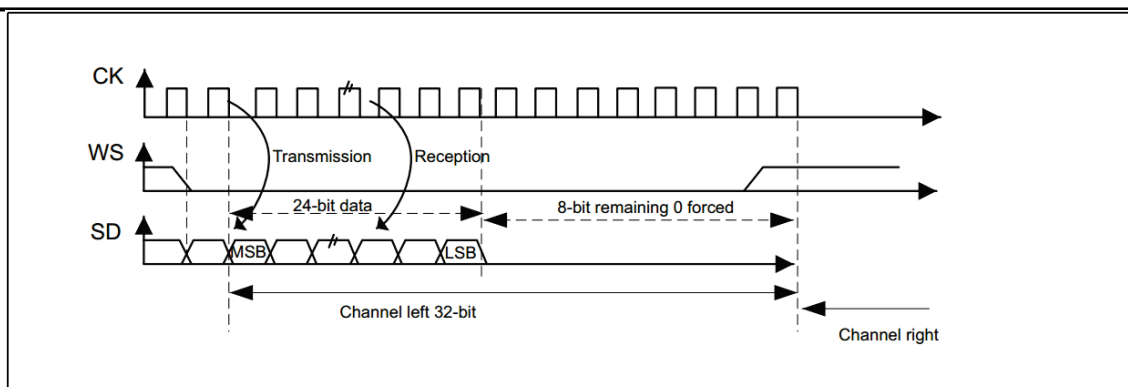


Figure223 I2S Philips Protocol Standard Waveform (24-bit frame, CPOL=0)

This mode requires 2 read or write operations to register SPI\_DR.

- In transmit mode: if 0x8EAA33 (24 bits) needs to be sent:

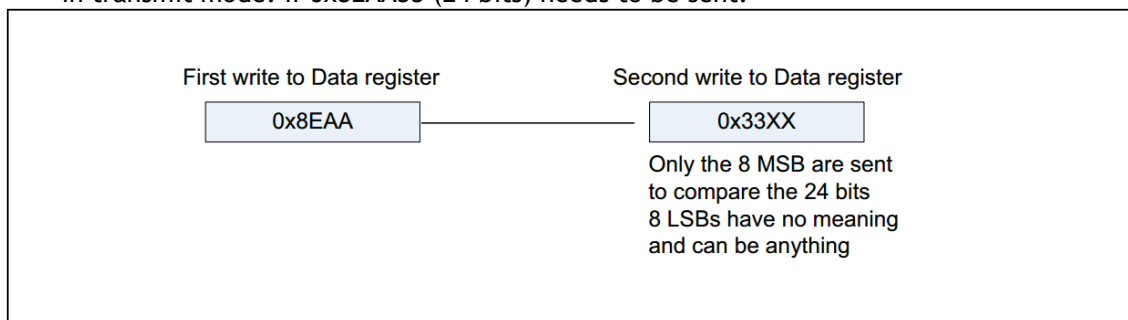


Figure224 sends 0x8EAA33

- In receive mode: if 0x8EAA33 is received:

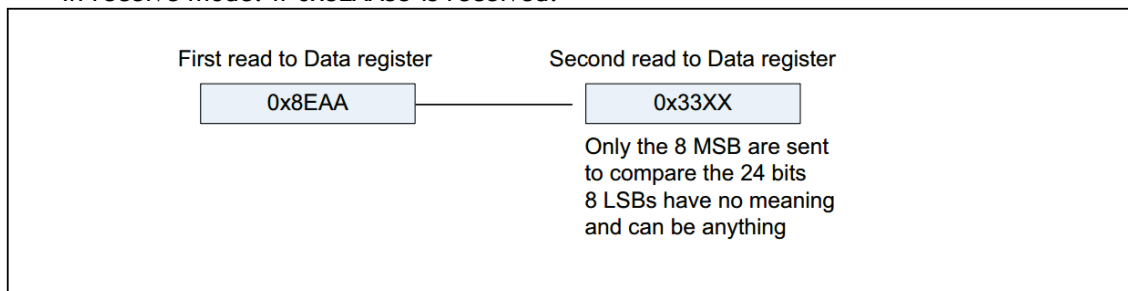


Figure225 Receive 0x8EAA33

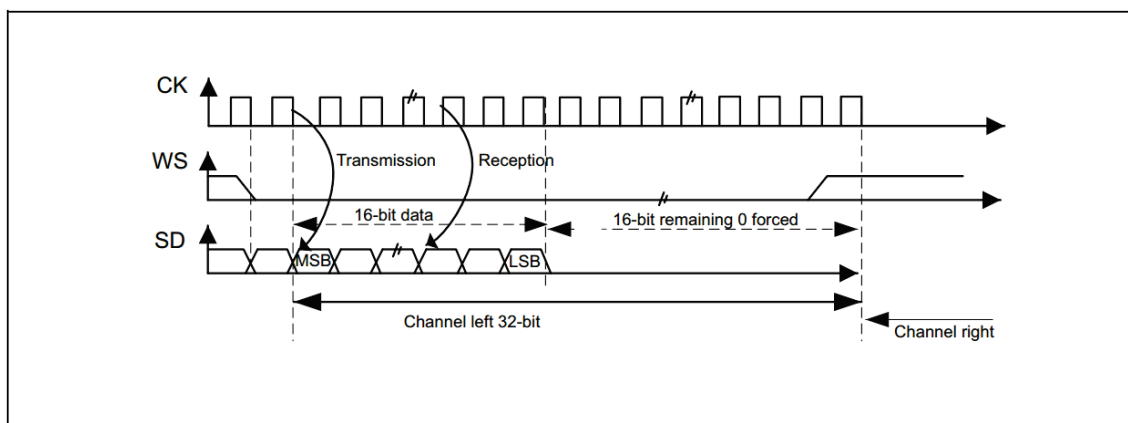


Figure226 I2S Philips Protocol Standard Waveforms (16-bit extended to 32-bit packet frame, CPOL=0)  
During the I2S configuration phase, if you choose to extend 16-bit data to a 32-bit voice channel frame, you only need to access register SPI\_DR once. the lower 16 bits used to extend to 32 bits are set to 0x0000 by hardware.  
If the data to be transmitted or received is 0x76A3 (extended to 32 bits is 0x76A3 0000), the required operation is shown below.

Only one access to SPIx\_DR

0x76A3

Figure227 Example

The MSB needs to be written to register SPI\_DR when transmitting; the flag bit TXE is '1' to indicate that new data can be written and an interrupt can be generated if the corresponding interrupt is allowed. The sending is done by hardware, even if the last 16 bits of 0x0000 have not been sent yet, TXE is set and the corresponding interrupt is generated.

When receiving, the flag bit RXNE is set to '1' each time the high 16-bit halfword (MSB) is received, and an interrupt can be generated if the corresponding interrupt is allowed.

This gives more time between 2 reads and writes and prevents underflow or overflow.

### MSB Alignment Standard

In this standard, the WS signal is generated at the same time as the first data bit, the highest bit (MSB).

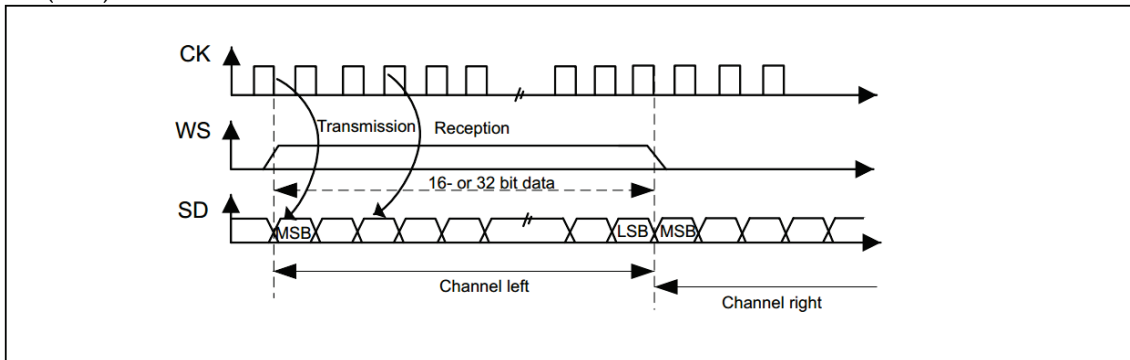


Figure228 MSB aligned 16-bit or 32-bit full-precision, CPOL=0

The sender changes the data on the falling edge of the clock signal; the receiver reads the data on the rising edge.

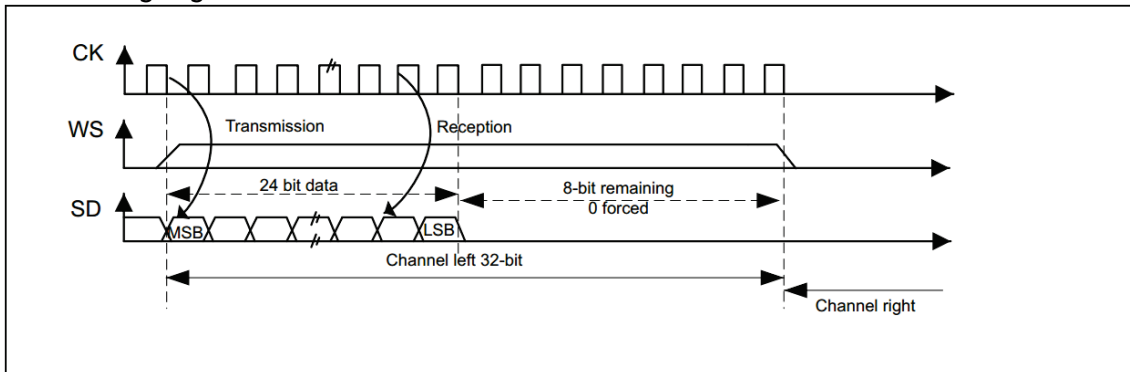


Figure229 MSB aligned 24-bit data, CPOL=0

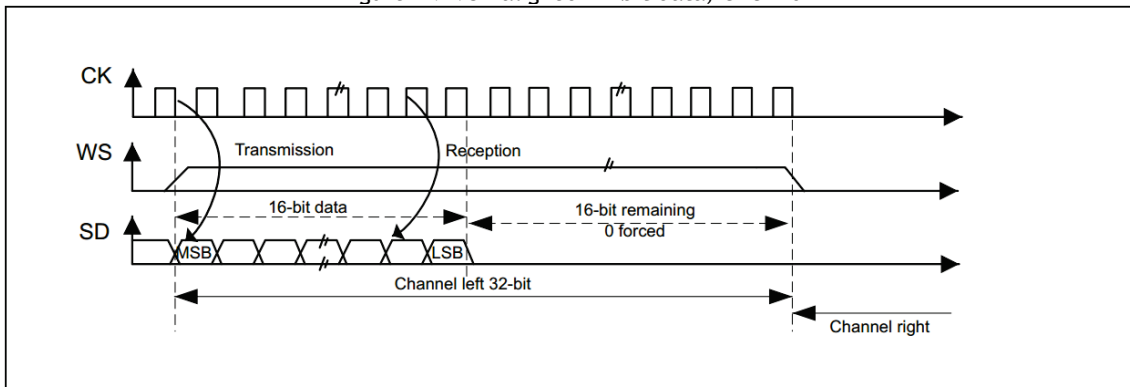


Figure230 MSB aligned 16-bit data extended to 32-bit packet frame with CPOL=0

## LSB Alignment Standard

This standard is similar to the MSB alignment standard (no difference in 16-bit or 32-bit full-precision frame format).

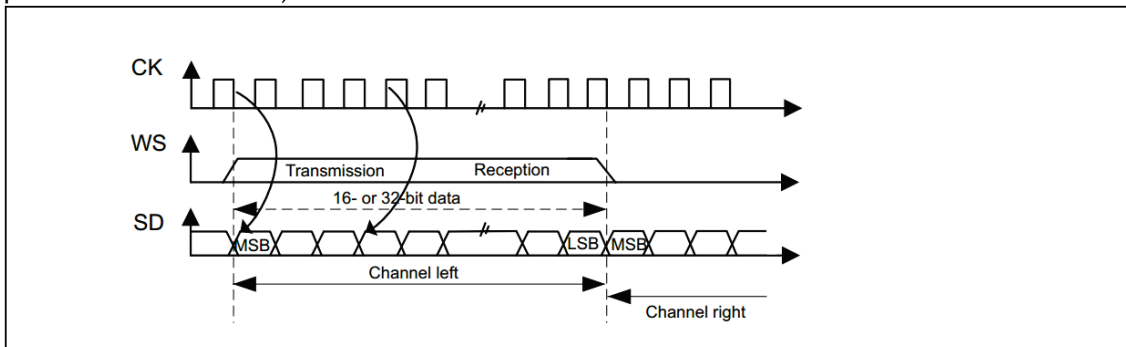


Figure231 LSB aligned 16-bit or 32-bit full precision, CPOL=0

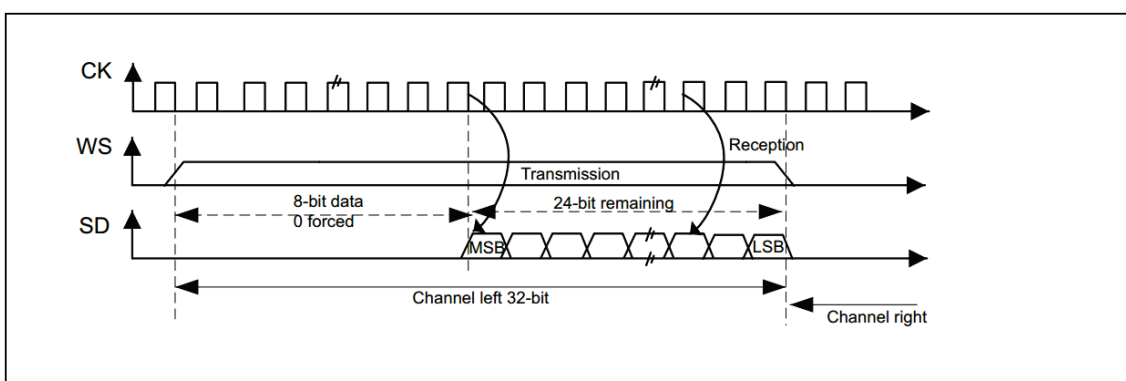


Figure232 LSB aligned 24-bit data, CPOL=0

### ● In Transmit Mode

To send the data 0x3478AE, 2 write operations to register SPI\_DR are required by software or DMA. The operation flow is shown below.

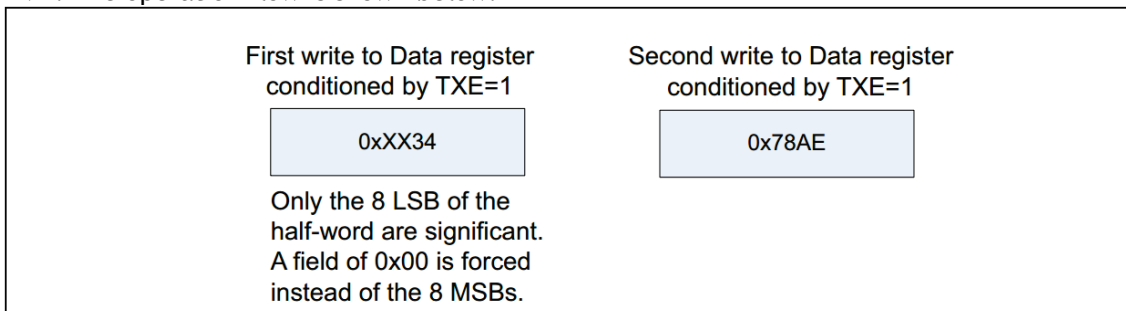


Figure233 Request to send 0x3478AE operation

### ● In Receive Mode

To receive the data 0x3478AE, 1 read operation of register SPI\_DR is required on each of 2 consecutive RXNE events.

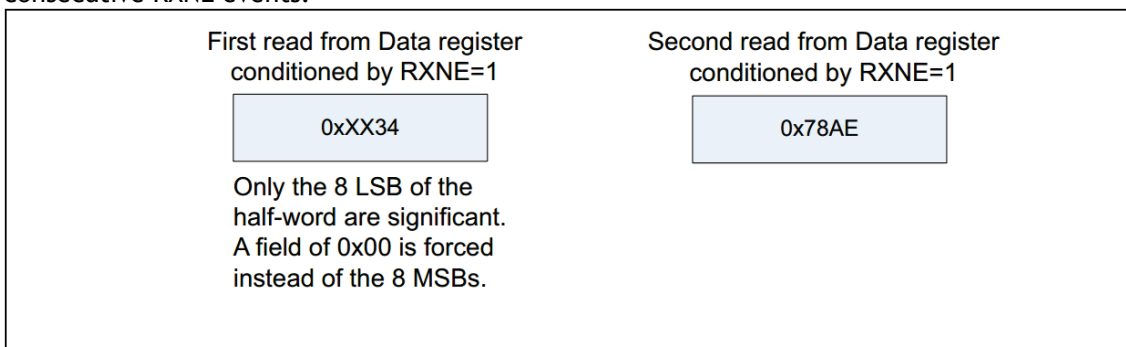


Figure234 Request to receive 0x3478AE operation

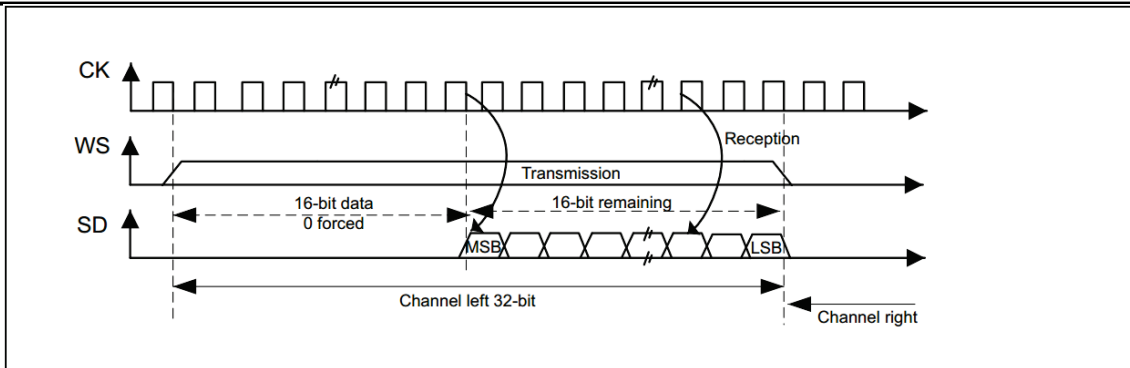


Figure235 LSB aligned 16-bit data extended to 32-bit packet frame with CPOL=0

During the I2S configuration phase, if you choose to extend the 16-bit data to a 32-channel frame, you only need to access register SPI\_DR once. at this point, the high halfword (16-bit MSB) after the extension to 32-bit is set to 0x0000 by the hardware.

If the data to be transmitted or received is 0x76A3 (extended to 32 bits is 0x0000 76A3), the required operation is shown below.

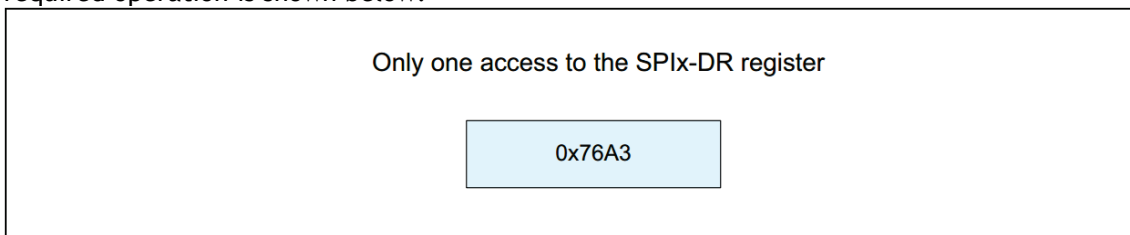


Figure236 Example

When sending, if TXE is '1', the user needs to write the data to be sent (i.e. 0x76A3). The 0x0000 portion used to extend to 32 bits is sent out first by the hardware, and the next TXE event occurs once valid data begins to be sent from the SD pin.

On receive, the RXNE event occurs as soon as valid data is received (instead of the 0x0000 part). This gives more time between 2 reads and writes and prevents underflow or overflow.

### PCM Standard

With the PCM standard, there is no channel selection information. The PCM standard has 2 available frame structures, short frame or long frame, which can be selected by setting the PCMSYNC bit of register SPI\_I2SCFGR.

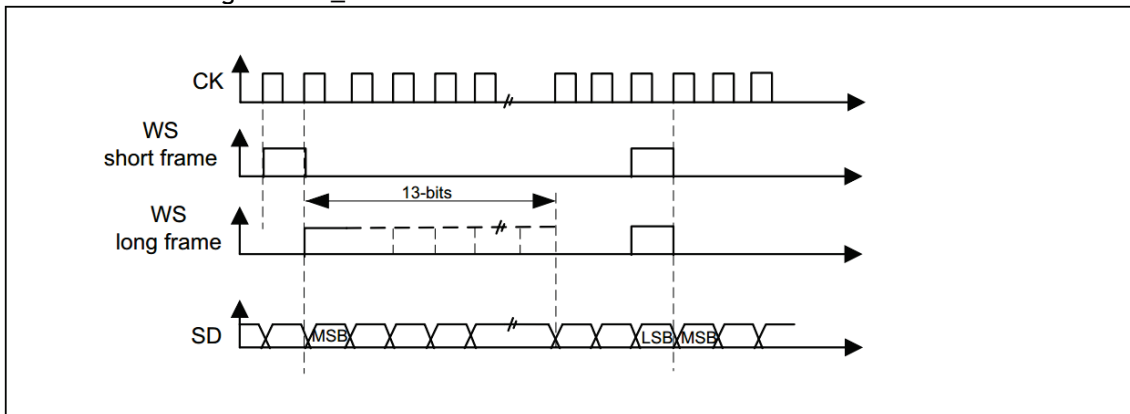


Figure237 PCM standard waveform (16-bit)

For long frames, the WS signal used to synchronize is valid for a fixed time of 13 bits in the master mode.

For short frames, the length of the WS signal used for synchronization is only 1 bit.

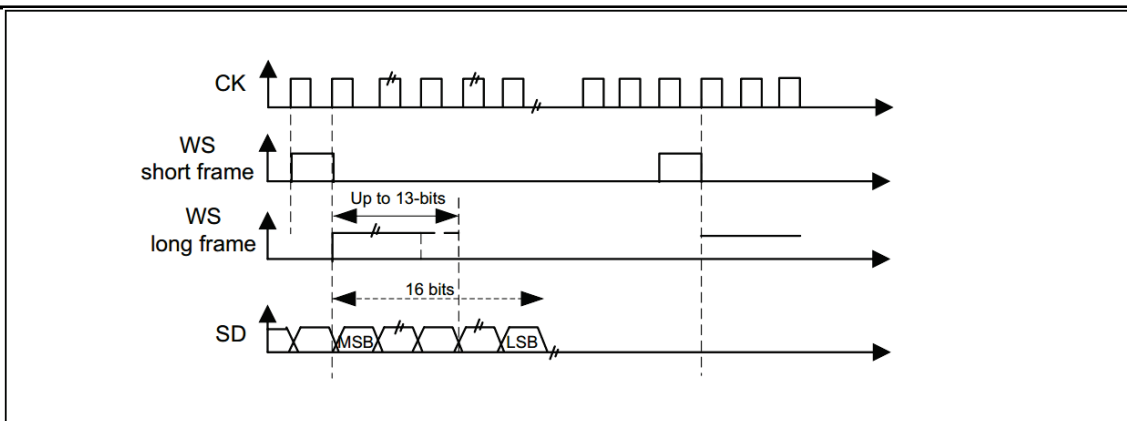


Figure238 PCM standard waveform (16-bit extended to 32-bit packet frame)

Notes: Regardless of the mode (master or slave) and the synchronization method (short or long frame), between 2 consecutive frames of data and between 2 synchronization signals  
The time difference, (even from slave mode) needs to be determined by setting the DATLEN bit and CHLEN bit of the SPI\_I2SCFGR register.

### 26.4.3 Clock Generator

The I2S bit rate determines the data stream on the I2S data lines and the frequency of the I2S clock signal. I2S bit rate= Number of bits per channel× Number of channels× Audio sampling frequency

For a signal with left and right channels and 16-bit audio, 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, we have: the I2S bit rate =  $32 \times 2 \times F_s$

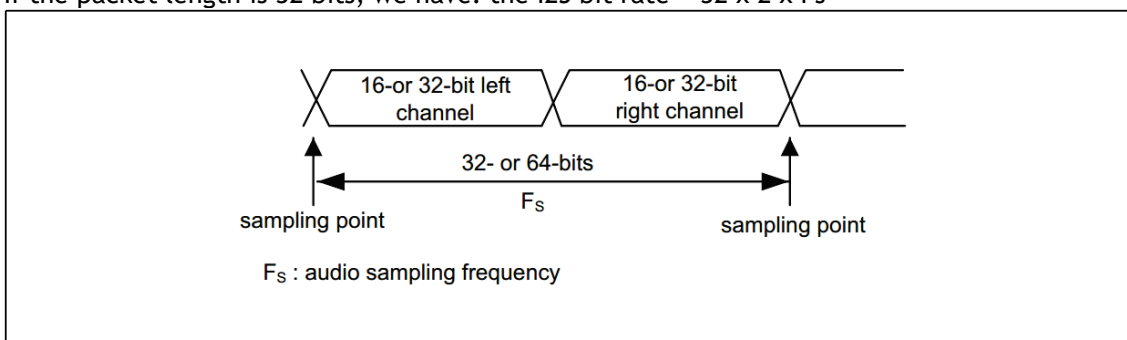


Figure239 Audio Sampling Frequency Definition

In master mode, the linear crossover needs to be set correctly in order to obtain the desired audio frequency.

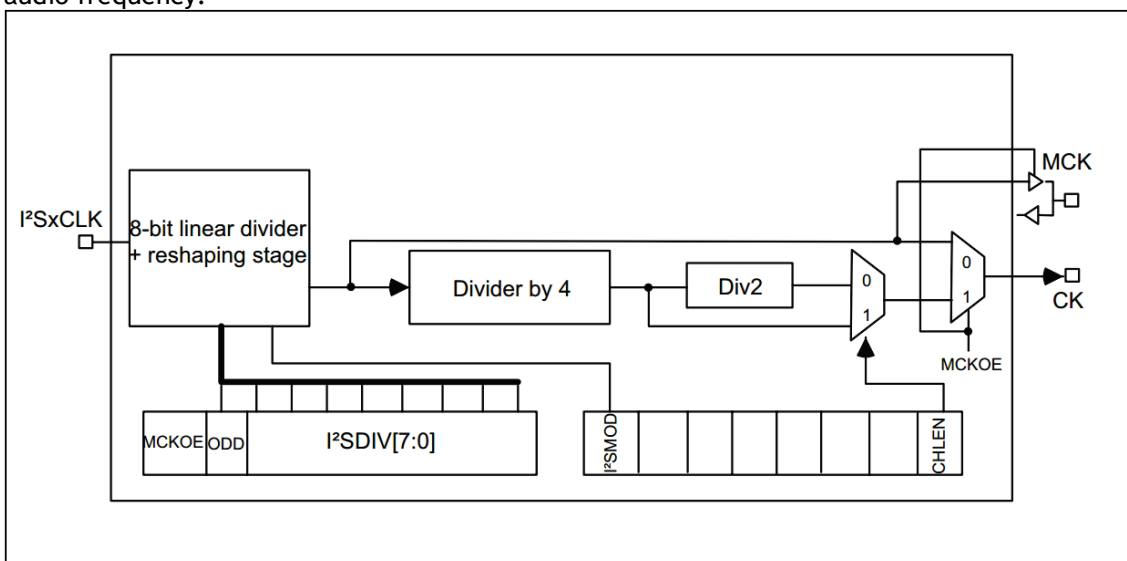


Figure240 I2S Clock Generator Architecture

1. x can be 2 or 3 in the diagram.



The clock source for I2SxCLK in the above figure is the system clock (i.e., the HSI, HSE, or PLL that drives the AHB clock).

The sampling frequency of the audio can be 96kHz, 48kHz, 44.1kHz, 32kHz, 22.05kHz, 16kHz, 11.025kHz, or 8kHz (or any value within this range). To obtain the desired frequency, the linear crossover needs to be set according to the following formula:

When master clock generation is required (MCKOE bit of register SPI\_I2SPR is '1'):

$F_s = I2SxCLK / [(16 \times 2) \times ((2 \times I2SDIV) + ODD) \times 8]$  for a 16-bit frame length for the voice channel

When the frame length of the sound 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'):

$F_s = I2SxCLK / [(16 \times 2) \times ((2 \times I2SDIV) + ODD)]$  for a 16-bit frame length for the voice channel

When the frame length of the channel is 32 bits,  $F_s = I2SxCLK / [(32 \times 2) \times ((2 \times I2SDIV) + ODD)]$  The following 2 tables give examples of the exact parameters for different clock configurations.

Notes: Other configurations can be used to achieve optimized clock accuracy.

Table139 Getting Accurate Audio Frequencies with a Standard 8MHz HSE Clock

SYSCLK (MHz)	12S_DIV		12S_ODD		MCLK	Desired value FS (Hz)	Actual FS (Hz)		inaccuracies	
	16-bit		32-bit				16-bit	32-bit	16-bit	32-bit
72	11	6	1	0	not have	96000	97826.09	93750	1.90%	2.34%
72	23	11	1	1	not have	48000	47872.34	48913.04	0.27%	1.90%
72	25	13	1	0	not have	44100	44117.65	43269.23	0.04%	1.88%
72	35	17	0	1	not have	32000	32142.86	32142.86	0.44%	0.44%
72	51	25	0	1	not have	22050	22058.82	22058.82	0.04%	0.04%
72	70	35	1	0	not have	16000	15675.75	16071.43	0.27%	0.45%
72	102	51	0	0	not have	11025	11029.41	11029.41	0.04%	0.04%
72	140	70	1	1	not have	8000	8007.11	7978.72	0.09%	0.27%
72	2	2	0	0	there are	96000	70312.15	70312.15	26.76 %	26.76%
72	3	3	0	0	there are	48000	46875	46875	2.34%	2.34%
72	3	3	0	0	there are	44100	46875	46875	6.29%	6.29%
72	9	9	0	0	there are	32000	31250	31250	2.34%	2.34%
72	6	6	1	1	there are	22050	21634.61	21634.61	1.88%	1.88%
72	9	9	0	0	there are	16000	15625	15625	2.34%	2.34%
72	13	13	0	0	there are	11025	10817.3	10817.3	1.88%	1.88%
72	17	17	1	1	there are	8000	8035.71	8035.71	0.45%	0.45%

## 26.4.4 I2S Master Mode

Set the I2S to operate in master mode, with the serial clock output from pin CK and the word select signal generated from pin WS. You can choose to output or not output the master clock (MCK) by setting the MCKOE bit of register SPI\_I2SPR.

### Workflows

- Setting I2SDIV[7:0] of register SPI\_I2SPR defines the serial clock baud rate that matches the audio sampling frequency. Also define the ODD bit of register SPI\_I2SPR.
- Setting the CKPOL bit defines the level state of the clock for communication when idle. If the master clock MCK needs to be provided to an external DAC/ADC audio device, set the MCKOE position of register SPI\_I2SPR to '1'. (Calculate the values of I2SDIV and ODD according to the different MCK output states, see section26.4.3 for details).
- Setting the I2SMOD bit of register SPI\_I2SCFGR to '1' activates the I2S function, setting the I2SSTD[1:0] and PCMSYNC bits selects the I2S standard used, and setting CHLEN selects the number of data bits per channel. Also set I2SCFG[1:0] of register SPI\_I2SCFGR to select the I2S master mode and direction (transmitter or receiver).

4. If required, the desired interrupt function and DMA function can be turned on by setting register SPI\_CR2.
5. The I2SE location of register SPI\_I2SCFGR must be set to '1'.
6. Pins WS and CK need to be configured for output mode. If the MCKOE bit of register SPI\_I2SPR is '1', pin MCK also needs to be configured for output mode.

### Send Process

When 1 half word (16 bits) of data is written to the transmit buffer, the transmit process starts. Assume that the first data written to the transmit buffer corresponds to the left channel data. When the data is moved from the transmit buffer to the shift register, the flag bit TXE is set to '1', which is when the data corresponding to the right channel is to be written to the transmit buffer. Flag bit CHSIDE indicates which channel corresponds to the data currently to be transmitted. The value of the flag bit CHSIDE is updated when TXE is '1', so it is meaningful when TXE is '1'. A complete data frame is not considered until all the data for the left channel and then the right channel have been transmitted. It is not possible to transmit only part of the data frame, e.g. only the data of the left channel.

While the first bit of data is sent out, the half-word data is transferred in parallel to the 16-bit shift register, and then the following bits are sent out from pin MOSI/SD in the order of high bit first. The flag bit TXE is set to '1' each time data is shifted from the transmit buffer to the shift register, and an interrupt is generated if the TXEIE bit of register SPI\_CR2 is '1'.

The operation of writing data depends on the selected I2S standard, see section 26.4.2 for details.

To ensure continuous audio data transfer, it is recommended to write the next data to be transferred to register SPI\_DR before the current transfer is completed.

It is recommended to wait for the flag bit TXE=1 and BSY=0 before clearing the I2SE bit to '0' when you want to disable the I2S function.

### Receiving Process

The configuration steps for the receive process are the same as those for the transmit process (see "Transmit Process" above), except for point 3, where you need to select the primary receive mode by configuring I2SCFG[1:0].

Audio data is always received in 16-bit packets regardless of data and channel length. That is, each time the receive buffer is filled, the flag bit RXNE is set to '1' and an interrupt is generated if the RXNEIE bit of register SPI\_CR2 is '1'. Depending on the configured data and the channel length, receiving data for the left or right channel will require either 1 or 2 passes of the data into the receive buffer.

A read operation of register SPI\_DR clears the RXNE flag bit.

CHSIDE is updated after each reception and its value depends on the WS signal generated by the I2S unit. The operation of reading data depends on the selected I2S standard, see section 26.4.2 for details.

If the previous received data has not been read and new data is received, i.e., an overflow occurs, the flag bit OVR is set to '1', and if the ERRIE bit of register SPI\_CR2 is '1', an interrupt is generated indicating that an error has occurred.

To disable the I2S function, a special operation needs to be performed to ensure that the I2S module can complete the transmission cycle normally without starting a new data transmission. The operation procedure is related to the data configuration and channel length, as well as the mode of the audio protocol:

- 16-bit data extended to 32-bit channel length (DATLEN=00 and CHLEN=1) using LSB (low bit) alignment mode (I2SSTD=10)
  1. Wait for the penultimate (n-1) RXNE = 1;
  2. Wait 17 I2S clock cycles (using software delay);
  3. Turn off I2S (I2SE=0).
- 16-bit data extended to 32-bit channel length (DATLEN=00 and CHLEN=1) using MSB (high bit) alignment, I2S, or PCM mode (I2SSTD=00, I2SSTD=01, or I2SSTD=11, respectively)
  1. Wait for the last RXNE=1;
  2. Wait 1 I2S clock cycle (using software delay);
  3. Turn off I<sup>(2)</sup>S (I2SE=0).
- All other combinations of DATLEN and CHLEN, any audio mode selected by I2SSTD, turn off I2S using the following:
  1. Wait for the penultimate (n-1) RXNE = 1;
  2. Wait one I2S clock cycle (using software delay);

3. Turn off I2S (I2SE=0).

Notes: The BSY flag is always low during transmission.

## 26.4.5 I2S Slave Mode

In slave mode, the I2S can be set to transmit and receive modes. Slave mode is configured in a way that basically follows the same process as configuring the master mode. In slave mode, no clock is required from the I2S interface. Both the clock signal and the WS signal are provided by the external master I2S device, connected to the appropriate pins. Therefore the user does not need to configure the clock.

The configuration steps are listed below:

1. Setting the I2SMOD bits of register SPI\_I2SCFGR activates the I2S function; setting I2SSTD[1:0] selects the I2S standard to be used; setting DATLEN[1:0] selects the number of bits for the data; and setting CHLEN selects the number of data bits for each channel. Setting I2SCFG[1:0] of register SPI\_I2SCFGR selects the direction of data (transmitter or receiver) for the I2S slave mode.
2. Set register SPI\_CR2 to turn on the desired interrupt function and DMA function as needed.
3. The I2SE bit of register SPI\_I2SCFGR must be set to '1'.

### Send Process

The transmit process starts when the external master device sends a clock signal and when the NSS\_WS signal requests data transmission. The slave device must be enabled and the I2S data register must be written before the external master device can start communication.

For MSB-aligned and LSB-aligned modes of I2S, the first data item written to the data register corresponds to the data of the left channel. When communication is started, data is transferred from the transmit buffer to the shift register, and then the flag bit TXE is set to '1'; at this point, the data item corresponding to the right channel is to be written to the I2S data register.

The flag bit CHSIDE indicates which channel corresponds to the data currently to be transmitted. In contrast to the sending process in master mode, in slave mode CHSIDE depends on the WS signal from the external master I2S. This means that the first data to be sent is prepared before the clock signal is received from the master. This means that the first data to be sent is prepared by the slave I2S before it receives the clock signal generated by the master. A WS signal of '1' indicates that the left channel is sent first.

Notes: Setting the I2SE bit to '1' should be done at least 2 PCLK clock cycles before the main I2S clock signal on the CK pin.

When the first bit of data is sent, the half-word data is transmitted in parallel over the I2S internal bus to the 16-bit shift register, and then the other bits are sent out from pin MOSI/SD in the order of high bit first. The flag bit TXE is set to '1' each time data is transferred from the transmit buffer to the shift register, and an interrupt is generated if the TXEIE bit of register SPI\_CR2 is '1'.

Note that the flag bit TXE is recognized as '1' before writing data to the transmit buffer. The operation of writing data depends on the selected I2S standard, see section 26.4.2 for details.

To ensure continuous audio data transmission, it is recommended that the next data to be transmitted be written to register SPI\_DR before the current transmission is completed. If new data is still not written to register SPI\_DR before the first clock edge representing the next data transfer is reached, the underflow flag bit is set to '1' and an interrupt may be generated; it indicates that the software sent the data incorrectly. If the ERRIE bit of register SPI\_CR2 is '1', an interrupt is generated when the flag bit UDR of register SPI\_SR is high. It is recommended that I2S be turned off at this point, and then the data be resumed from the left channel.

It is recommended that you wait for TXE=1 and BSY=0 before clearing the I2SE bit to close I2S.

### Receiving Process

The configuration steps are the same as the transmit process except for point 1. You need to select the main receive mode by configuring I2SCFG[1:0].

Regardless of the data and channel length, audio data is always received in 16-bit packets, i.e., each time the receive buffer is filled, the flag bit RXNE is set to '1', and an interrupt is generated if the RXNEIE bit of register SPI\_CR2 is '1'. Depending on the data and channel length settings, receiving left or right channel data will require 1 or 2 transfers of data to the receive buffer. CHSIDE is updated every 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 will clear the RXNE bit.

The operation of reading data depends on the selected I<sup>2</sup>S standard, see section 26.4.2 for details.

When the previous received data has not been read and new data is received, an overflow is generated and the flag bit OVR is set is '1'; if the ERRIE bit of register SPI\_CR2 is '1', an interrupt is generated indicating that an error has occurred. To disable the I<sup>2</sup>S function, the I2SE bit needs to be cleared '0' when the last RXNE=1 is received.

*Notes: The external master I2S device needs to have the ability to send/receive 16-bit or 32-bit packets over the audio channel.*

## 26.4.6 Status Flag

There are 3 status flag bits for the user to monitor the status of the I2S bus.

### Busy Flag Bit (BSY)

The BSY flag is set and cleared by hardware (writing this bit has no effect), and this flag bit indicates the status of the I2S communication layer.

A '1' in this bit indicates that I<sup>2</sup>S communication is in progress, with one exception: in the main receive mode (I2SCFG=11), the BSY flag is always low during reception.

Before the software wants to shut down the SPI module, the BSY flag can be used to detect the end of the transmission, which prevents corrupting the last transmission, so the following procedure needs to be strictly followed.

The BSY flag is set to '1' when transmission starts, unless the I<sup>2</sup>S module is in the master receive mode. This flag bit is cleared in the following cases:

- When the transmission ends (except for the main transmit mode, in which communication is continuous);
- When the I<sup>2</sup>S module is turned off.

When communication is continuous:

- When in master transmit mode, the BSY flag is always high during the entire transmission;
- In slave mode, the BSY flag goes low for 1 I<sup>2</sup>S clock cycle between each data item transmission.

*Notes: Do not use the BSY flag to handle the sending and receiving of each data item; it is better to use the TXE and RXNE flags.*

### Transmit Buffer Empty Flag Bit (TXE)

A '1' in this flag bit indicates that the transmit buffer is empty and new data to be sent can be written to the transmit buffer. The flag bit clears '0' when there is already data in the transmit buffer. This flag bit is also '0' when I<sup>2</sup>S is turned off (I2SE bit is '0').

### Receive Cache Non-Empty Flag Bit (RXNE)

This flag position '1' indicates that there is received valid data in the receive buffer. This bit is cleared '0' when the register SPI\_DR is read.

### Channel Flag Bit (CHSIDE)

In transmit mode, this flag bit is refreshed when TXE is high, indicating the channel on which data is being sent from the SD pin. If an underflow error occurs in the transmit from mode, the value of this flag bit is invalid and I<sup>2</sup>S needs to be turned off and on again before communication can be restarted.

In receive mode, this flag bit is refreshed when data is received in register SPI\_DR, indicating the channel on which the received data is located. Note that if an error occurs (e.g., overflow OVR), this flag bit is meaningless and the I<sup>2</sup>S needs to be turned off and on again (also, if necessary, modify the I2S configuration).

Under the PCM standard, this flag bit has no meaning in either the short frame format or the long frame format.

If the flag bit OVR or UDR of register SPI\_SR is '1' and the ERRIE bit of register SPI\_CR2 is '1', an interrupt is generated. (After the interrupt source has been cleared) the interrupt flag can be cleared by reading register SPI\_SR.

## 26.4.7 Error Flag

The I<sup>2</sup>S cell has 2 error flag bits.

### Underflow Flag Bit (UDR)

In slave-transmit mode, this flag bit is set to '1' if new data is still not written to the SPI\_DR register when the first clock edge of the data transfer is reached. This flag bit is valid after the I2SMOD position '1' of register SPI\_I2SCFGR. If the ERRIE bit of register SPI\_CR2 is '1', an interrupt is generated.

This flag bit is cleared by a read operation of register SPI\_SR.

### Overflow Flag Bit (OVR)

If new data is received when the previous received data has not been read, an overflow is generated, this flag position '1', and if the ERRIE bit of register SPI\_CR2 is '1', an interrupt is generated indicating that an error has occurred.

At this point, the contents of the receive cache are not flushed to new data sent from the transmitting device. A read operation of register SPI\_DR returns the last correctly received data. All other 16-bit data sent by the transmitting device after an overflow has occurred is lost. This flag bit is cleared by reading register SPI\_SR followed by register SPI\_DR.

## 26.4.8 I<sup>2</sup>S Interrupt

The following table lists all I2S interrupts

Table140 I<sup>2</sup>S Interrupt Requests

disruption event	event marker	Enable Flag Bit
Send Buffer Empty Flag Bit	TXE	TXEIE
Receive buffer non-empty flag bit	RXNE	RXNEIE
underflow flag	OVR	ERRIE
overflow flag	UDR	

## 26.4.9 DMA Function

DMA operates in I2S mode in exactly the same way as in SPI mode except that the CRC function is not available. This is because there is no data transfer protection system in I2S mode.

## 26.5 SPI and I<sup>2</sup>S Register Descriptions

Abbreviations used in register descriptions can be found in Section1 .

These peripheral registers can be operated in half-word (16-bit) or word (32-bit) format.

### 26.5.1 SPI Control Register 1 (SPI\_CR1) (not used in I<sup>2</sup>S mode)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BIDI MODE	BIDIOE	CRCEN	CRCNE XT	DFF	RXO NLY	SSM	SSI	LSB FIRST	SPE	BR [2:0]			MSTR	CPOL	CPHA
rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW

Bit	notation	clarification
15	BIDIMODE	<b>BIDIMODE:</b> Bidirectional data mode enable 0: Selects "Two-Wire Bidirectional" mode; 1: Select the "Single Line Bidirectional" mode. Note: Not used in I2S mode.
14	BIDIOE	<b>BIDIOE:</b> Output enable in bidirectional mode, together with the BIDIMODE bit, determines the direction of data output in "single line bidirectional" mode. 0: Output disabled (receive-only mode); 1: Output enable (transmit-only mode). This "one-wire" data line is the MOSI pin on the master side and the MISO pin on the slave side. Note: Not used in I2S mode.
13	CRCEN	<b>CRCEN:</b> Hardware CRC calculation enable 0: CRC calculation is disabled; 1: Initiate CRC calculation.

		Note: This bit can only be written when SPI is disabled (SPE=0), otherwise an error occurs. This bit can only be used in full duplex mode. Note: Not used in I2S mode.
12	CRCNEXT	<b>CRCNEXT:</b> next transmit CRC (Transmit CRC next) 0: The next value sent comes from the send buffer. 1: The next value sent comes from the Send CRC register. Note: This bit should be set immediately after the last data is written to the SPI_DR register. Note: Not used in I2S mode.
11	DFF	<b>DFF:</b> Data frame format 0: Transmit/receive using 8-bit data frame format; 1: Transmit/receive using 16-bit data frame format. Note: This bit can only be written when SPI is disabled (SPE=0), otherwise an error occurs. Note: Not used in I2S mode.
10	RXONLY	<b>RXONLY:</b> Receive only This bit, together with the BIDIMODE bit, determines the direction of transmission in "two-wire bidirectional" mode. In a multiple slave configuration, the This bit is set to 1 on slave devices that are not being accessed, allowing only the accessed slave devices to have outputs, thus not causing data conflicts on the data lines. 0: full duplex (transmit and receive); 1: output disabled (receive-only mode). Note: Not used in I2S mode.
9	SSM	<b>SSM:</b> Software slave management (SSM) When SSM is set, the level on the NSS pin is determined by the value of the SSI bit. 0: Prohibit software from device management; 1: Enable the software to manage from the device. Note: Not used in I2S mode.
8	SSI	<b>SSI:</b> Internal slave select This bit is only significant when the SSM bit is '1'. It determines the level on the NSS and invalidates I/O operations on the NSS pin. Note: Not used in I2S mode.
7	LSBFIRST	<b>LSBFIRST:</b> Frame format 0: MSB is sent first; 1: Send the LSB first. Note: The value of this bit cannot be changed while communication is in progress. Note: Not used in I2S mode.
6	SPE	<b>SPE:</b> SPI enable 0: Disable SPI devices; 1: Turn on the SPI device. Note: Not used in I2S mode. Note: When turning off the SPI device, follow the procedure in Section 26.3.8.
5:3	BR [2:0]	<b>BR[2:0]:</b> Baud rate control 000: fPCLK/2      100: fPCLK/32      010: fPCLK/8      011: fPCLK/16 001: fPCLK/4      101: fPCLK/64      110: fPCLK/128      111: fPCLK/256 These bits cannot be modified while communication is in progress. Note: Not used in I2S mode.
2	MSTR	<b>MSTR:</b> Master selection 0: Configured as a slave device; 1: Configure as the master device. Note: This bit cannot be modified while communication is in progress. Note: Not used in I2S mode.
1	CPOL	<b>CPOL:</b> Clock polarity 0: SCK is held low during the idle state; 1: SCK is held high during the idle state. Note: This bit cannot be modified while communication is in progress. Note: Not used in I2S mode.
0	CPHA	<b>CPHA:</b> Clock phase 0: Data sampling starts from the first clock edge; 1: Data sampling starts from the second clock edge. Note: This bit cannot be modified while communication is in progress. Note: Not used in I2S mode.



## 26.5.2 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
Reserved								TXEIE	RXNEIE	ERRIE	Reserved	SSOE	TXDMA EN	RXDMA EN	
								rw	rw	rw	res	res	rw	rw	rw

Bit	notation	clarification
15:8	Reserved	Reserved bit, hardware forced to 0
7	TXEIE	<b>TXEIE</b> : Tx buffer empty interrupt enable 0: Disable TXE interrupt; 1: TXE interrupt is allowed and an interrupt request is generated when the TXE flag is set to '1'.
6	RXNEIE	<b>RXNEIE</b> : receive buffer not empty interrupt enable (Rx buffer not empty interrupt enable) 0: Disable RXNE interrupt; 1: RXNE interrupt is allowed and an interrupt request is generated when the RXNE flag is set.
5	ERRIR	<b>ERRIR</b> : Error interrupt enable When an error (CRCERR, OVR, MODF) is generated, this bit controls whether or not an interrupt is generated 0: Disable error interrupt; 1: Error interrupts are allowed.
4:3	Reserved	Reserved bit, hardware forced to 0.
2	SSOE	<b>SSOE</b> : SS output enable 0: Disable SS output in master mode, the device can work in multi-master device mode; 1: When the device is turned on, the SS output in master mode is turned on, and the device cannot work in multi-master device mode. Note: Not used in I2S mode.
1	TXDMAEN	<b>TXDMAEN</b> : Transmit buffer DMA enable (Tx buffer DMA enable) When this bit is set, the TXE flag sends out a DMA request as soon as the TXE flag is set 0: Disable sending buffer DMA; 1: Initiate transmit buffer DMA.
0	RXDMAEN	<b>RXDMAEN</b> : Receive buffer DMA enable (Rx buffer DMA enable) When this bit is set, the RXNE flag sends out a DMA request as soon as the RXNE flag is set. 0: Disable receive buffer DMA; 1: Initiate receive buffer DMA.

## 26.5.3 SPI Status Register (SPI\_SR)

Address offset: 0x08

Reset value: 0x0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								BSY	OVR	MODF	CRC ERR	UDR	CHSIDE	TXE	RXNE
								r	r	r	rc_w0	r	r	r	r

Bit	notation	clarification
15:8	Reserved	Reserved bit, hardware forced to 0
7	BSY	<b>BSY</b> : Busy flag 0: SPI is not busy; 1: SPI is busy communicating or the transmit buffer is not empty. This bit is set or reset by hardware. Note: Special care is required when using this flag, see Section26.3.7 and Section26.3.8 for details.
6	OVR	<b>OVR</b> : Overrun flag 0: No overflow error occurred; 1: An overflow error occurs. This bit is set by hardware and reset by a software sequence. For more information on software sequences, refer to the26.4.7 section.
5	MODF	<b>MODF</b> : Mode fault 0: No mode error occurred; 1: A mode error occurs. This bit is set by hardware and reset by a software sequence. For more information on software sequences, refer to the26.3.10 section. Note: Not used in I2S mode.
4	CRCERR	<b>CRCERR</b> : CRC error flag (CRC error flag) 0: The received CRC value matches the value in the SPI_RXCRCR register;

		1: The received CRC value does not match the value in the SPI_RXCRCR register. This bit is set by hardware and reset by writing '0' by software. Note: Not used in I2S mode.
3	UDR	<b>UDR:</b> Underrun flag bit 0: No underflow occurred; 1: Underflow occurs. This flag bit is set '1' by hardware and cleared '0' by a software sequence as detailed in section 26.4.7. Note: Not used in SPI mode.
2	CHSIDE	<b>CHSIDE:</b> Sound channel (Channel side) 0: Required to transmit or receive the left channel; 1: The right channel needs to be transmitted or received. Note: Not used in SPI mode. Not meaningful in PCM mode.
1	TXE	<b>TXE:</b> send buffer empty (Transmit buffer empty) 0: Send buffer is not empty; 1: Send buffer is empty.
0	RXNE	<b>RXNE:</b> Receive buffer not empty (Receive buffer not empty) 0: The receive buffer is empty; 1: The receive buffer is not empty.

## 26.5.4 SPI Data Register (SPI\_DR)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DR[15:0]															
rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW
Bit	notation	clarification													
15:0	DR[15:0]	<b>DR[15:0]:</b> Data register (Data register) to be sent or received data The data register corresponds to two buffers: one for writing (transmit buffer); the other for reading (receive buffer). A write operation writes data to the transmit buffer; a read operation returns data to the receive buffer. Note on SPI mode: Depending on the selection of the data frame format by the DFF bit of SPI_CR1, data can be sent and received in either 8-bit or 16-bit. For proper operation, the data frame format needs to be determined before enabling SPI. For 8-bit data, the buffer is 8-bit and only SPI_DR[7:0] will be used when sending and receiving. On receive, SPI_DR[15:8] is forced to 0. For 16-bit data, the buffer is 16-bit and the entire data register, SPI_DR[15:0], is used for sending and receiving.													

## 26.5.5 SPI CRC Polynomial Register (SPI\_CRCPR) (not used in I<sup>2</sup>S mode)

Address offset: 0x10 Reset value: 0x0007

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRCPOLY [15:0]															
rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW
Bit	notation	clarification													
15:0	CRCPOLY [15:0]	<b>CRCPOLY[15:0]:</b> CRC polynomial register This register contains the polynomials used in CRC calculations. Its Reset value is 0x0007, other values can be set depending on the application. Note: Not used in I2S mode.													

## 26.5.6 SPIRxCRC Register (SPI\_RXCRCR) (Not Used in I<sup>2</sup>S Mode)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RxCRC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
Bit	notation	clarification													
15:0	RXCRC[15:0]	<b>RXCRC[15:0]:</b> receive CRC registers When CRC calculation is enabled, RXCRC[15:0] contains the CRC value calculated based on the bytes received. This register is reset when a '1' is written to the CRCEN bit of SPI_CR1. the CRC calculation uses the polynomial in SPI_CRCPR. When the data frame format is set to 8-bit, only the lower 8 bits are involved in the calculation and follow the CRC8 method; when the data frame format is 16-bit, all													



		16 bits in the register are involved in the calculation and follow the CRC16 standard. Note: Reading this register when the BSY flag is '1' will likely read an incorrect value. Note: Not used in I2S mode.
--	--	---

## 26.5.7 SPITxCRC Register (SPI\_TXCRCR) (Not Used in I<sup>2</sup>S Mode)

Address offset: 0x18

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TxCRC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
Bit	notation		clarification												
15:0	TxCRC[15:0]		<p><b>TxCRC[15:0]:</b> transmit CRC registers</p> <p>When CRC calculation is enabled, TXCRC[15:0] contains the CRC value calculated based on the bytes to be sent. When CRC calculation is enabled in This register is reset when the CRCEN bit in SPI_CR1 is written to '1'. the 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 and follow the CRC8 method; when the data frame format is 16-bit, all 16 bits in the register are involved in the calculation and follow the CRC16 standard.</p> <p>Note: Reading this register when the BSY flag is '1' will likely read an incorrect value.</p> <p>Note: Not used in I2S mode.</p>												

## 26.5.8 SPI\_I<sup>2</sup>S Configuration Register (SPI\_I2SCFGR)

Address offset: 0x1C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				I2SMOD	I2SE	I2SCFG		PCM SYNC	Reserved	I2SSTD		CKPOL	DATLEN		CHLEN
rw				rw	rw	rw	rw	rw	res	rw	rw	rw	rw	rw	rw
Bit	notation				clarification										
15:12	Reserved				Reserved bit, hardware forced to 0										
11	I2SMOD				I2SMOD: I2S mode selection 0:Select SPI mode; 1:Select I2S mode. Note: This bit can only be set when SPI or I2S is turned off.										
10	I2SE				I2SE: I2S enable 0:Turn off I2S; 1:I2S Enable. Note: Not used in SPI mode.										
9:8	I2SCFG				I2SCFG: I2S configuration mode 00:Send from device; 01:Received from the device; 10:Master device sends; 11:Master device accepted. Note: This bit is only set when I2S is turned off. It is not used in SPI mode.										
7	PCMSYNC				PCMSYNC: PCM frame synchronization (PCM frame synchronization) 0:Short frame synchronization; 1:Long frame synchronization. Note:This bit is only significant when I2SSTD=11 (using PCM standard). It is not used in SPI mode.										
6	Reserved				Reserved bit, hardware forced to 0.										
5:4	I2SSTD				I2SSTD: I2S standard selection (I2S standard selection) 00:I2S Philips Standard; 01:High-byte alignment standard (left-aligned); 10:Low byte alignment standard (right justified); 11:PCM Standard. For details on the I2S standard, see section26.4.2 . Note: For correct operation, this bit can only be set when I2S is turned off. It is not used in SPI mode.										
3	CKPOL				CKPOL: Steady state clock polarity 0:I2S clock quiescent state is low; 1:I2S clock quiescent state is high. Note: For correct operation, this bit is only set when I2S is turned off. It is not used in SPI mode.										

2:1	DATLEN	<b>DATLEN:</b> Data length to be transferred (Data length to be transferred) 00:16-bit data length; 01:24-bit data length; 10:32-bit data length; 11:Not allowed. Note: For correct operation, this bit is only set when I2S is turned off. It is not used in SPI mode.
0	CHLEN	<b>CHLEN:</b> Channel length (number of bits per audio channel) 0:16 bits wide; 1:32 bit width. Write operations on this bit only make sense if DATLEN=00, otherwise the channel lengths are all fixed by hardware to 32 bits. Note: For correct operation, this bit is only set when I2S is turned off. It is not used in SPI mode.

## 26.5.9 SPI\_I<sup>2</sup>S 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
Reserved						MCKOE	ODD	I2SDIV							
						rW	rW	rW	rW	rW	rW	rW	rW	rW	rW

Bit	notation	instructions
15:10	Reserved	Reserved bit, hardware forced to 0
9	MCKOE	<b>MCKOE:</b> Master clock output enable 0:Turn off the master device clock output; 1:Master device clock output enable. Note: For proper operation, this bit can only be set when I2S is turned off. This bit is only used in I2S master mode. It is not used in SPI mode.
8	ODD	<b>ODD:</b> Odd factor for the prescaler 0:Actual crossover coefficient = I2SDIV*2; 1:Actual crossover factor = (I2SDIV*2)+1. See section26.4.3 . Note: For proper operation, this bit can only be set when I2S is turned off. This bit is only used in I2S master mode. It is not used in SPI mode.
7:0	I2SDIV	<b>I2SDIV:</b> I2S linear prescaler disable Setting I2SDIV[7:0]=0 or I2SDIV[7:0]=1 See section26.4.3 . Note: For proper operation, this bit can only be set when I2S is turned off. This bit is only used in I2S master mode. It is not used in SPI mode.

## 26.5.10 Register Address Map

Table141 List of SPI registers and their Reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0							
000h	SPI_CR1	Reserved																BIDIMODE	BIDIOE	CRCEN	CRCNEXT	DFE	RXOnly	SSM	SSI	LSBFIRST	SPE	BR [2:0]			MSTR	CPOL	CPHA							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
004h	SPI_CR2	Reserved																								TXEIE	RXNEIE	ERRIE	Reserved		SSOE	TXDMAE	RXDMAE							
	Reset value																									0	0	0			0	0	0			0	0	0		
008h	SPI_SR	Reserved																								BSY	OVR	MODF	CRCER	Reserved		TXE	RXNE							
	Reset value																									0	0	0	0			1	0							
00Ch	SPI_DR	Reserved																DR[15:0]																						
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
010h	SPI_CRCPR	Reserved																CRCPOLY [15:0]																						
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1						
014h	SPI_RXCR	Reserved																RxCRC[15:0]																						
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
018h	SPI_TXCR	Reserved																TxCRC[15:0]																						
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
01Ch	SPI_I2SCFGR	Reserved																				I2SMOD	I2SE	I2SCFG	PCMSYNC	Reserved	I2SSTD	CKPOL	DATLEN	CHLEN										
	Reset value																					0	0	0	0	0	Reserved	0	0	0	0	0	0	0	0					
020h	SPI_I2SPR	Reserved																								MCKOE	ODD	I2SDIV												
	Reset value																									0	0	0	0	0	0	0	0	0	0	0	0	1	0	

SeeTable1 for register start addresses.

## 27 Inter-Integrated Circuit(I<sup>2</sup>C) Interface

### 27.1 Introduction to I<sup>2</sup>C

The I<sup>2</sup>C<sup>(2)</sup> (inter-chip) bus interface connects the microcontroller to the serial I<sup>2</sup>C bus. It provides multi-host functionality to control all I<sup>2</sup>C bus specific timing, protocol, arbitration and timing. Both standard and fast modes are supported, and it is also compatible with SMBus 2.0.

The I<sup>2</sup>C module has a variety of uses, including CRC code generation and checksumming, SMBus (System Management Bus - SystemManagementBus) and PMBus (Power Management Bus - PowerManagementBus).

Depending on the needs of a particular device, DMA can be used to reduce the burden on the CPU.

### 27.2 I<sup>2</sup>C Main Features

- Parallel Bus/I<sup>2</sup>C Bus Protocol Converter
- Multi-master functionality: the module can be used as both master and slave devices
- I<sup>2</sup>C Master Device Function
  - generate a clock
  - Generate start and stop signals
- I<sup>2</sup>C slave device function
  - Programmable I<sup>2</sup>C address detection
  - Dual address capability in response to 2 slave addresses
  - Stop Bit Detection
- Generate and detect 7-bit/10-bit addresses and broadcast calls
- Supports different communication speeds
  - Standard speed (up to 100kHz)
  - Fast (up to 400kHz)
- Status Flags:
  - Transmitter/receiver mode flag
  - Byte send end flag
  - I<sup>2</sup>C-bus busy flag
- error message
  - Loss of arbitration in master mode
  - Answer after address/data transfer (ACK) error
  - Misaligned start or stop conditions detected
  - Prohibit overflow or underflow when stretching the clock function
- 2 interrupt vectors
  - 1 interrupt for successful address/data communication
  - 1 interrupt for errors
- Optional elongated clock function
- DMA with single byte buffer
- Configurable PEC (Packet Error Check) generation or checksum:
  - The PEC value can be transmitted as the last byte in transmit mode
  - PEC error checksum for last received byte
- SMBus 2.0 compatible
  - 25ms clock low timeout delay
  - 10ms cumulative clock low extension time for master device
  - 25ms cumulative clock low extension time from device
  - Hardware PEC generation/checksum with ACK control
  - Support for Address Resolution Protocol (ARP)
- SMBus compatible

### 27.3 I<sup>2</sup>C Functional Description

The I<sup>2</sup>C module receives and sends data, and converts data from serial to parallel, or parallel to serial. Interrupts can be enabled or disabled. The interface connects to the I<sup>2</sup>C bus via the data pin (SDA) and the clock pin (SCL). Connection to standard (up to 100kHz) or fast (up to 400kHz) I<sup>2</sup>C buses is allowed.

#### 27.3.1 Mode Selection

The interface can operate in one of the four modes described below:

- From transmitter mode
- slave mode
- Master Transmitter Mode
- Master Receiver Mode

The module operates in slave mode by default. The interface automatically switches from slave to master mode after generating a start condition; when arbitration is lost or a stop signal is generated, it switches from master to slave mode. Multi-master functionality is allowed.

### Communications Flow

In master mode, the I<sup>2</sup>C interface initiates the data transfer and generates the clock signal. Serial data transfers always start with a start condition and end with a stop condition. Both the start and stop conditions are generated under software control in the master mode.

In slave mode, the I<sup>2</sup>C interface recognizes its own address (7 or 10 bits) and the broadcast call address. The software has the ability to control the enabling or disabling of broadcast call address recognition.

Data and address are transmitted at 8 bits/byte, with the high bit coming first. The 1 or 2 bytes following the start condition are the address (1 byte for 7-bit mode, 2 bytes for 10-bit mode). Addresses are only sent in master mode.

During the 9th clock after 8 clocks of a byte transmission, the receiver must send an answer bit (ACK) back to the transmitter. Refer to the figure below.

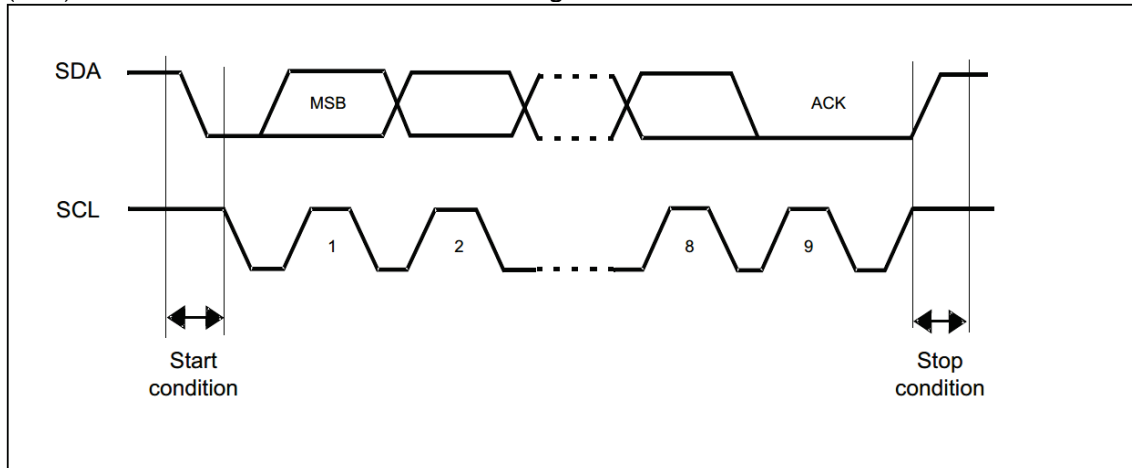


Figure241 I<sup>2</sup>C Bus Protocols

The software can enable or disable answering (ACK) and can set the address of the I<sup>2</sup>C interface (7-bit, 10-bit address, or broadcast call address).

The functional block diagram of the I<sup>2</sup>C interface is shown in the following figure.

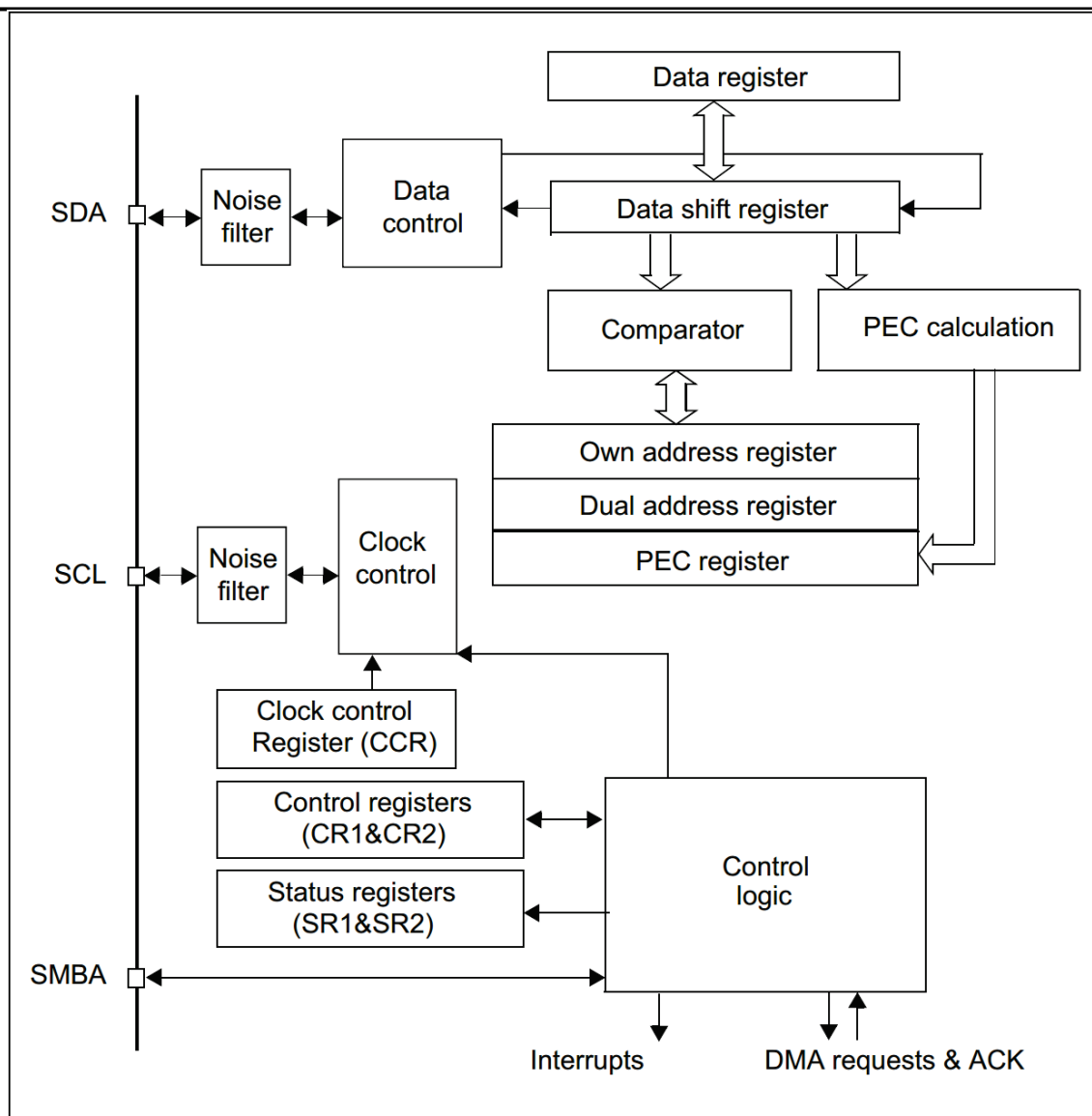


Figure 242 Functional Block Diagram of I<sup>2</sup>C

Notes: In SMBus mode, SMBALERT is an optional signal. If SMBus is disabled, this signal cannot be used.

## 27.3.2 I<sup>2</sup>C Slave Mode

By default, the I<sup>2</sup>C interface always operates in slave mode. Switching from slave mode to master mode requires generating a start condition.

In order to generate the correct timing, the input clock for this module must be set in the I<sup>2</sup>C\_CR2 register. The frequency of the input clock must be at least:

- Standard mode is: 2MHz
- In fast mode: 4MHz

Once the start condition is detected, the address received on the SDA line is sent to the shift register. It is then compared to the chip's own addresses OAR1 and OAR2 (when ENDUAL=1) or the broadcast call address (if ENGCB=1).

Notes: In 10-bit address mode, the comparison includes the header segment sequence (11110xx0), where the xx's are the two highest valid bits of the address.

Header segment or address mismatch: the I<sup>2</sup>C interface ignores it and waits for another start condition.

Header segment match (10-bit mode only): If the ACK bit is set to '1', the I<sup>2</sup>C interface generates an answer pulse and waits for the 8-bit slave address. Address Matching: The I<sup>2</sup>C interface generates the following timing:

- If ACK is set to '1', an answer pulse is generated
- Hardware sets the ADDR bit; if the ITEVFEN bit is set, an interrupt is generated

- If ENDUAL=1, software must read the DUALF bit to confirm which slave address was responded to.

In 10-bit mode, the slave device is always in receiver mode after receiving the address sequence. After receiving a header sequence that matches the address and the lowest bit is a '1' (i.e., 11110xx1), the transmitter mode is entered when a repeated start condition is received.

The TRA bit in Slave mode indicates whether you are currently in Receiver or Transmitter mode.

### From the Transmitter

After receiving the address and clearing the ADDR bit, the slave transmitter sends the bytes from the DR register to the SDA line via the internal shift register.

The slave device holds SCL low until the ADDR bit is cleared and the data to be sent has been written to the DR register. (See EV1 and EV3 in the figure below).

When an answer pulse is received.

- The TxE bit is set by hardware and generates an interrupt if the ITEVFEN and ITBUFEN bits are set.

If the TxE bit is set but no new data is written to the I2C\_DR register until the end of the next data send, the BTF bit is set and the I2C interface will hold SCL low until BTF is cleared; reading I2C\_SR1 and then writing to the I2C\_DR register afterward will clear the BTF bit.

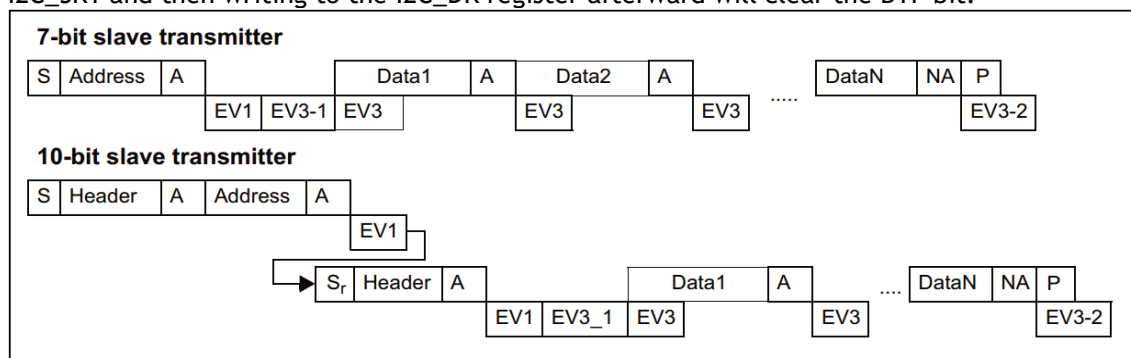


Figure243 Transmission Sequence Diagram from Transmitter

Description: S=Start (Start condition), S<sub>r</sub>=Repeat start condition, P=Stop (Stop condition), A=Should, NA=Non-response EVx=Event (Interrupt is generated when ITEVFEN=1)

EV1: ADDR=1, reading SR1 and then SR2 will clear this event.

EV3-1: TxE=1, shift register empty, data register empty, write DR.

EV3: TxE=1, shift register is non-empty, data register is empty, write DR will clear this event.

EV3-2: AF=1, writing '0' to the AF bit in the SR1 register clears the AF bit.

Notes: 1-EV1 and EV3 1 events elongate the SCL low until the end of the corresponding software sequence.

2-The software sequence for EV3 must be completed before the end of the current byte transfer.

### From the Receiver

After the address is received and ADDR is cleared, the slave receiver stores the bytes received from the SDA line through the internal shift register into the DR register. The I<sup>2</sup>C interface performs the following operations after each byte is received.

- If the ACK bit is set, an answer pulse is generated
- The hardware sets RxNE=1. If the ITEVFEN and ITBUFEN bits are set, an interrupt is generated.

If RxNE is set and the DR register is not read before the end of receiving new data, the BTF bit is set and the I<sup>2</sup>C interface will hold SCL low until BTF is cleared; reading I2C\_SR1 and then writing the I2C\_DR register afterward will clear the BTF bit. (See figure below).

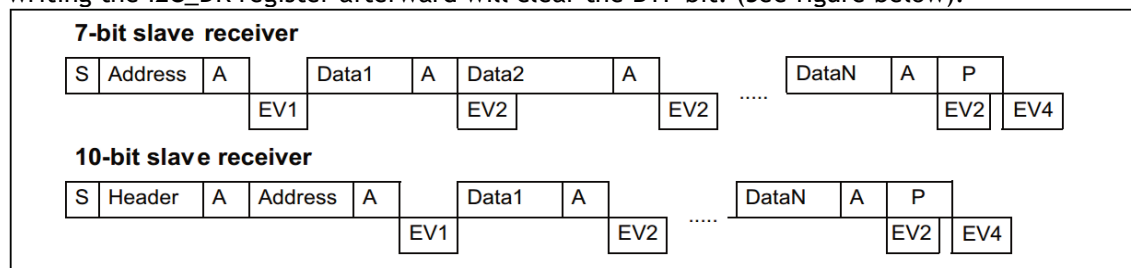


Figure244 Transmission Sequence Diagram from Receiver

Description: S=Start(Start condition), S<sub>r</sub>=Repeat start condition, P=Stop(Stop condition), A=Response, NA=Non-response, EV<sub>x</sub>=Event(interrupt generated when ITEVFEN=1)  
 EV1: ADDR=1, reading SR1 and then SR2 will clear this event.  
 EV2: RxNE=1, reading the DR will eliminate the event.  
 EV4: STOPF=1, reading SR1 and then writing the CR1 register will clear the event.

Notes: 1-EV1 events elongate the SCL low until the end of the corresponding software sequence.  
 2-EV2 software sequence must be completed before the end of the current byte transfer.

#### Close Slave Communication

After transmitting the last data byte, the master device generates a stop condition, which is detected by the I<sup>2</sup>C interface when:

- Setting STOPF=1 generates an interrupt if the ITEVFEN bit is set.
- The I<sup>2</sup>C interface then waits to read the SR1 register and then writes the CR1 register. (See EV4 above).

### 27.3.3 I<sup>2</sup>C Master Mode

In master mode, the I<sup>2</sup>C interface initiates the data transfer and generates the clock signal. Serial data transfers always begin with a start condition and end with a stop condition. When a start condition is generated on the bus via the START bit, the device enters the master mode. The following is the sequence of operations required by the Master Mode:

- Set the module's input clock in the I2C\_CR2 register to produce the correct timing
- Configuring the Clock Control Register
- Configuring the Rise Time Register
- Program the I2C\_CR1 register to start the peripheral
- Set the START bit in the I2C\_CR1 register to 1 to generate a start condition

The input clock frequency of the I2C module must be at least.

- In standard mode: 2MHz
- In fast mode: 4MHz

#### Starting Condition

Setting START=1 when BUSY=0, the I2C interface will generate a start condition and switch to master mode (M/SL position bit).

Notes: In master mode, setting the START bit will generate a restart condition by hardware after the current byte has been transferred.

Once the start condition is issued.

- The SB bit is set by hardware and an interrupt is generated if the ITEVFEN bit is set.
- The master device then waits to read the SR1 register, followed immediately by writing the slave address to the DR register (see EV5 of Figure 244 and Figure 245).

#### From the Address of the Sender

The slave address is sent to the SDA line through the internal shift register.

- When in 10-bit address mode, sending a header segment sequence generates the following events:

- The ADD10 bit is set by hardware and generates an interrupt if the ITEVFEN bit is set.
- The master device then waits to read the SR1 register before writing the second address byte to the DR register (see Figure 245 and Figure 246).

The ADDR bit is set by hardware and an interrupt is generated if the ITEVFEN bit is set.

The master device then waits for one read of the SR1 register, followed by a read of the SR2 register (see Figure 245 and Figure 246).

- In 7-bit address mode, only one address byte is sent.
  - Once that address byte has been delivered, the
  - The ADDR bit is set by hardware and an interrupt is generated if the ITEVFEN bit is set.

The master device then waits for one read of the SR1 register, followed by a read of the SR2 register (see Figure 245 and Figure 246).

Depending on the lowest bit of the delivered slave address, the master device decides whether to enter transmitter mode or receiver mode.

- In 7-bit address mode, the
  - To enter transmitter mode, the master device sends the slave address with the lowest bit set to '0'.



- To enter receiver mode, the master device sends the slave address with the lowest bit set to '1'.
  - In 10-bit address mode
    - To enter transmitter mode, the master device sends the header byte (11110xx0) and then the slave address with the lowest bit '0'. (Here xx represents the highest 2 bits of the 10-bit address.)
    - To enter receiver mode, the master device sends the header byte (11110xx0) followed by the slave address with the lowest bit '1'. Then resend a start condition followed by the header byte (11110xx1) (Here xx represents the highest 2 bits of the 10-bit address.)
- The TRA bit indicates whether the master device is in receiver or transmitter mode.

### Master Transmitter

After sending the address and clearing the ADDR bit, the master device sends the bytes from the DR register to the SDA line via the internal shift register.

The master device waits until TxE is cleared, (see EV8 of Figure 245). When an answer pulse is received.

- The TxE bit is set by hardware and generates an interrupt if the INEVFEN and ITBUFEN bits are set.

If TxE is set and no new data byte is written to the DR register before the end of the last data send, BTF is hardware set and the I2C interface will hold SCL low until BTF is cleared; reading I2C\_SR1 and then writing to the I2C\_DR register will clear the BTF bit.

### Shutting Down Communications

After writing the last byte in the DR register, a stop condition is generated by setting the STOP bit (see EV8\_2 of Figure 245), and then the I2C interface will automatically return to slave mode (M/S bit clear).

**Notes:** When the TxE or BTF position bits are present, the stop condition should be scheduled for the occurrence of an EV8\_2 event.

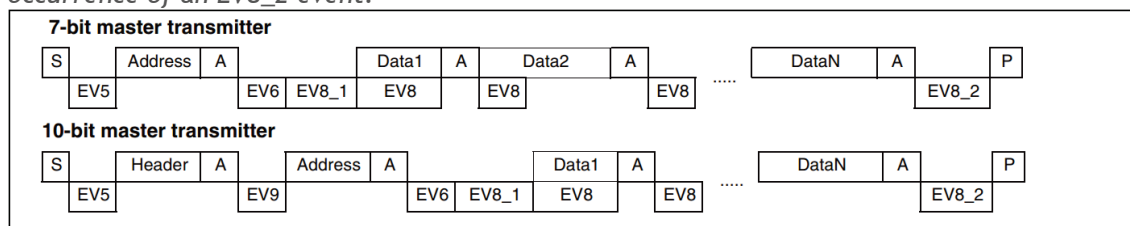


Figure 245 Master Transmitter Transmission Sequence Diagram

Description: S=Start (Start condition), S;=Repeat start condition, P=Stop (Stop condition), A=Response, NA=Non-Response, EVx=Event (Interrupt is generated when ITEVFEN=1).

EV5: SB=1, reading SR1 and then writing the address to the DR register will clear the event.

EV6: ADDR=1, reading SR1 and then SR2 will clear this event.

EV8\_1: TxE=1, shift register empty, data register empty, write DR register.

EV8: TxE=1, shift register is non-empty, data register is empty, write to DR register will clear this event.

EV8\_2: TxE=1, BTF=1, request to set the stop bit. the TxE and BTF bits are cleared by hardware when a stop condition is generated.

EV9: ADDR10=1, reading SR1 and then writing DR register will clear this event.

**Notes:** 1-EV5, EV6, EV9, EV8\_1 and EV8\_2 events elongate the SCL low until the end of the corresponding software sequence.

### Primary Receiver

After sending the address and clearing the ADDR, the I<sup>2</sup>C interface enters the master receiver mode. In this mode, the I<sup>2</sup>C interface receives data bytes from the SDA line and sends them through the internal shift register to the DR register. After each byte, the I2C interface performs the following operations in sequence:

- If the ACK bit is set, an answer pulse is issued.
- Hardware sets RxNE=1 and generates an interrupt if the INEVFEN and ITBUFEN bits are set (see Figure 246 EV7 of the bookmark 2040).

If the RxNE bit is set and the data in the DR register is not read before the end of receiving new data, the hardware will set BTF = 1. The I<sup>2</sup>C interface will hold SCL low until BTF is cleared; reading I2C\_SR1 followed by reading the I2C\_DR register will clear the BTF bit.

## Shutting Down Communications

The master device sends a NACK after the last byte is received from the slave device. Upon receipt of the NACK, the slave device releases control of the SCL and SDA lines; the master device can then send a stop/restart condition.

- In order to generate a NACK pulse after the last byte is received, the ACK bit must be cleared after the penultimate data byte is read (after the penultimate RxNE event).
- To generate a stop/restart condition, software must set the STOP/START bit after the penultimate data byte is read (after the penultimate RxNE event).
- When only one byte is received, just after EV6 (after clearing ADDR in case of EV6\_1) the generate bit of the answer and stop condition is to be turned off.

After a stop condition is generated, the I2C interface automatically returns to slave mode (M/SL bit is cleared).

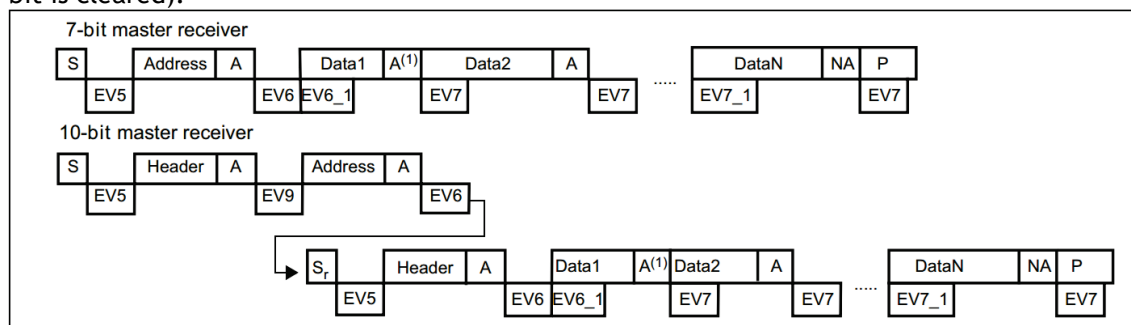


Figure 246 Master Receiver Transmission Sequence Diagram

Description: S=Start (Start condition), S<sub>r</sub>=Repeat start condition, P=Stop (Stop condition), A=Response, NA=Non-Response, EVx=Event (Interrupt is generated when ITEVFEN=1)

EV5: SB=1, reading SR1 and then writing the address to the DR register will except this event. EV6: ADDR=1, reading SR1 and then SR2 will clear this event. In 10-bit master receive mode, this event should be followed by setting CR2's START=1. EV6\_1: There is no corresponding event flag. is only suitable for receiving 1 byte. Exactly after EV6 (i.e., after clearing ADDR), the response and stop condition generation bits are to be cleaned out. EV7: RxNE=1 and the DR register is read to clear the event. EV7: RxNE=1 and the DR register is read to clear the event.

EV7\_1: RxNE=1, read DR register to clear this event. Set ACK=0 and STOP request.

EV9: ADDR10=1, reading SR1 and then writing DR register will clear this event.

1. If a single byte is received, it is NA.
2. EV5, EV6, and EV9 events stretch SCL low until the end of the corresponding software sequence.
3. The EV7's software sequence must be completed before the end of the current byte transfer.
4. The software sequence for EV6\_1 or EV7\_1 must be completed before the ACK pulse for the currently transmitted byte.

## 27.3.4 Error Condition (math.)

The following conditions may cause communication failure.

### Bus Error (BERR)

A bus error is generated when the I<sup>2</sup>C interface detects an external stop or start condition during an address or data byte transfer. At this time:

- The BERR bit is set to '1'; if the ITERREN bit is set, an interrupt is generated;
- In the slave mode case, the data is discarded and the hardware releases the bus:
  - If it is an incorrect start condition, the slave device considers it a restart and waits for an address or stop condition.
  - If it is an incorrect stop condition, the slave device operates as a normal stop condition while the hardware releases the bus.
- In the master mode case, the hardware does not release the bus while not affecting the current transfer state. It is up to the software to decide whether or not to abort the current transfer.

### Answer Fault (AF)

---

An answer error is generated when the interface detects a no answer bit. At this time:

- The AF bit is set and an interrupt is generated if the ITERREN bit is set;
- When the transmitter receives a NACK, it must reset the communication:
  - If it is in slave mode, the hardware releases the bus.
  - If it is in master mode, the software must generate a stop condition.

#### Arbitration Lost (ARLO)

An arbitration loss error is generated when the I<sup>2</sup>C interface detects an arbitration loss. at this point:

- The ARLO bit is set by hardware and an interrupt is generated if the ITERREN bit is set;
- The I<sup>2</sup>C interface automatically returns to slave mode (M/SL bit is cleared). When the I<sup>2</sup>C interface loses arbitration, it cannot respond to its slave address on the same transfer, but it can respond after the master device that won the bus sends a restart condition;
- Hardware release bus.

#### Overload/Underload Error (OVR)

In slave mode, if clock extension is disabled and the I<sup>2</sup>C interface is receiving data, an overload error occurs when it has received a byte (RxNE=1) but the previous byte of data in the DR register has not been read out. At this time:

- The last received data is discarded;
- On an overload error, software should clear the RxNE bit and the transmitter should retransmit the last byte sent.

In slave mode, an underload error occurs if the clock extension is disabled and the I<sup>2</sup>C interface is transmitting data when the new data has not been written to the DR register before the clock for the next byte arrives (TxE=1). At this time:

- The previous byte in the DR register will be issued repeatedly;
- The user should determine that in the event of an underload error, the receiving end should discard duplicate received data. The transmitting side shall update the DR register at the specified time per the I<sup>2</sup>C 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 cannot be done, the receiver should discard the first data.

## 27.3.5 SDA/SCL Line Control

- If clock extension is allowed:
  - Transmitter mode: if TxE=1 and BTF=1: The I<sup>2</sup>C interface keeps the clock line low before transmitting to wait for the software to read SR1 and then writes the data into the data registers (buffer and shift registers are empty).
  - Receiver mode: if RxNE=1 and BTF=1: The I<sup>2</sup>C interface holds the clock line low after receiving a data byte to wait for software to read SR1 and then reads the data register DR (buffer and shift register are full).
- If clock extension is disabled in slave mode:
  - If RxNE=1 and DR has not been read before the next byte is received, an overload error occurs. The last byte received is lost.
  - If TxE=1 and no new data is written to the DR before the next byte must be sent, an underload error occurs. The same byte will be sent repeatedly.
  - Does not control duplicate write conflicts.

## 27.3.6 SMBus

#### Present (sb for a job etc)

The System Management Bus (SMBus) is a two-wire interface. Through it, devices can communicate with each other and between devices and other parts of the system. It is based on the principle of I<sup>2</sup>C operation. SMBus provides a control bus for system and power management related tasks. A system utilizing SMBus can intercommunicate with multiple devices without the use of separate control lines.

The System Management Bus (SMBus) standard involves three types of devices. Slave devices: devices that receive or respond to commands. Master devices: devices used to send commands, generate clocks, and terminate transmissions. Host: a specialized master device that provides the primary interface to the system CPU. The host must have master-slave capabilities and must support the SMBus alert protocol. Only one host is allowed in a system.

### Similarities between SMBus and I<sup>2</sup>C

- 2 lines of bus protocol (1 clock, 1 data) + optional SMBus reminder line;
- Master-slave communication, with the master device providing the clock;
- Multi-Host Functions
- The SMBus data format is similar to the I<sup>2</sup>C 7-bit address format (see Figure 241);

### Differences between SMBus and I<sup>2</sup>C

The following table lists the differences between SMBus and I<sup>2</sup>C.

Table 142 Comparison of SMBus and I<sup>2</sup>C

SMBus	I <sup>2</sup> C
Maximum transmission speed 100kHz	Maximum transmission speed 400kHz
Minimum transmission speed 10kHz	No minimum transmission speed
35ms clock low timeout	No clock timeout
Fixed logic levels	Logic level is determined by VDD
Different address types (reserved, dynamic, etc.)	7-bit, 10-bit, and broadcast call slave address types
Different bus protocols (quick commands, handling calls, etc.)	No bus protocol

### SMBus Application Uses

Using the system management bus, a device can provide manufacturer information, tell the system its model/part number, save the status of suspend events, report different types of errors, receive control parameters, and return its status. SMBus provides the control bus for system and power management related tasks.

### Equipment Identification

On the System Management Bus, any device that is in slave mode has a unique address called the slave address. Refer to the SMBus specification version 2.0 (<http://smbus.org/specs/>) for a table of reserved slave addresses.

### Bus Protocol

The SMBus specification supports nine bus protocols. For detailed information on these protocols and the SMBus address types, refer to the SMBus specification version 2.0 (<http://smbus.org/specs/>). These protocols are implemented by the user's software.

### Address Resolution Protocol (ARP)

SMBus slave address conflicts can be resolved by dynamically assigning a new unique address to each slave device. The Address Resolution Protocol (ARP) has the following properties:

- Addresses are assigned using the standard SMBus physical layer arbitration mechanism;
- The assigned address remains the same while the device maintains power, and also allows the device to retain its address after a power failure.
- There is no additional SMBus packing overhead after the address is assigned (i.e., accessing a device with an assigned address takes the same amount of time as accessing a device with a fixed address);
- Any SMBus master device can traverse the bus.

### Unique Device Identifier (UDID)

In order to assign addresses, a mechanism is needed to distinguish each device, and each device must have a unique device identifier.

For more information about the 128-bit UDID on ARP, refer to the SMBus specification for version 2.0 (<http://smbus.org/specs/>).

### SMBus Alert Mode

SMBus alert is an optional signal with an interrupt line for devices that wish to extend their control capabilities at the expense of a pin. SMBALERT is a line-and-signal like the SCL and SDA signals. SMBALERT is usually used in conjunction with the SMBus broadcast call address. SMBus-related messages are 2 bytes.

A slave-only device can use SMBALERT to signal the host that it wishes to communicate by setting the ALERT bit on the I2C\_CR1 register. The host handles this interrupt and accesses all

SMBALERT devices via the Alert Response Address ARA (address value 0001 100x). Only those devices that have pulled SMBALERT low can respond to ARA. This state 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 supplied by the transmitting device is placed on the seven highest bits of the byte, and the eighth bit can be either a '0' or a '1'.

If more than one device pulls SMBALERT low, the highest priority device (smallest address) will win the right to communicate through standard arbitration during the address transfer. After acknowledging the slave address, this device must not pull its SMBALERT down again, and if the host still sees a low SMBALERT when the message transfer is complete, it knows it needs to read the ARA again.

Hosts that do not implement the SMBALERT signal can access the ARA periodically.

For more detailed information on SMBus alert mode, please refer to the SMBus specification for version 2.0 (<http://smbus.org/specs/>).

### Timeout Error

There are many differences between I<sup>2</sup>C and SMBus in terms of timing specifications.

SMBus defines a clock low timeout, a 35ms timeout. SMBus specifies TLOW:SEXT as the cumulative clock low extension time for the slave. SMBus specifies TLOW:MEXT as the cumulative clock low extension time for the master. SMBus specifies TLOW:MEXT as the cumulative clock low extension time for the master. For more timeout details, please refer to the SMBus specification for version 2.0 (<http://smbus.org/specs/>).

The status flags Timeout or Tlow error in I2C\_SR1 indicate the state of this feature.

How to use the SMBus mode interface

In order to switch from I<sup>2</sup>C mode to SMBus mode, the following steps should be performed:

- Sets the SMBus bit in the I2C\_CR1 register;
- Configure the SMBTYPE and ENARP bits in the I2C\_CR1 register as required by the application.

If the device is to be configured as a master, see Section 24.3.3, I<sup>2</sup>C Master Mode, for the procedure to generate the start condition. Otherwise, see Section 22.3.2 I<sup>2</sup>C Slave Mode.

The software program must handle multiple SMBus protocols.

- If ENARP=1 and SMBTYPE=0, the SMB device default address is used.
- If ENARP=1 and SMBTYPE=1, use the SMB Master Device header field.
- If SMBALERT=1, use the SMB alert response address.

## 27.3.7 DMA Request

DMA requests (when enabled) are used only for data transfers. A DMA request is generated when the data register becomes empty when transmitting or full when receiving. The DMA request must be responded to before the end of the current byte transfer. When the amount of data transfer set for the corresponding DMA channel has been completed, the DMA controller sends the end-of-transfer signal ETO to the I<sup>2</sup>C interface and generates a transfer-complete interrupt when interrupts are allowed:

- Main transmitter: in the EOT interrupt service program, the DMA request needs to be disabled and then the stop condition is set after waiting for the BTF event.
- Master Receiver: When the number of data to be received is greater than or equal to 2, the DMA controller sends a hardware signal EOT\_1, which corresponds to a DMA transfer (number of bytes - 1). If the LAST bit is set in the I2C\_CR2 register, the hardware will automatically send a NACK for the next byte after sending EOT\_1. Where interrupts are allowed, the user can generate a stop condition in the interrupt service program for the completion of a DMA transfer.

### Sending with DMA

DMA mode can be activated by setting the DMAEN bit in the I2C\_CR2 register. As soon as the TxE bit is set, data will be loaded into the I2C\_DR register by DMA from the preset memory area. To assign a DMA channel to the I2C, the following steps must be performed (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. The memory address is set in the DMA\_CMARx register. Data is transferred from this memory area to I2C\_DR after each TxE event.
  3. Set the number of bytes to be transferred in the DMA\_CNDTRx register. This value is decremented after each TxE event.
  4. Configure the channel priority using the PL[0:1] bits in the DMA\_CCRx register.
  5. Sets the DIR bit in the DMA\_CCRx register and, depending on application requirements, can be configured to issue an interrupt request when the entire transfer is half or fully completed.
  6. Activate the channel by setting the EN bit on the DMA\_CCTx register.
- When the number of data transfers set in the DMA controller has been completed, the DMA controller sends an end-of-transfer EOT/EOT\_1 signal to the I2C interface. A DMA interrupt will be generated if interrupts are allowed.

*Notes: Do not set the ITBUFEN bit of the I2C\_CR2 register if DMA is used for transmitting.*

#### Receive using DMA

DMA receive mode can be activated by setting the DMAEN bit in the I2C\_CR2 register. Each time a data byte is received, the data in the I2C\_DR register will be transferred by DMA to the set memory area (refer to the DMA description). To set the DMA channel for I<sup>2</sup>C reception, the following steps must be performed (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 storage 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 storage area after each RxNE event.
3. Set the number of bytes to be transferred in the DMA\_CNDTRx register. This value is decremented after each RxNE event.
4. Configure the channel priority with PL[0:1] in the DMA\_CCRx register.
5. Clearing the DIR bit in the DMA\_CCRx register sets the interrupt request to be issued when the data transfer is half or fully completed, depending on the application requirements.
6. Setting the EN bit in the DMA\_CCRx register activates the channel.

When the number of data transfers set in the DMA controller has been completed, the DMA controller sends an end-of-transfer EOT/EOT\_1 signal to the I<sup>2</sup>C interface. A DMA interrupt will be generated if interrupts are allowed.

*Notes: Do not set the ITBUFEN bit of the I2C\_CR2 register if DMA is used for reception.*

## 27.3.8 Packet Error Check (PEC)

The Packet Error Checksum (PEC) calculator is used to improve the reliability of communications by using the following CRC-8 polynomial for each bit of serial data:

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

- The PEC calculation is activated by the ENPEC bit in the I2C\_CR1 register. The PEC uses the CRC-8 algorithm for all message bytes, including address and read/write bits.
  - On transmit: set the PEC transmit bit of the I2C\_CR1 register at the last TxE event and the PEC will be sent after the last byte.
  - On receive: set the PEC bit of the I2C\_CR1 register after the last RxNE event. If the next received byte is not equal to the internally calculated PEC, the receiver sends a NACK. In case of a master receiver, a NACK will be sent after the PEC, regardless of the result of the proofreading. The PEC bit has to be set before the ACK pulse for the current byte is received.
- The PECERR error flag/interrupt is available in the I2C\_SR1 register.
- If both DMA and PEC calculators are activated:
  - On transmit: when the I2C interface receives the EOT signal from the DMA controller, it automatically transmits the PEC after the last byte.
  - On receive: when the I2C interface receives an EOT\_1 signal from the DMA, it will automatically take the next byte as a PEC and will check it. A DMA request is generated after the PEC is received.
- To allow intermediate PEC transfers, there is a control bit (LAST bit) in the I2C\_CR2 register that is used to determine if it is indeed the last DMA transfer. If it is indeed the last DMA request from the master receiver, a NACK is automatically sent after the last byte is received.
- PEC calculations fail when arbitration is lost.

## 27.4 I<sup>2</sup>C Interrupt Request

The following table lists all I<sup>2</sup>C interrupt requests

Table143 I<sup>2</sup>C Interrupt Request Table:

disruption event	event marker	Open control bit
Start bit sent (master)	SB	ITEVFEN
Address sent (master) or address match (slave)	ADDR	
10-bit header segment sent (master)	ADD10	
Stop (from) received	STOPF	
Data byte transfer complete	BTF	
Receive buffer not empty	RxNE	ITEVFEN and ITBUFEN
Send buffer empty	TxE	
bus error	BERR	ITERREN
Arbitration lost (main)	ARLO	
Response Failure	AF	
Overload/Underload	OVR	
PEC error	PECERR	
Timeout/Tlow Error	TIMEOUT	
SMBus alert	SMBALERT	

Notes: 1. SB, ADDR, ADD10, STOPF, BTF, RxNE and TxE are converged into the same interrupt channel by logical or.

2. BERR, ARLO, AF, OVR, PECERR, TIMEOUT, and SMBALERT are converged into the same interrupt channel via logical or.

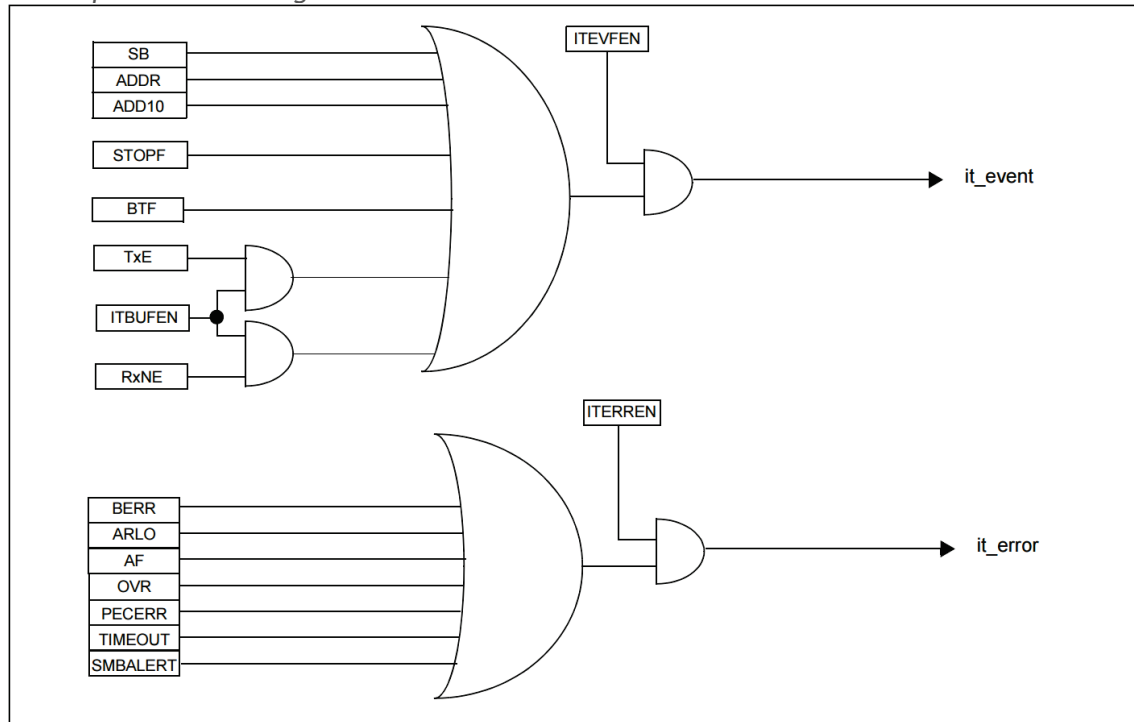


Figure247 I<sup>2</sup>C Interrupt Mapping Map

## 27.5 I<sup>2</sup>C Debug Mode

When the microcontroller enters debug mode (the Cortex-M3 core is stopped), the SMBUS timeout control either continues to work normally or can be stopped depending on the DBG\_I2Cx\_SMBUS\_TIMEOUT configuration bit in the DBG module. See Section31.16.2 for details.

## 27.6 I<sup>2</sup>C Register Description

See Section1 for details on the abbreviations used in the register descriptions.

These peripheral registers can be operated in half-word (16-bit) or word (32-bit) format.

## 27.6.1 Control Register 1 (I2C\_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWRST	Reserved	ALERT	PEC	POS	ACK	STOP	START	STRETCH	ENG	ENPEC	ENARP	SMBTY PE	Reserved	SMBUS	PE
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw

Bit	notation	clarification
15	SWRST	<b>SWRST:</b> Software reset When set, I2C is in a reset state. Be sure that the I2C pins are released and the bus is empty before resetting the bit. 0: The I2C module is not in the reset state; 1: I2C module is in reset state. Note: This bit can be used when the BUSY bit is '1' and no stop condition is detected on the bus.
14	Reserved	Reserved, always reads 0.
13	ALERT	<b>ALERT:</b> SMBus alert This bit can be set or cleared by software; when PE=0, it is cleared by hardware. 0: Release the SMBAlert pin to make it high. The alert response address header immediately follows the NACK signal; 1: Drive the SMBAlert pin low. The alert response address header immediately follows the ACK signal.
12	PEC	<b>PEC:</b> Packet error checking (Packet error checking) Software can set or clear this bit; hardware clears it after a PEC has been transmitted, or during a start or stop condition, or when PE=0. 0: No PEC transmission; 1: PEC transmission (in transmit or receive mode). Note: The calculation of the PEC fails when arbitration is lost.
11	POS	<b>POS:</b> Answer/PEC Position (for data reception) (Acknowledge/PEC Position (for data reception)) This bit can be set or cleared by software, or cleared by hardware when PE=0. 0: The ACK bit controls the (N)ACK of the byte being received in the current shift register. The PEC bit indicates that the byte in the current shift register is a PEC; 1: ACK bit controls the (N)ACK of the next byte received in the shift register. The PEC bit indicates that the next byte received in the shift register is a PEC. Note: The POS bit can only be used in a 2-byte receive configuration and must be configured before receiving data. In order to NACK the 2nd byte, the ACK bit must be cleared after clearing ADDR for. In order to detect the PEC of the 2nd byte, the PEC bit must be set when stretching the ADDR event after the POS bit has been configured.
10	ACK	<b>ACK:</b> Acknowledge enable This bit can be set or cleared by software or, when PE=0, by hardware. 0: No answer returned; 1: Returns an answer (matching address or data) after receiving a byte.
9	STOP	<b>STOP:</b> Stop generation This bit can be set or cleared by software; or cleared by hardware when a stop condition is detected; and set by hardware when a timeout error is detected. In master mode: 0: No stopping conditions are generated; 1: Generate a stop condition after the current byte is transmitted or after the current start condition is issued. In slave mode: 0: No stopping conditions are generated; 1: Transmit or release the SCL and SDA lines at the current byte. Note: When the STOP, START, or PEC bit is set, software should not perform any write operations to I2C_CR1 until the hardware clears this bit; otherwise there is a possibility that the STOP, START, or PEC bit will be set a 2nd time.
8	START	<b>START:</b> Start generation This bit can be set or cleared by software or cleared by hardware when the start condition is issued or when PE=0. In master mode: 0: No starting conditions are generated; 1: Repeat to generate the starting condition. In slave mode: 0: No starting conditions are generated; 1: When the bus is idle, a start condition is generated.
7	NOSTRETCH	<b>NOSTRETCH:</b> Clock stretching disable (Slave mode) This bit is used when the ADDR or BTF flag is set to disable clock extension in



		slave mode until it is reset by software. 0: Allow clock extension; 1: Clock extension is prohibited.
6	ENG	<b>ENG</b> : Broadcast call enable (General call enable) 0: Broadcast calls are prohibited. Responds to address 00h with a non-answer; 1: Allow broadcast calls . to answer response address 00h.
5	ENPEC	<b>ENPEC</b> : PEC enable 0: PEC calculation is disabled; 1: Turn on PEC calculations.
4	ENARP	<b>ENARP</b> : ARP enable 0: ARP is disabled; 1: Enable ARP. If SMBTYPE=0, the default address of the SMBus device is used. If SMBTYPE=1, the primary address of the SMBus is used.
3	SMBTYPE	<b>SMBTYPE</b> : SMBus type 0: SMBus device; 1: SMBus host.
2	Reserved	Reserved bit, hardware forced to 0.
1	SMBUS	<b>SMBUS</b> : SMBus mode (SMBus mode) 0: I2C mode; 1: SMBus mode.
0	PE	<b>PE</b> : I2C module enable (Peripheral enable) 0: Disable the I2C module; 1: Enable I2C module: According to the setting of SMBus bits, the corresponding I/O ports need to be configured for multiplexing. Note: If communication is in progress when this bit is cleared, the I2C module is disabled and returns to the idle state at the end of the current communication. All bits are cleared as a result of PE = 0 occurring at the end of communication. This bit must never be cleared before communication ends in master mode.

## 27.6.2 Control Register 2 (I2C\_CR2)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			LAST	DMAEN	ITBUFEN	ITEVTEN	ITERREN	Reserved			FREQ[5:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bit	notation	clarification
15:13	Reserved	Reserved bit, hardware forced to 0
12	LAST	<b>LAST</b> : DMA last transfer 0: The EOT of the next DMA is not the last transmission; 1: The EOT of the next DMA is the last transmission. Note: This bit is used in master receive mode to enable a NACK to be generated on the last data received.
11	DMAEN	<b>DMAEN</b> : DMA requests enable 0: DMA request is disabled; 1: DMA request is allowed when Tx=1 or Rx=1.
10	ITBUFEN	<b>ITBUFEN</b> : Buffer interrupt enable 0: When Tx=1 or Rx=1, no interrupt is generated; 1: An event interrupt is generated when Tx=1 or Rx=1 (regardless of the state of DMAEN).
9	ITEVTEN	<b>ITEVTEN</b> : Event interrupt enable 0: Disable event interruption; 1: Allow event interruptions. This interrupt will be generated under the following conditions: -SB=1 (master mode); -ADDR=1 (master/slave mode); -ADD10=1 (master mode); -STOPF=1 (slave mode); -BTF=1, but no Tx or Rx event; -Tx event is 1 if ITBUFEN=1; -If ITBUFEN=1, the Rx event is 1.
8	ITERREN	<b>ITERREN</b> : Error interrupt enable 0: Disable error interrupt; 1: Error interrupts are allowed. This 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 bit, hardware forced to 0.
5:0	FREQ[5:0]	<b>FREQ[5:0]:</b> I2C module clock frequency (Peripheral clock frequency) The correct input clock frequency must be set to produce the correct timing, the allowable range is between 2 ~ 36MHz: 000000: Disabled 000001: Disabled 000010: 2MHz ... 100100: 36MHz Greater than 100100: Disable.

### 27.6.3 Self Address Register 1 (I2C\_OAR1)

Reset address offset: 0x08

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDMODE	Reserved	Reserved				ADD[9:8]		ADD[7:1]							ADD0
rw						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	notation	clarification
15	ADDMODE	<b>ADDMODE:</b> Addressing mode (slave mode) 0: 7-bit slave address (does not respond to 10-bit addresses); 1: 10-bit slave address (does not respond to 7-bit addresses).
14	Reserved	It must always be kept at '1' by the software.
13:10	Reserved	Reserved bit, hardware forced to 0.
9:8	ADD[9:8]	<b>ADD[9:8]:</b> interface address (Interface address) 7-bit address mode do not care. Bits 9 to 8 of the address in 10-bit address mode.
7:1	ADD[7:1]	<b>ADD[7:1]:</b> bits 7-1 of the interface address (Interface address) address.
0	ADD0	<b>ADD0:</b> Interface address (Interface address) 7-bit address mode do not care. Address bit 0 in 10-bit address mode.

### 27.6.4 Own Address Register 2 (I2C\_OAR2)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								ADD2[7:1]							ENDUAL
								rw	rw	rw	rw	rw	rw	rw	rw

Bit	notation	clarification
15:8	Reserved	Reserved bit, hardware forced to 0
7:1	ADD2[7:1]	<b>ADD2[7:1]:</b> bits 7-1 of the address of the interface address (Interface address) in dual address mode.
0	ENDUAL	<b>ENDUAL:</b> Dual addressing mode enable bit (Dual addressing mode enable) 0: In 7-bit address mode, only OAR1 is recognized; 1: In 7-bit address mode, both OAR1 and OAR2 are recognized.

### 27.6.5 Data Register (I2C\_DR)

Address offset: 0x10

Reset value: 0x0000

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

Bit	notation	clarification
15:8	Reserved	Reserved bit, hardware forced to 0
7:0	DR[7:0]	<b>DR[7:0]:</b> 8-bit data register (8-bit data register) Used to store received data or to place data to be sent to the bus. Transmitter Mode: automatically initiates a data transfer when a byte is written to the DR register. Once the transfer has started (Tx=1), the I2C module will maintain

		<p>a continuous flow of data if the next data to be transferred is written to the DR register in time.</p> <p>Receiver mode: The received byte is copied to the DR register (RxNE=1). Continuous data transfer can be realized by reading the data register before the next byte (RxNE=1) is received.</p> <p>Note: In slave mode, the address is not copied into the data register DR;</p> <p>Note: Hardware does not manage write conflicts (if TxNE=0, it can still write to the data register);</p> <p>Note: If an ARLO event occurs while the ACK pulse is being processed, the received byte is not copied into the data register and therefore it cannot be read.</p>
--	--	--

## 27.6.6 Status Register 1 (I2C\_SR1)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMBAL ERT	TIMEO UT	Reserv ed	PECER R	OVR	AF	ARLO	BERR	TxE	RxNE	Reserv ed	STOPF	ADD10	BTF	ADDR	SB
rc_w0	rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	r	r		r	r	r	r	r

Bit	notation	clarification
15	SMBALERT	<p><b>SMBALERT:</b> SMBus alert (SMBus alert) in SMBus host mode:</p> <p>0: No SMBus alert;</p> <p>1: Generate SMBAlert alert events on pins. In SMBus slave mode:</p> <p>0: No SMBAlert response address header sequence;</p> <p>1: SMBAlert response address header sequence received until SMBAlert goes low. -This bit is cleared by software writing '0' or by hardware when PE=0.</p>
14	TIMEOUT	<p><b>TIMEOUT:</b> Timeout or Tlow error</p> <p>0: No timeout error;</p> <p>1: SCL is in low has reached 25ms (timeout); or the cumulative clock extension time of the host low has exceeded 10ms (Tlow:mext); or the cumulative clock extension time of the slave device low has exceeded 25ms (Tlow:sext).</p> <p>-When this bit is set in slave mode: the slave device resets the communication and the hardware releases the bus.</p> <p>-When this bit is set in master mode: hardware issues a stop condition.</p> <p>-This bit is cleared by software writing '0' or by hardware when PE=0.</p>
13	Reserved	Reserved bit, hardware forced to 0.
12	PECERR	<p><b>PECERR:</b> PEC Error in reception</p> <p>0: No PEC Error: the receiver returns an ACK after receiving a PEC (if ACK=1);</p> <p>1: There is a PEC error: the receiver returns NACK (whatever value ACK is) after receiving a PEC. -This bit is cleared by software writing '0' or by hardware when PE=0.</p>
11	OVR	<p><b>OVR:</b> Overrun/Underrun (Overrun/Underrun) 0: No overrun/underrun;</p> <p>1: Overload/underload occurs.</p> <p>-When NOSTRETCH=1, this bit is set by hardware in slave mode at the same time:</p> <p>-When a new byte is received in receive mode (including the ACK answer pulse) and the contents of the data register have not been read out, the newly received byte is lost.</p> <p>-In send mode when a new byte is to be sent without new data being written to the data register, the same byte will be sent twice.</p> <p>-This bit is cleared by software writing '0' or by hardware when PE=0.</p> <p>Note: If a write operation to the data register occurs very close to the rising edge of SCL, the data sent is indeterminate and a hold time error occurs.</p>
10	AF	<p><b>AF:</b> Acknowledge failure</p> <p>0: No answer failure;</p> <p>1: Failed to answer.</p> <p>-When no answer is returned, the hardware will set this bit to '1'.</p> <p>-This bit is cleared by software writing '0' or by hardware when PE=0.</p>
9	ARLO	<p><b>ARLO:</b> Arbitration lost (master mode)</p> <p>0: No arbitration loss detected;</p> <p>1: Loss of arbitration detected.</p> <p>Hardware will set this bit to '1' when the interface loses control of the bus to another host. -This bit is cleared by software writing '0' or by hardware when PE=0.</p> <p>After an ARLO event, the I2C interface automatically switches back to slave mode (M/SL=0).</p> <p>Note: In SMBUS mode, arbitration of data in slave mode occurs only during the data phase, or the answer transmission interval (excluding answers to addresses).</p>
8	BERR	<p><b>BERR:</b> Bus error</p> <p>0: No start or stop condition error;</p> <p>1: Error in start or stop condition.</p> <p>-Hardware sets this bit '1' when the interface detects an incorrect start or stop</p>

		condition. -This bit is cleared by software writing '0' or by hardware when PE=0.
7	TxE	<p><b>TxE:</b> Data register empty (transmitters)</p> <p>0: The data register is not empty;</p> <p>1: Data register empty.</p> <p>-This bit is set to '1' when the data register is empty when sending data, and it is not set during the send address phase.</p> <p>-Software writing data to the DR register clears this bit; or it is automatically cleared by hardware after a start or stop condition occurs, or when PE=0.</p> <p>This bit is not set if a NACK is received, or if the next byte to be sent is a PEC (PEC=1).</p> <p>Note: The TxE bit cannot be cleared after writing the 1st data to be sent, or writing data when BTF is set, because the data register is still empty.</p>
6	RxNE	<p><b>RxNE:</b> Data register not empty(receivers)</p> <p>0: The data register is empty;</p> <p>1: The data register is not empty.</p> <p>-This bit is set '1' when the data register is not empty during reception. This bit is not set during the receive address phase. -Software read and write operations to the data register clear this bit, or it is cleared by hardware when PE=0.</p> <p>RxNE is not set when an ARLO event occurs.</p> <p>Note: When BTF is set, reading data does not clear the RxNE bit because the data register remains full.</p>
5	Reserved	Reserved bit, hardware forced to 0
4	STOPF	<p><b>STOPF:</b> Stop condition detection bit (slave mode) (Stop detection (slave mode))</p> <p>0: No stop condition detected;</p> <p>1: Stop condition detected.</p> <p>-After an answer (if ACK=1), the hardware places this position '1' when the slave device detects a stop condition on the bus.</p> <p>-After a software read of the SR1 register, a write operation to the CR1 register clears the bit, or when PE=0, hardware clears the bit.</p> <p>Note: The STOPF bit is not set after a NACK is received.</p>
3	ADD10	<p><b>ADD10:</b> 10-bit header sent (Master mode)</p> <p>0: No ADD10 event occurred;</p> <p>1: The master device has sent the first address byte.</p> <p>-In 10-bit address mode, when the master device has sent the first byte out, the hardware sets this position '1'.</p> <p>-After a software read of the SR1 register, a write operation to the CR1 register clears the bit, or hardware clears the bit when PE=0.</p> <p>Note: The ADD10 bit is not set after a NACK is received.</p>
2	BTF	<p><b>BTF:</b> Byte transfer finished</p> <p>0: Byte sending not completed;</p> <p>1: End of byte sending.</p> <p>When NOSTRETCH=0, the hardware places this position '1' in the following cases:</p> <p>-On reception, when a new byte is received (including the ACK pulse) and the data register has not been read (RxNE=1).</p> <p>-When a new data will be sent and the data register has not yet been written with the new data (TxNE=1) at the time of sending.</p> <p>-This bit is cleared by a read or write operation to the data register after a software read of the SR1 register, or by hardware after a start or stop condition is sent in a transmission, or when PE=0.</p> <p>Note: The BTF bit is not set after a NACK is received.</p> <p>If the next byte to be transmitted is a PEC (TRA is '1' in the I2C_SR2 register while PEC is '1' in the I2C_CR1 register), the BTF bit will not be set.</p>
1	ADDR	<p><b>ADDR:</b> Address sent(master mode)/matched(slave mode)</p> <p>A read operation to the SR2 register after a software read of the SR1 register will clear this bit, or when PE=0, the bit will be cleared by hardware.</p> <p><b>Address Matching (Slave Mode)</b></p> <p>0: Address mismatch or no address received;</p> <p>1: Received address match.</p> <p>-Hardware sets this position '1' (when the corresponding setting is enabled) when a slave address is received that matches the contents of the OAR register, or when a broadcast call occurs, or when the default address of the SMBus device or the SMBus host recognizes an SMBus alert.</p> <p><b>Address has been sent (master mode)</b></p> <p>0: Address sending is not finished;</p> <p>1: End of address sending.</p> <p>-10-bit address mode, this bit is set to '1' when an ACK is received for the second byte of the address. -7-bit address mode, this bit is set to '1' when an ACK of the address is received.</p> <p>Note: The ADDR bit is not set after a NACK is received.</p>
0	SB	<p><b>SB:</b> Start bit (Master mode)</p> <p>0: Starting condition not sent;</p>

1: Starting conditions have been sent.  
 -This bit is set to '1' when the start condition is sent out.  
 -After software reads the SR1 register, a write data register operation clears the bit, or hardware clears the bit when PE=0.

## 27.6.7 Status Register 2 (I2C\_SR2)

Address offset: 0x18

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PEC[7:0]								DUALF	SMBHOST	SMBDEFAULT	GENCALL	Reserved	TRA	BUSY	MSL
r	r	r	r	r	r	r	r	r	r	r	r		r	r	r

Bit	notation	clarification
15:8	PEC[7:0]	<b>PEC[7:0]:</b> packet error checking register When ENPEC=1, PEC[7:0] stores the value of internal PEC.
7	DUALF	<b>DUALF:</b> Dualflag (Slave mode) 0: The received address matches the contents within OAR1; 1: The received address matches the contents within OAR2. -Hardware clears this bit when a stop condition or a repeated start condition is generated, or when PE=0.
6	SMBHOST	<b>SMBHOST:</b> SMBus host header (Slave mode) 0: SMBus host address not received; 1: SMBus host address received when SMBTYPE=1 and ENARP=1. -Hardware clears this bit when a stop condition or a repeated start condition is generated, or when PE=0.
5	SMBDEFAULT	<b>SMBDEFAULT:</b> SMBus device default address (Slave mode) 0: The default address of the SMBus device was not received; 1: When ENARP=1, the default address of the SMBus device is received. -Hardware clears this bit when a stop condition or a repeated start condition is generated, or when PE=0.
4	GENCALL	<b>GENCALL:</b> Broadcast call address (Slave mode) (General call address (Slave mode)) 0: No broadcast call address received; 1: The address at which the broadcast call was received when ENGC=1. -Hardware clears this bit when a stop condition or a repeated start condition is generated, or when PE=0.
3	Reserved	Reserved bit, hardware forced to 0
2	TRA	<b>TRA:</b> Transmitter/receiver 0: Data received; 1: Data has been sent; At the end of the entire address transfer phase, this bit is set according to the R/W bit of the address byte. Hardware clears it after a stop condition (STOPF=1), a repeated start condition, or a bus arbitration loss (ARLO=1) is detected, or when PE=0.
1	BUSY	<b>BUSY:</b> Bus busy 0: No data communication on the bus; 1: Data communication is in progress on the bus. -Hardware sets this position '1' when SDA or SCL is detected to be low; -Hardware clears this bit when a stop condition is detected. This bit indicates the bus communication currently in progress, and this information is still updated when the interface is disabled (PE=0).
0	MSL	<b>MSL:</b> Master/slave mode (Master/slave) 0: From mode; 1: Master mode. -When the interface is in master mode (SB=1), the hardware bits this position; -Hardware clears this bit when a stop condition is detected on the bus, when arbitration is lost (when ARLO=1), or when PE=0.

## 27.6.8 Clock Control Register (I2C\_CCR)

Address offset: 0x1C

Reset value: 0x0000

Notes: 1. It is required that FPCCLK1 should be an integer multiple of 10MHz so that a fast 400KHz clock can be generated correctly.

2. CCR register can only be set when I2C is turned off (PE=0)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F/S	DUTY	Reserved	CCR[11:0]												
rW	rW			rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW

Bit	notation	clarification
15	F/S	F/S: I2C master mode selection (I2C master mode selection) 0: Standard mode I2C; 1: Fast mode I2C.
14	DUTY	DUTY: Duty cycle in fast mode (Fast mode duty cycle) 0: In fast mode: Tlow/Thigh = 2; 1: In fast mode: Tlow/Thigh = 16/9 (see CCR).
13:12	Reserved	Reserved bit, hardware forced to 0.
11:0	CCR[11:0]	CCR[11:0]: Clock control divider coefficients in Fast/Standard mode (Mastermode) (ClockcontrolregisterinFast/Standardmode(Mastermode)) This division factor is used to set the SCL clock in master mode. In I2C standard mode or SMBus mode: Thigh=CCR× TPCLK1Tlow=CCR× TPCLK1 In I2C fast mode: if DUTY=0: Thigh=CCR× TPCLK1 Tlow=2× CCR× TPCLK1 If DUTY=1: (speed up to 400kHz) Thigh=9× CCR× TPCLK1 Tlow=16× CCR× TPCLK1 For example, in standard mode, a frequency of 100kHz SCL is generated: If FREQR=08 and TPCLK1=125ns, the CCR must write 0x28 (40× 125ns=5000ns). Note: 1. The minimum value allowed to be set is 0x04, and the minimum value allowed in fast DUTY mode is 0x01; 2. Thigh=tr(SCL)+tw(SCLH), see the definition of these parameters in the datasheet for details; 3. Tlow=tf(SCL)+tw(SCLL), see the definition of these parameters in the datasheet for details; 4. There is no filter for these delays; 5. The CCR register can only be set when I2C is turned off (PE=0); 6. fCK should be an integer multiple of 10MHz so that a fast 400kHz clock can be generated correctly.

## 27.6.9 TRISE Register (I2C\_TRISE)

Address offset: 0x20

Reset value: 0x0002

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

Bit	notation	clarification
15:6	Reserved	Reserved bit, hardware forced to 0
5:0	TRISE[5:0]	TRISE[5:0]: Maximum rise time in Fast/Standard mode (Master mode) These bits must be set to the maximum SCL rise time given in the I2C bus specification, with a growth step of one. For example, the maximum allowable SCL rise time in standard mode is 1000ns. if the value in FREQ[5:0] in the I2C_CR2 register is equal to 0x08 and TPCLK1=125ns, then 09h must be written in TRISE[5:0] (1000ns/125ns=8+1). Filter values can also be added within TRISE[5:0]. If the result is not an integer, the integer portion is written to TRISE[5:0] to secure the THIGH parameter. Note: TRISE[5:0] can only be set when I2C is disabled (PE=0).

## 27.6.10 I2C Register Address Map

Table144 I2C Register Address Maps and Reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																															
0x00	I2C_CR1	Reserved																0	SWRST	Reserved	0	ALERT	0	PEC	0	POS	0	ACK	0	STOP	0	START	0	NOSTRETCH	0	ENG	0	ENPEC	0	ENARP	0	SMBTYPE	Reserved	0	SMBUS	0	PE	0																
	Reset value																	0			0		0		0		0		0		0		0		0		0		0		0		0		0																			
0x04	I2C_CR2	Reserved																				0	LAST	0	DMAEN	0	ITBUFEN	0	ITEVTEN	0	ITERREN	0	Reserved	FREQ[5:0]										0																				
	Reset value																					0		0		0		0		0		0		0		0		0		0		0		0		0																		
0x08	I2C_OAR1	Reserved																0	ADDMODE	Reserved						0		ADD [9:8]	ADD[7:1]										0	ADD0																								
	Reset value																	0								0		0		0		0		0		0		0		0		0		0		0																		
0x0C	I2C_OAR2	Reserved																										ADD2[7:1]										0		ENDUAL																								
	Reset value																																					0		0		0		0		0		0		0		0		0										
0x10	I2C_DR	Reserved																										DR[7:0]										0																										
	Reset value																																					0		0		0		0		0		0		0		0		0										
0x14	I2C_SR1	Reserved																0	SMBALERT	0	TIMEOUT	Reserved	0	PECERR	0	OVR	0	AF	0	ARLO	0	BERR	0	TxE	0	RxNE	0	Reserved	0	STOPF	0	ADD10	0	BTF	0	ADDR	0	SB	0															
	Reset value																	0		0			0		0		0		0		0		0		0		0		0		0		0		0		0																	
0x18	I2C_SR2	Reserved																PEC[7:0]										DUALF										0	SMBHOST	0	SMBDEFAU	0	GENCALL	0	Reserved	0	TRA	0	BUSY	0	MSL	0												
	Reset value																											0										0		0		0		0		0		0		0		0		0		0		0						
0x1C	I2C_CCR	Reserved																0	F/S	0	DUTY	Reserved	CCR[11:0]																																									
	Reset value																	0		0			0		0		0		0		0		0		0		0		0		0		0		0		0		0															
0x20	I2C_TRISE	Reserved																										TRISE[5:0]																																				
	Reset value																																					0										0		0		0		0		0		0		0		0		0

SeeTable1 for register start addresses.

## 28 Universal Synchronous Asynchronous Transceiver (USART)

### 28.1 Introduction to USART

The Universal Synchronous Asynchronous Transceiver (USART) provides a flexible method of exchanging full-duplex data with external devices using the industry-standard NRZ asynchronous serial data format. The USART utilizes a fractional baud rate generator to provide a wide range of baud rate options.

It supports synchronous unidirectional and half-duplex single-wire communications, as well as LIN (Local Interconnect Network), smart card protocols and IrDA (Infrared Data Organization) SIRENDEC specifications, and modem (CTS/RTS) operation. It also allows multiprocessor communication.

High-speed data communication can be realized by using the DMA method with multi-buffer configuration.

### 28.2 USART Key Features

- Full-duplex, asynchronous communication
- NRZ standard format
- Fractional Baud Rate Generator System
  - Programmable baud rate up to 13.5Mbits/s common to transmit and receive
- Programmable data word length (8 or 9 bits)
- Configurable stop bits - supports 1 or 2 stop bits
- Ability for the LIN master to send synchronized disconnect symbols and for the LIN slave to detect disconnect symbols
  - Generates 13-bit disconnect when USART hardware is configured for LIN; detects 10/11-bit disconnects
- The sender provides the clock for the synchronized transmission
- IRDASIR Encoder Decoder
  - Supports 3/16-bit duration in normal mode
- Smart Card Analog Function
  - The smart card interface supports the asynchronous smart card protocol defined in the ISO7816-3 standard.
  - 0.5 and 1.5 stop bits for smart cards
- single-line half-duplex communication
- Configurable multi-buffer communication using DMA
  - Receive/send bytes in SRAM using centralized DMA buffering
- Separate transmitter and receiver enable bits
- Flagging
  - Receive buffer full
  - Send buffer empty
  - End-of-transmission flag
- calibration control
  - transmit parity bit
  - Performs checksums on incoming data
- Four False Detection Signs
  - overflow error
  - noise error
  - frame error
  - calibration error
- 10 interrupt sources with flags
  - CTS change
  - LIN break character detection
  - Send Data Register Empty
  - Send complete
  - Receive data register full
  - Bus idle detected
  - overflow error
  - frame error
  - noise error
  - calibration error



- Multi-processor communication - silent mode if addresses do not match
- Wake-up from silent mode (via idle bus detection or address flag detection)
- Two ways to wake up the receiver: address bit (MSB, bit 9), bus idle

## 28.3 USART Function Overview

The interface is connected to other devices via three pins (seeFigure248 ). At least two pins are required for any USART bidirectional communication: receive data in (RX) and transmit data out (TX).

RX: Receive data serial transmission. Data is recovered by an oversampling technique to differentiate between data and noise.

TX: Transmit data output. When the transmitter is disabled, the output pin reverts to its I/O port configuration. When the transmitter is activated and no data is being sent, the TX pin is high. In single-wire and smart card modes, this I/O port is used for both sending and receiving data.

- The bus should be idle before transmitting or receiving
- A starting position
- One data word (8 or 9 bits) with the least significant bit first
- 0.5, 1.5, 2 stop bits, thus indicating the end of the data frame
- Using the Fractional Baud Rate Generator - 12-digit integer and 4-digit decimal representation.
- A status register (USART\_SR)
- Data register (USART\_DR)
- A baud rate register (USART\_BRR), 12-bit integer and 4 decimal bits
- A smart card mode guard time register (USART\_GTPR)

For specific definitions of each bit in the above registers, refer to the Register Description Section28.6 : USART Register Description. The following pins are required in synchronization mode:

- CK: Transmitter clock output. This pin outputs the clock used to synchronize the transmission, (there is no clock pulse on the Start and Stop bits, software optionally, a clock pulse can be sent on 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). The clock phase and polarity are software programmable. In smart card mode, the CK can provide the clock for the smart card. The following pins are required in IrDA mode:

- IrDA\_RDI: Data input in IrDA mode.
- IrDA\_TDO: Data output in IrDA mode. The following pins are required in hardware flow control mode:
- nCTS:Clear Transmit, if high, blocks the next data transmission at the end of the current data transmission.
- nRTS: Request to Send, if low, indicates that the USART is ready to receive data

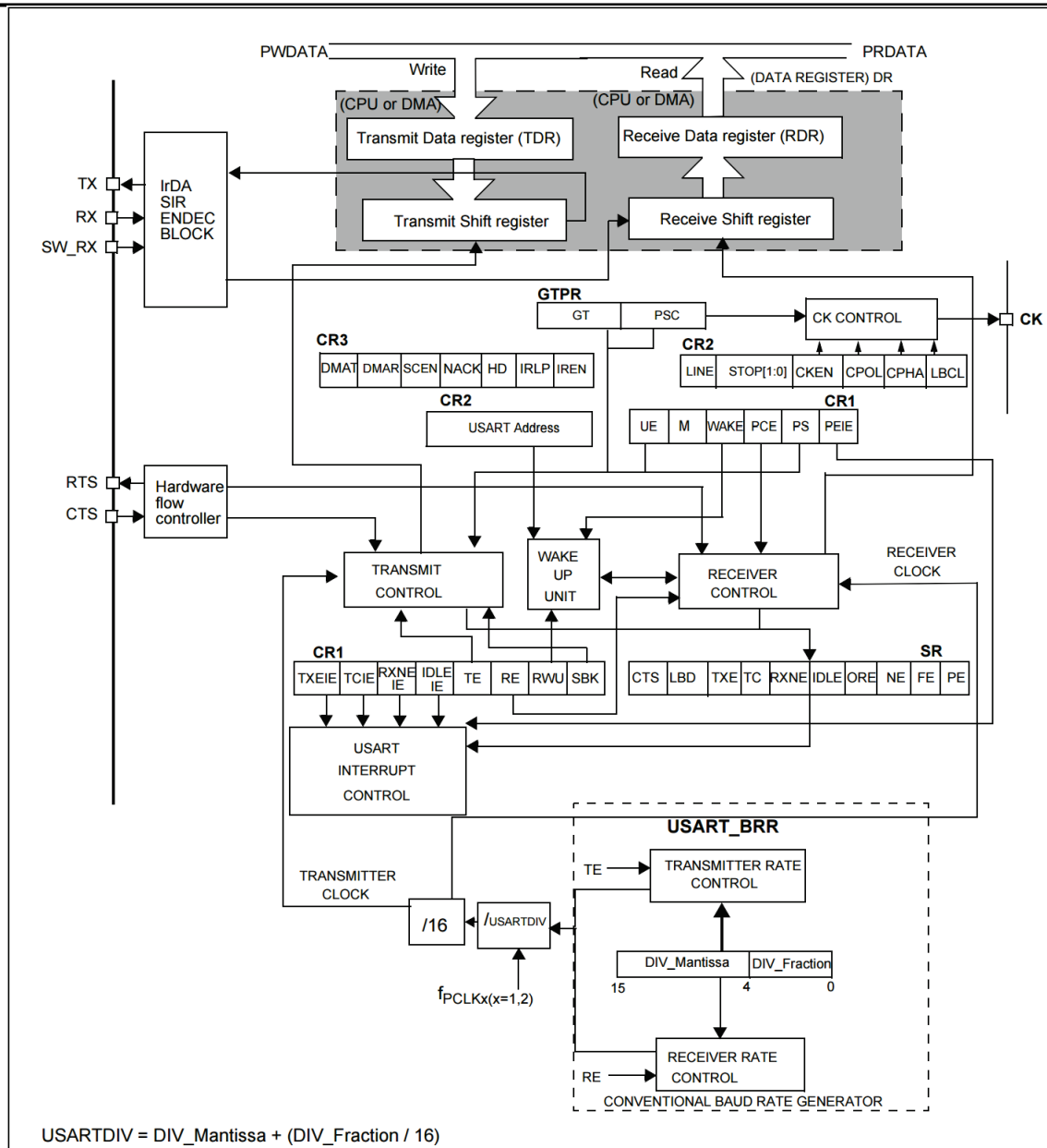


Figure248 USART Block Diagram

### 28.3.1 USART Characterization

The word length can be selected as 8 or 9 bits by programming the M bit in the USART\_CR1 register (see Figure 249). The TX pin is low during the start bit and high during the stop bit. The idle symbol is considered to be a complete data frame consisting entirely of '1's followed by the start bit of the next frame containing the data (the number of bits in the '1' also includes the number of bits in the stop bit).

The break symbol is considered to be received as all '0's (including during the stop bit, which is also a '0') in one frame cycle. At the end of the break frame, the transmitter inserts 1 or 2 more stop bits ('1') to answer the start bit.

Transmission and reception are driven by a common baud rate generator, which generates separate clocks for the transmitter and receiver when their respective enable bits are set.

A detailed description of each function block will follow.

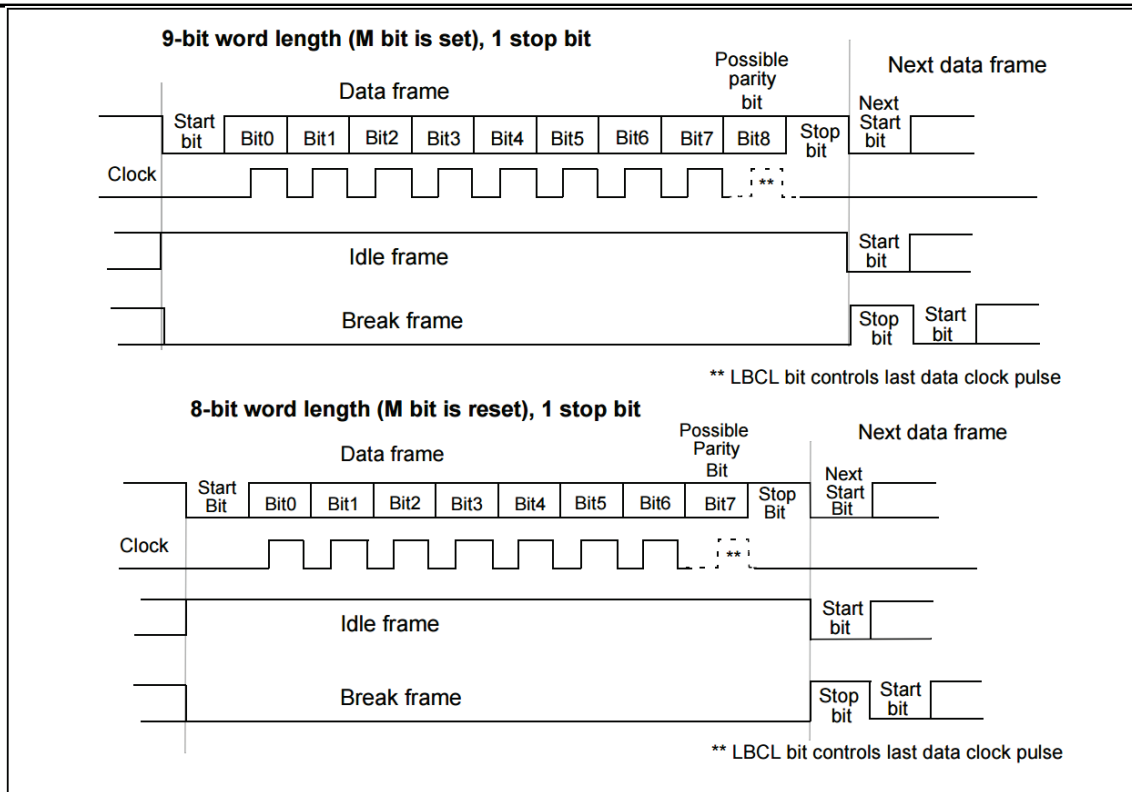


Figure249 Word length setting

## 28.3.2 Transmitters

The transmitter sends an 8-bit or 9-bit data word depending on the state of the M bit. When the transmit enable bit (TE) is set, the data in the transmit shift register is output on the TX pin and the corresponding clock pulse is output on the CK pin.

### Character Sending

During a USART transmit, the least significant bit of data is shifted first on the TX pin. In this mode, the USART\_DR register contains a buffer between the internal bus and the transmit shift register (see Figure 248).

Each character is preceded by a low start bit; this is followed by a configurable number of stop bits. The USART supports multiple stop bit configurations: 0.5, 1, 1.5, and 2 stop bits.

- Notes:
1. The TE bit cannot be reset during data transmission, otherwise the data on the TX pin will be corrupted because the baud rate counter stops counting. The current data being transferred will be lost.
  2. An idle frame is sent when the TE bit is activated.

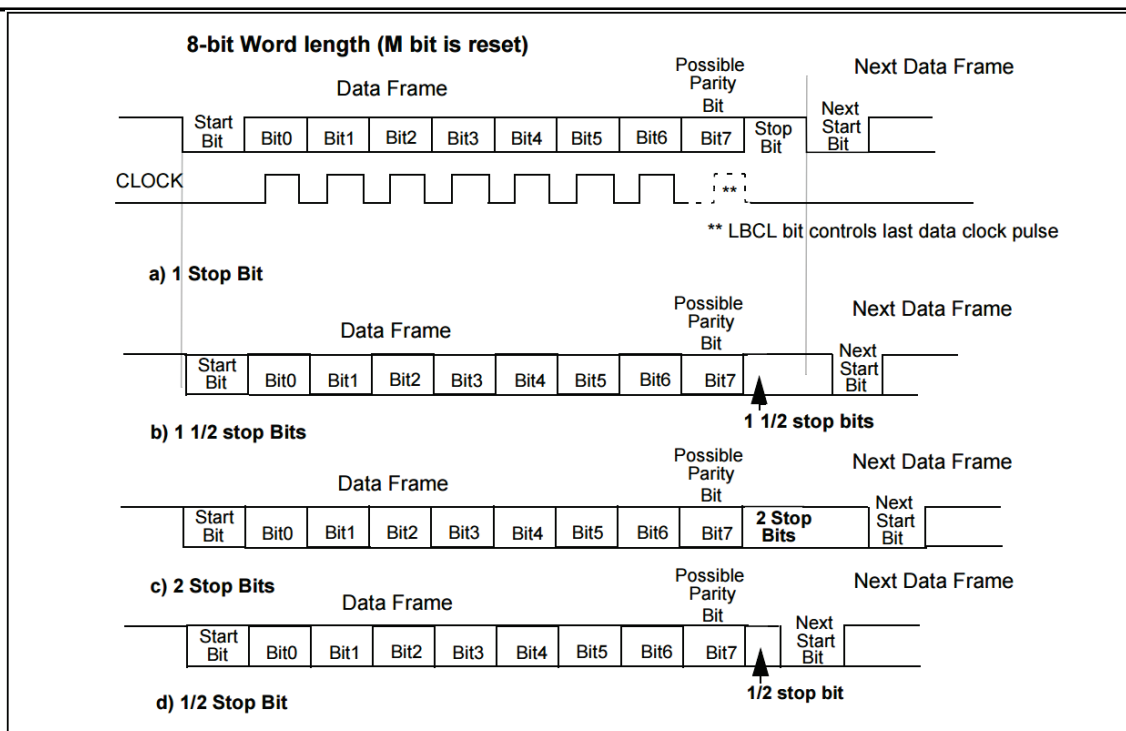
### Configurable Stop Bits

The number of bits of the stop bit sent with each character can be programmed by bits 13, 12 of control register 2.

1. 1 stop bit: default value for the number of stop bits.
2. 2 stop bits: can be used in regular USART mode, single wire mode, and modem mode.
3. 0.5 stop bits: used when receiving data in smart card mode.
4. 1.5 stop bits: used when sending and receiving data in smart card mode.

Idle frames include stop bits.

The break frame is 10 bits low followed by a stop bit (when m=0); or 11 bits low followed by a stop bit (when m=1). It is not possible to transmit longer disconnect frames (longer than 10 or 11 bits).



#### Configuration Steps:

1. Activate the USART by setting the UE bit on the USART\_CR1 register
2. Program the M bit of USART\_CR1 to define the word length.
3. Program the number of stop bits in USART\_CR2.
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. Use the USART\_BRR register to select the requested baud rate.
6. Set the TE bit in USART\_CR1 to send an idle frame as the first data transmission.
7. Write the data to be sent to the USART\_DR register (this action clears the TXE bit). Repeat step 7 for each data to be sent when there is only one buffer.
8. After writing the last data word in the USART\_DR register, wait for TC=1, which indicates the end of transmission of the last data frame. When it is necessary to turn off the USART or before it is necessary to enter the shutdown mode, it is necessary to confirm the end of the transmission to avoid destroying the last transmission.

#### Single-Byte Communication

Clearing the TXE bit is always accomplished by a write operation to the data register. the TXE bit is set by hardware and it indicates:

- Data has been shifted from the TDR to the shift register, data transmission has started
- TDR register is cleared
- The next data can be written to the USART\_DR register without overwriting the previous data If the TXEIE bit is set, this flag generates an interrupt.

If the USART is transmitting data at this time, a write operation to the USART\_DR register stores the data into the TDR register and copies that data into the shift register at the end of the current transmission.

If the USART is not transmitting data at this time and is in the idle state, a write operation to the USART\_DR register puts the data directly into the shift register, the data transfer starts, and the TXE bit is immediately set.

When a frame is sent (after the stop bit is sent) and the TXE bit is set, the TC bit is set and an interrupt is generated if the TCIE bit in the USART\_CR1 register is set.

After the last data word has been written in the USART\_DR register, you must wait for TC=1 before turning off the USART module or setting the microcontroller to enter low-power mode (see below for details).

Use the following software procedure to clear the TC bit:

1. Read the USART\_SR register once;
2. Write the USART\_DR register once.

Notes: The TC bit can also be cleared by software writing '0' to it. This clearing method is only recommended in multi-buffer communication mode.

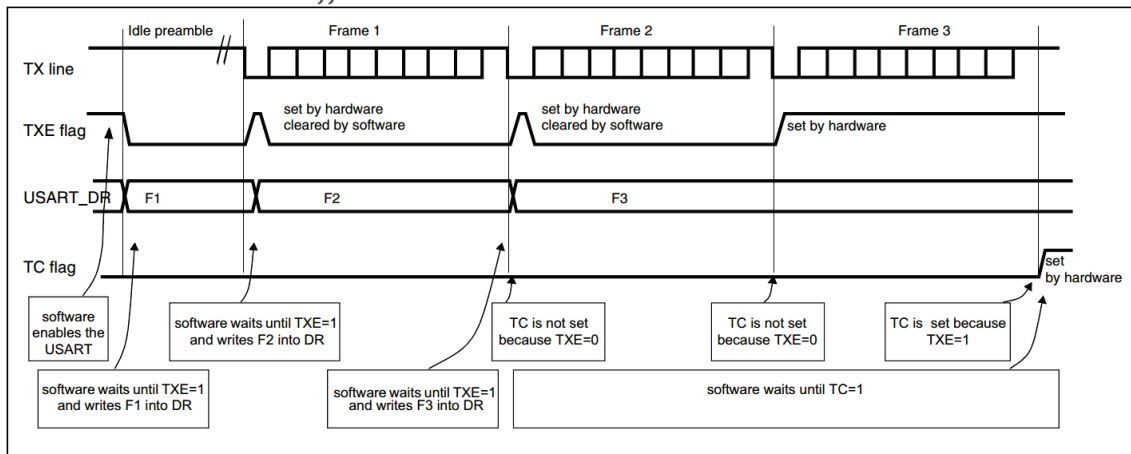


Figure251 Changes in TC/TXE when transmitting

### Disconnect Symbol

Setting SBK sends a break symbol. The disconnect frame length depends on the M-bit (see Figure 249). If SBK=1 is set, a break symbol is sent on the TX line after completion of the current data transmission. SBK is reset by hardware when the disconnect character is sent (at the stop bit of the disconnect symbol). The USART inserts a logical '1' at the end of the last disconnect frame to ensure that the start bit of the next frame is recognized.

Notes: If the software resets the SBK bit again before starting to send the disconnect frame, the disconnect symbol will not be sent. If two consecutive disconnect frames are to be sent, the SBK bit should be set after the stop bit of the previous disconnect symbol.

### Free Symbol

Setting TE will cause the USART to send an idle frame before the first data frame.

## 28.3.3 Refraction

The USART can receive 8-bit or 9-bit data words based on the M bit of USART\_CR1.

### Start Bit Detection

In USART, if a particular sample sequence is recognized, then a start bit is considered to be detected.

The sequence is: 1110X0X0X0000

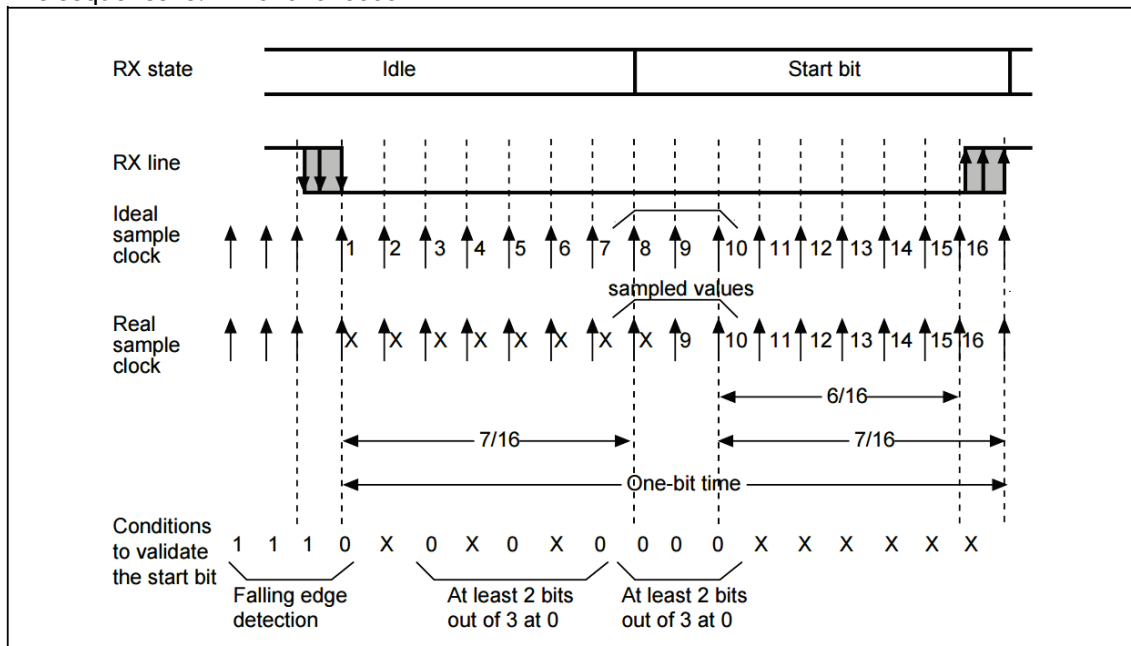


Figure252 Start Bit Detection

**Notes:** *If the sequence is incomplete, then the receiver will exit start bit detection and return to the idle state (no flag bits set) waiting for a falling edge.*  
*If all 3 samples are '0' (the first sample at bits 3, 5, and 7, and the second sample at bits 8, 9, and 10 are '0'), receipt of the start bit is acknowledged, at which time the RXNE flag bit is set, and if RXNEIE = 1, an interrupt is generated.*  
*If only 2 of the 3 samples are '0' on two occasions (the samples in bits 3, 5, and 7 and the samples in bits 8, 9, and 10), then the start bit is still valid, but the NE noise flag bit is set. If this condition cannot be met, the start bit detection process is aborted and the receiver returns to the idle state (no flag bit is set).*  
*If there is a time when only 2 of the 3 sample points are '0' (sample points in bits 3, 5, and 7 or sample points in bits 8, 9, and 10), then the start bit is still valid, but the NE noise flag bit is set.*

### Character Reception

During USART reception, the least significant bit of data is first shifted in 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 UE of the USART\_CR1 register to 1 to activate the USART.
2. Programming the M bit of USART\_CR1 defines the word length
3. Number of stop bits to write in USART\_CR2
4. If multi-buffer communication is required, 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. Sets the RE bit of USART\_CR1. Activates the receiver so that it starts looking for the start bit. When a character is received, the
  - The RXNE bit is set. It indicates that the contents of the shift register are transferred to RDR. in other words, the data has been received and can be read (including the error flag associated with it).
  - If the RXNEIE bit is set, an interrupt is generated.
  - If a framing error, noise, or overflow error is detected during reception, the error flag will be set
  - In multi-buffer communication, RXNE is set up after each byte is received and cleared by a DMA read operation to the data register.
  - In single buffer mode, clearing of the RXNE bit is accomplished by software reading the USART\_DR register. The RXNE flag can also be cleared by writing a 0 to it. The RXNE bit must be cleared before the end of the next character reception to avoid overflow errors.

**Notes:** *The RE bit should not be reset during data reception. If the RE bit is cleared on reception, reception of the current byte is lost.*

### Disconnect Symbol

When a disconnect frame is received, the USART handles it like a frame error.

### Free Symbol

When an idle frame is detected, the procedure is the same as if a normal data frame had been received, but an interrupt is generated if the IDLEIE bit is set.

### Overflow Error

If RXNE has not been reset and another character is received, an overflow error occurs. Data can be transferred from the shift register to the RDR register only after the RXNE bit has been cleared. the RXNE flag is set after each byte is received. The RXNE flag remains set if the next data has been received or if a previous DMA request has not yet been serviced, and an overflow error is generated.

When an overflow error is generated:

- The ORE bit is set.
- The RDR contents will not be lost. Reading the USART\_DR register will still yield the previous data.
- The previous contents of the shift register will be overwritten. Any subsequent data received will be lost.
- If the RXNEIE bit is set or both the EIE and DMAR bits are set, an interrupt is generated.

- Sequentially performs a read operation of the USART\_SR and USART\_DR registers, which can reset the ORE bit

*Notes: When the ORE position bit is present, it indicates that at least 1 piece of data has been lost. There are two possibilities:*

- If RXNE=1, the last valid data is still on the receive register RDR and can be read out.
- If RXNE=0, this means that the last valid data has been read away and there is nothing left to read in the RDR. Such a situation can occur when the last valid data is read in the RDR while new (i.e., lost) data is received. This can also occur when new data is received during a read sequence (between a USART\_SR register read access and a USART\_DR read access).

### Noise Error

Uses oversampling techniques (except in synchronized mode) to perform data recovery by distinguishing valid input data from noise.

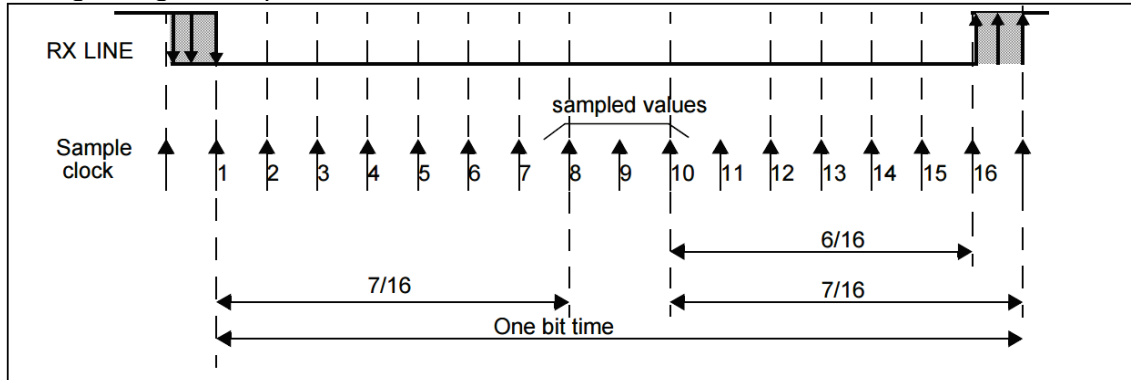


Figure253 Data sampling for noise detection

Table145 Data sampling for detection of noise

sampling value	NE status	received bit	Data validity
000	0	0	efficiently
001	1	0	null
010	1	0	null
011	1	1	null
100	1	0	null
101	1	1	null
110	1	1	null
111	0	1	efficiently

When noise is detected in a received frame:

- Set the NE flag on the rising edge of the RXNE bit.
- Invalid data is transferred from the shift register to the USART\_DR register.
- In the case of single byte communication, no interrupt is generated. However, since the NE flag bit and the RXNE flag bit are set at the same time, RXNE will generate an interrupt. In the case of multi-buffer communication, if the EIE bit in the USART\_CR3 register has been set, an interrupt will be generated.

Reading the USART\_SR first and then the USART\_DR register will clear the NE flag bit

### Frame Error

A frame error is detected when the following occurs:

Stop bits are not picked up and received recognized at the expected time due to not being synchronized on or a lot of noise.

When a frame error is detected:

- FE bit is set by hardware
- Invalid data is transferred from the shift register to the USART\_DR register.
- During single-byte communication, no interrupt is generated. However, this bit and the RXNE bit are set at the same time, and the latter will generate an interrupt. In the case of multi-buffer communication, an interrupt is generated if the EIE bit in the USART\_CR3 register is set.



Sequential execution of read operations to the USART\_SR and USART\_DR registers can reset the FE bit.

### Configurable Stop Bits During Reception

The number of stop bits to be received can be configured by the control bits in control register 2, which can be 1 or 2 in normal mode, and may be 0.5 or 1.5 in smart card mode.

1. 0.5 stop bits (reception in smart card mode): 0.5 stop bits are not sampled. Therefore, frame errors and break frames cannot be detected if 0.5 stop bits are selected.
2. 1 stop bit: Sampling of 1 stop bit is performed on the 8th, 9th and 10th sampling points.
3. 1.5 stop bits (smart card mode): when transmitting in smart card mode, the device has to check that the data has been sent correctly. So the receiver function block must be activated (RE=1 in the USART\_CR1 register) and sample the signal on the data line during the sending of the stop bit. If a checksum error occurs, the smart card indicates a framing error by pulling down the data line when the transmitter samples the NACK signal, i.e., during the time corresponding to the stop bit on the bus. FE is set at the end of the 1.5 stop bits together with RXNE. The 1.5 stop bits are sampled at sample points 16, 17, and 18. 1.5 stop bits can be divided into 2 parts: a 0.5 clock cycle during which nothing is done. This is followed by 1 clock cycle of stop bits, sampled at the midpoint of this period. See Section 28.3.11 : Smart Cards for details.
4. 2 Stop Bits: Sampling of the 2 stop bits is done at the 8th, 9th and 10th sample points of the first stop bit. If a frame error is detected during the first stop bit, the frame error flag is set. The second stop bit is no longer checked for frame errors. The RXNE flag will be set at the end of the first stop bit.

## 28.3.4 Fractional Baud Rate Generation

The baud rates of the receiver and transmitter should be set to the same values in the integer and decimal registers of the USARTDIV.

$$\text{Tx/Rx baud} = \frac{f_{\text{ck}}}{(16 * \text{USARTDIV})}$$

Here fCK is the clock for the peripherals (PCLK1 for USART2, 3, 4, 5 and PCLK2 for USART1) USARTDIV is an unsigned fixed-point number. The value of these 12 bits is set in the USART\_BRR register.

*Notes: After writing USART\_BRR, the baud rate counter is replaced by the new value of the baud rate register. Therefore, do not change the value of the baud rate register while communication is in progress.*

### How to value from USART\_BRR register to USARTDIV

#### Example 1:

If DIV\_Mantissa=27, DIV\_Fraction=12 (USART\_BRR=0x1BC),  
the as a result  
Mantissa(USARTDIV)=27  
Fraction(USARTDIV)=12/16=0.75  
So USARTDIV = 27.75

#### Example 2:

Requires USARTDIV=25.62, the  
There it is:  
DIV\_Fraction=16\*0.62=9.92  
The closest integer is: 10 = 0x0A  
DIV\_Mantissa=mantissa(25.620)=25=0x19  
So, USART\_BRR = 0x19A therefore USARTDIV = 25.625

#### Example 3:

Requires USARTDIV=50.99  
There it is:  
DIV\_Fraction=16\*0.99=15.84  
The closest integer is: 16=0x10=>DIV\_frac[3:0] overflow=> The rounding must be added to the decimal part  
DIV\_Mantissa=mantissa(50.990+feed)=51=0x33



So: USART\_BRR=0x330, USARTDIV=51.000

Table146 Error calculation when setting baud rate

baud		fPCLK=36MHz			fPCLK=72MHz		
serial number	Kbps	practice	Value placed in baud rate register	% Error	practice	Value placed in baud rate register	% 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	not able	not able	not able	4500	1	0%

Notes: 1. The lower the clock frequency of the CPU, the lower the error for a particular baud rate. The upper limit of the achievable baud rate can be determined from this set of numbers According to get.

2. Only USART1 uses PCLK2 (up to 216MHz). Other USARTs use PCLK1 (up to 108MHz).

### 28.3.5 USART Receiver Tolerates Clock Changes

The USART Asynchronous Receiver will only operate properly if the overall clock system ground variations are less than what the USART Asynchronous Receiver can tolerate. Factors that affect these variations are:

- DTRA: variations due to transmitter errors (including variations in the oscillator at the transmitter side)
- DQUANT: Error due to baud rate rounding at receiver side
- DREC: Receiver-side oscillator change
- DTCL: Variation due to the transmission line (usually due to inconsistency between the transceiver's low-to-high transition timing and the high-to-low transition timing).

Need to meet: DTRA+DQUANT+DREC+DTCL<USART receiver tolerance

For normal reception of data, the tolerance of the USART receiver is equal to the maximum tolerable variation, which depends on the following selection:

- 10 or 11-bit character length defined by bit M of the USART\_CR1 register
- Whether to use fractional baud rate generation

Table147 USART receiver tolerance when DIV\_Fraction=0

M position	Thinking NF is wrong	Don't think NF is wrong
0	3.75%	4.375%
1	3.41%	3.97%

Table148 USART receiver tolerance when DIV\_Fraction!=0

M position	Thinking NF is wrong	Don't think NF is wrong
0	3.33%	3.88%
1	3.03%	3.53%

Notes: In special cases, i.e., when the received frame contains some idle frames that are exactly 10 bits at M=0 (11 bits at M=1), the data in the 2 tables above may be slightly different.

### 28.3.6 Multiprocessor Communication

Multi-processor communication (connecting several USARTs in a network) can be realized through USART. For example, a USART device may be the master, and its TX output is connected to the RX inputs of the other USART slave devices; the TX outputs of the respective USART slave devices are logically coupled together and connected to the RX inputs of the master device.

In multiprocessor configurations, we typically want only addressed receivers to be activated to receive subsequent data, which reduces the excess USART service overhead introduced by the involvement of unaddressed receivers.

Unaddressed devices can be placed in silent mode by enabling their silent function. In silent mode:

- Any receive status bits are not set.
- All receive interrupts are disabled.
- The RWU bit in the USART\_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 USART\_CR1 register, the USART 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.

#### Idle bus detection (WAKE=0)

When the RWU bit is written 1, the USART enters silent mode. It is woken up when an idle frame is detected, and the RWU is then cleared by hardware, but the IDLE bit in the USART\_SR register is not reset. The RWU is then cleared to zero by hardware, but the IDLE bit in the USART\_SR register is not set; the RWU can also be written to 0 by software. The following figure gives an example of using idle bus detection to wake up and enter silent mode

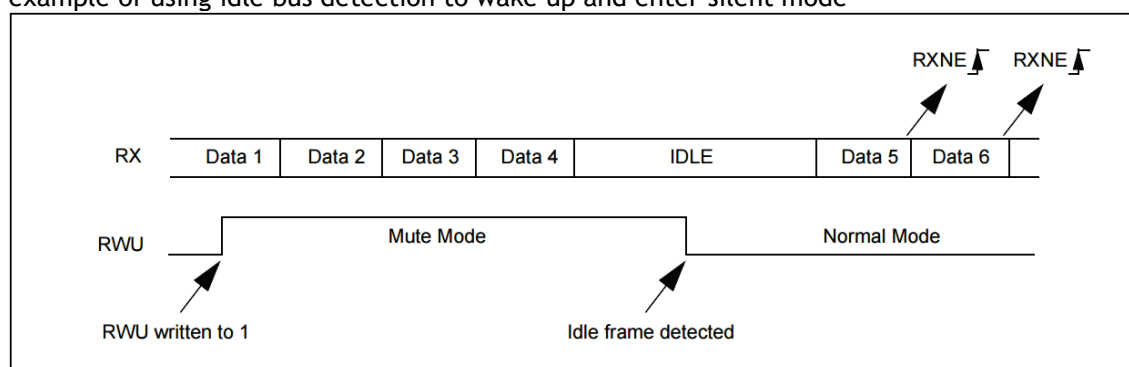


Figure254 Silent mode utilizing idle bus detection

#### Addressmark Detection (WAKE=1)

In this mode, if the MSB is 1, the byte is considered as an address, otherwise it is considered as data. In an address byte, the address of the target receiver is placed in 4 LSBs. This 4-bit address is compared by the receiver to its own address, which is programmed in the ADDR of the USART\_CR2 register.

The USART enters silent mode if the received byte does not match its programmed address. At this point, the hardware sets the RWU bit. Receiving this byte neither sets the RXNE flag nor generates an interrupt or issues a DMA request because the USART is already in silent mode.

When the received byte matches the programmed address within the receiver, the USART exits silent mode. The RWU bit is then cleared and the subsequent byte is received normally. Receipt of this matching address byte will set the RXNE bit as the RWU bit is cleared.

When the receive buffer does not contain data (RXNE=0 for USART\_SR), the RWU bit can be written 0 or 1. Otherwise, this write operation is ignored. The following figure gives an example of using address marker detection to wake up and enter silent mode.

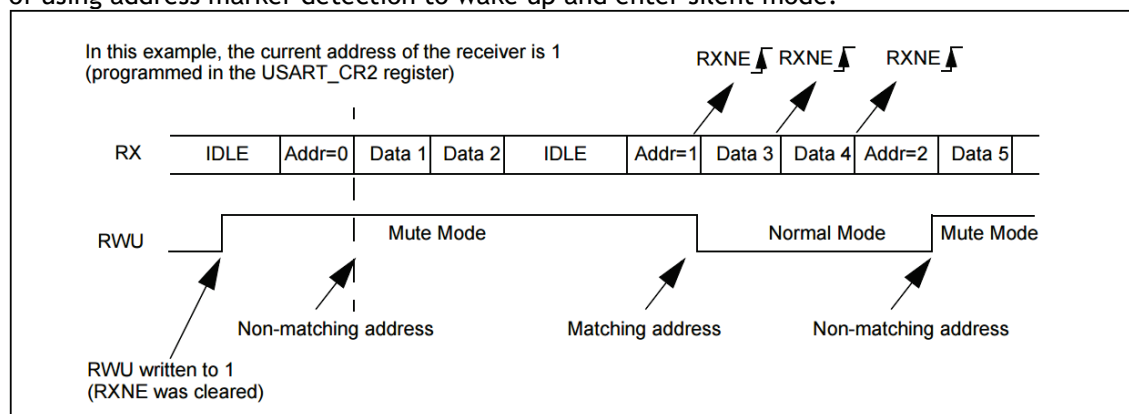


Figure255 Silent mode using address tag detection

## 28.3.7 Calibration Control

Setting the PCE bit on the USART\_CR1 register enables parity control (generates a parity bit on transmit and parity check on receive). The possible USART frame formats, based on the frame length defined by the M bit, are listed in the table below.

Table149 Frame format

M position	PCE level	USART frame
0	0	Start Bit   8-bit Data   Stop Bit
0	1	Start   7-bit data   Parity check bit   Stop bit
1	0	Start Bit   9 bits of data   Stop Bit
1	1	Start   8-bit data   Parity check bit   Stop bit

**Notes:** When waking up a device with an address marker, the address is matched taking into account only the MSB bits of the data, without concern for the parity bits. (The MSB is the number of (the last of the data bits to be sent, followed immediately by a parity or stop bit)

Even parity: the parity bit makes the number of 7 or 8 LSB data in a frame and the number of '1's in the parity bit even.

Example: data = 00110101 with 4 '1's, if even parity is selected (PS = 0 in USART\_CR1) the parity bit will be '0'.

Odd parity: This parity bit makes the number of 7 or 8 LSB data in a frame and the number of '1's in the parity bit odd.

Example: data = 00110101 with 4 '1's, if odd parity is selected (PS = 1 in USART\_CR1), the parity bit will be a '1'.

Transmission mode: If the PCE bit of USART\_CR1 is set, the MSB bit of the data written to the data register is replaced by the parity bit and sent out (even parity even '1' if selected, odd parity odd '1' if selected). If parity fails, the PE flag in the USART\_SR register is set to '1' and an interrupt is generated if the PEIE in the USART\_CR1 register is pre-set.

## 28.3.8 LIN (Local Area Network) Mode

LIN mode is selected by setting the LINEN bit in the USART\_CR2 register. In LIN mode, the lower column bit must be held at 0:

- CLKEN bit of the USART\_CR2 register
- STOP[1:0], SCEN, HDSEL, and IREN of USART\_CR3 registers

### LIN Transmission

28.3.2The same steps described in the section apply to LIN master transmissions with the following differences from normal USART transmissions:

- Clear the M bit to configure an 8-bit word length
- Set the LINEN bit to enter LIN mode. At this point, setting SBK will send 13 bits of '0' as a disconnect symbol. A bit '1' is then sent to allow detection of the next start bit.

### LIN Reception

When LIN mode is enabled, the disconnect symbol detection circuit is activated. This detection is completely independent of the USART receiver. The disconnect symbol is detected as soon as it appears, either when the bus is idle or during the sending of a data frame, which has not yet been completed, and the sending of the disconnect symbol is inserted.

When the receiver is activated (RE=1 for USART\_CR1), the circuit monitors the start signal on RX. The method of monitoring the start bit is the same as detecting the break symbol or data. When the start bit is detected, the circuit samples each of the next bits at the 8th, 9th, and 10th oversampling clock points of each bit. If 10 (when LBDL of USART\_CR2 = 0) or 11 (when LBDL of USART\_CR2 = 1) consecutive bits are '0' and followed by a delimiter, the LBD flag of USART\_SR is set. If the LBDIE bit = 1, an interrupt is generated. Check the delimiter before acknowledging the disconnect symbol, as it implies that the RX line has returned to high.

If a '1' is sampled before the 10th or 11th sample point, the detection circuit cancels the current detection and re-searches for the start bit. If LIN mode is disabled, the receiver continues to operate as a normal USART without regard to detecting the disconnect symbol.

If LIN mode is not activated (LINEN=0), the receiver still operates normally in USART mode and does not perform a disconnect detection. If LIN mode is activated (LINEN=1), the receiver stops as soon as a framing error occurs (i.e., a stop bit detects a '0', which occurs in a break frame)

until the break symbol detection circuitry receives a '1' (which occurs when the break symbol has not been sent out in its entirety), or a delimiter (which occurs when a complete break symbol has been detected) .

Figure256 illustrates the behavior of the disconnected symbol detector state machine in relation to the disconnected symbol flag.

Figure257 gives an example of a disconnected frame.

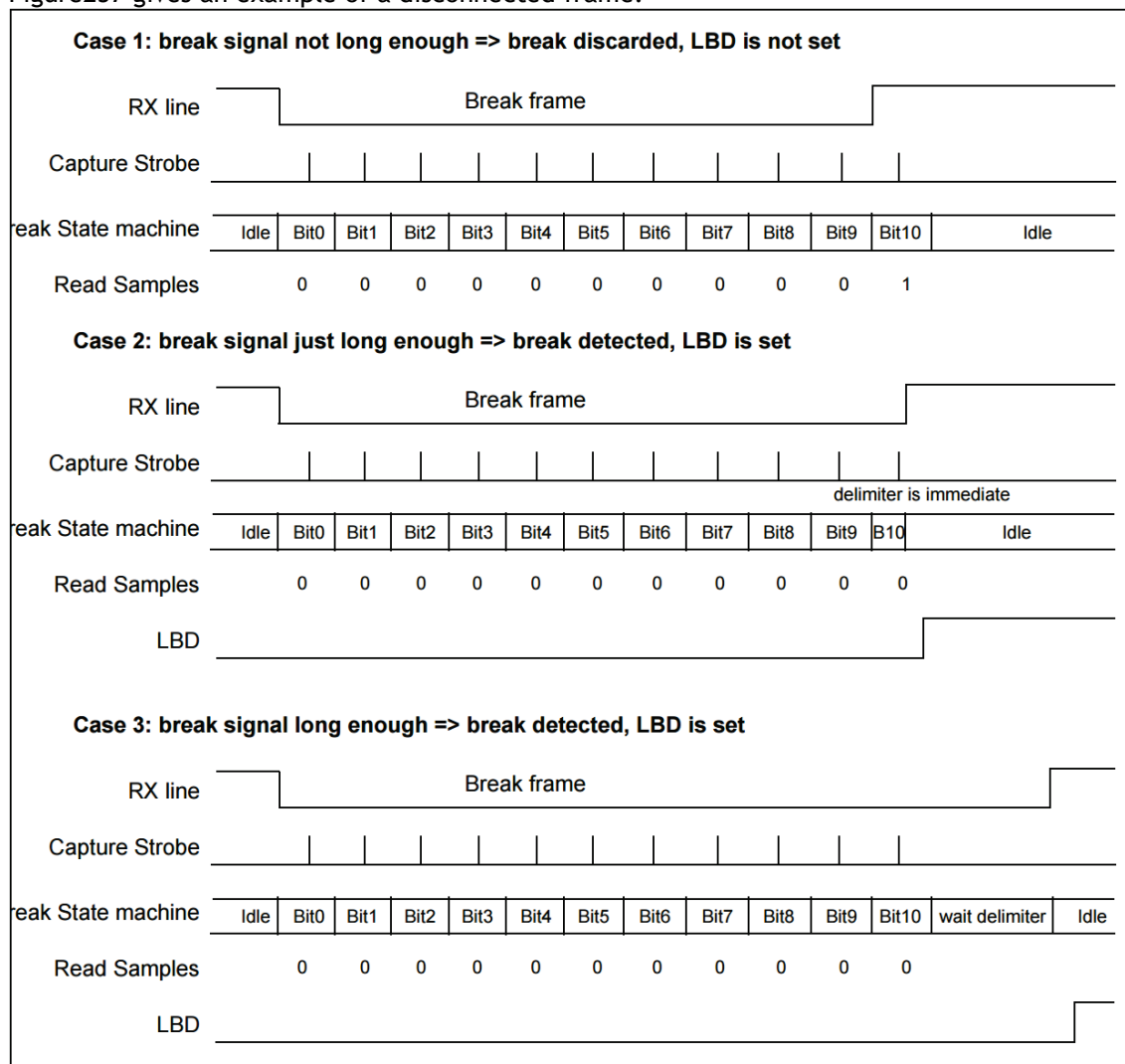


Figure256 Disconnect detection in LIN mode (11-bit disconnect length - LBDL bit set)

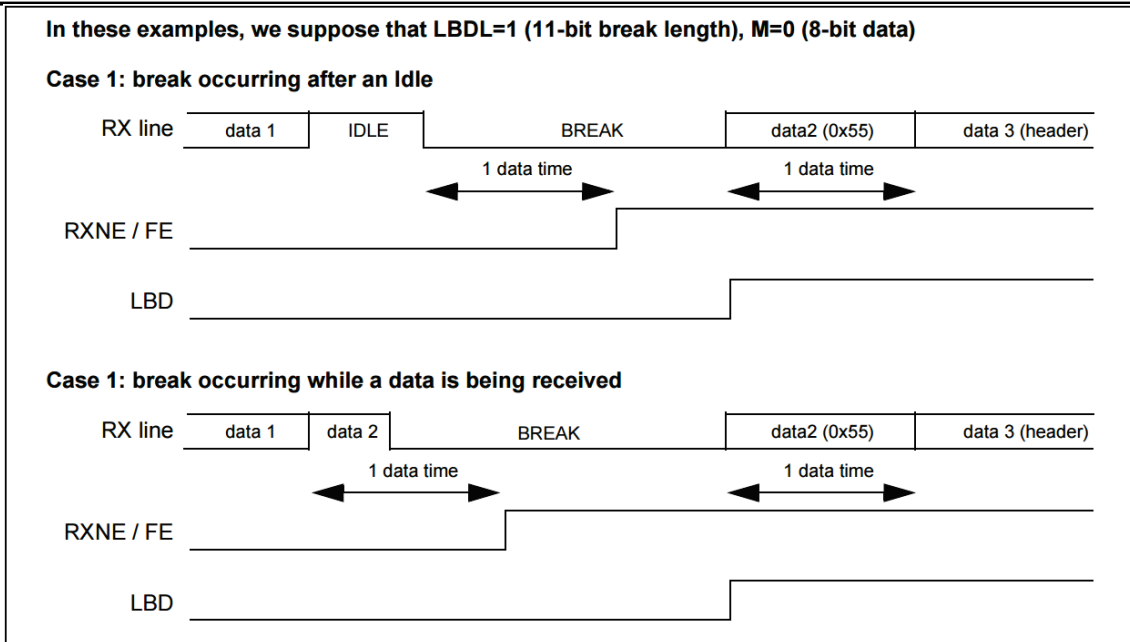


Figure257 Disconnect detection in LIN mode with frame error detection

## 28.3.9 USART Synchronization Mode

Synchronization mode is selected by writing the CLKEN bit on the USART\_CR2 register

In synchronized mode, the lower column must be kept clear:

- LINEN bit in the USART\_CR2 register
- SCEN, HDSEL, and IREN bits in the USART\_CR3 register

USART allows the user to control bi-directional synchronous serial communications in a master mode manner. the CK pin is the output of the USART transmitter clock. There are no clock pulses on the CK pin during the start and stop bits. Depending on the state of the LBCL bit in the USART\_CR2 register, it is determined that a clock pulse is 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 on the USART\_CR2 register allows the user to select the phase of the external clock (see Figure258, Figure259 and Figure260).

The external CK clock is not activated during the bus idle period, before the actual data arrives and when the disconnect symbol is sent.

In synchronous mode, the USART transmitter works exactly the same as in asynchronous mode. However, since CK is synchronized with TX (according to CPOL and CPHA), the data on TX is sent synchronously with CK.

The synchronous mode USART receiver works differently than the asynchronous mode. If RE=1, the data is sampled on CK (on the rising or falling edge depending on CPOL and CPHA) without any oversampling. However, the build-up time and duration (depending on the baud rate, 1/16 bit time) must be taken into account.

- Notes:
1. The CK pin works jointly with the TX pin. Thus, the clock is only provided when the transmitter is enabled (TE = 1) and data is sent (written to the USART\_DR register). This means that it is not possible to receive a synchronized data when no data is sent.
  2. The LBCL, CPOL and CPHA bits should be properly configured when both the transmitter and receiver are disabled; these bits cannot be changed when the transmitter or receiver is enabled.
  3. It is recommended to set TE and RE in the same instruction to minimize the receiver setup time and hold time.
  4. USART only supports master mode: it cannot receive or send data with an input clock from another device (CK is always an output).

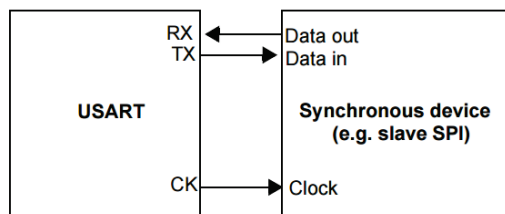


Figure258 Example of a synchronized USART transmission

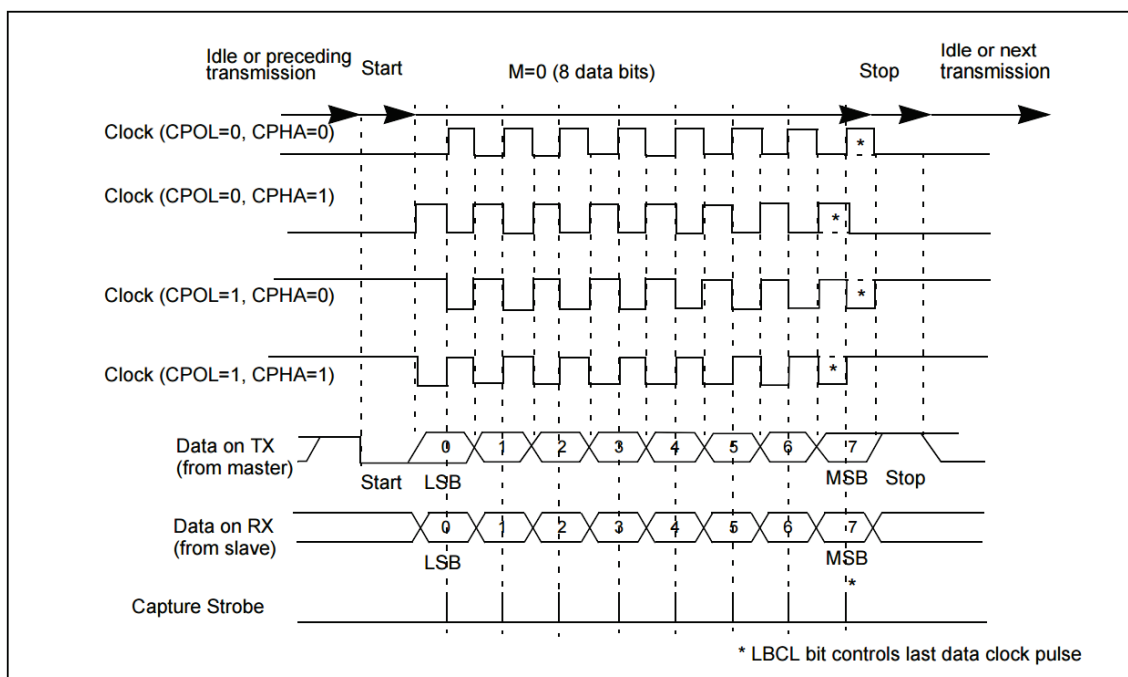


Figure259 USART Data Clock Timing Example (M=0)

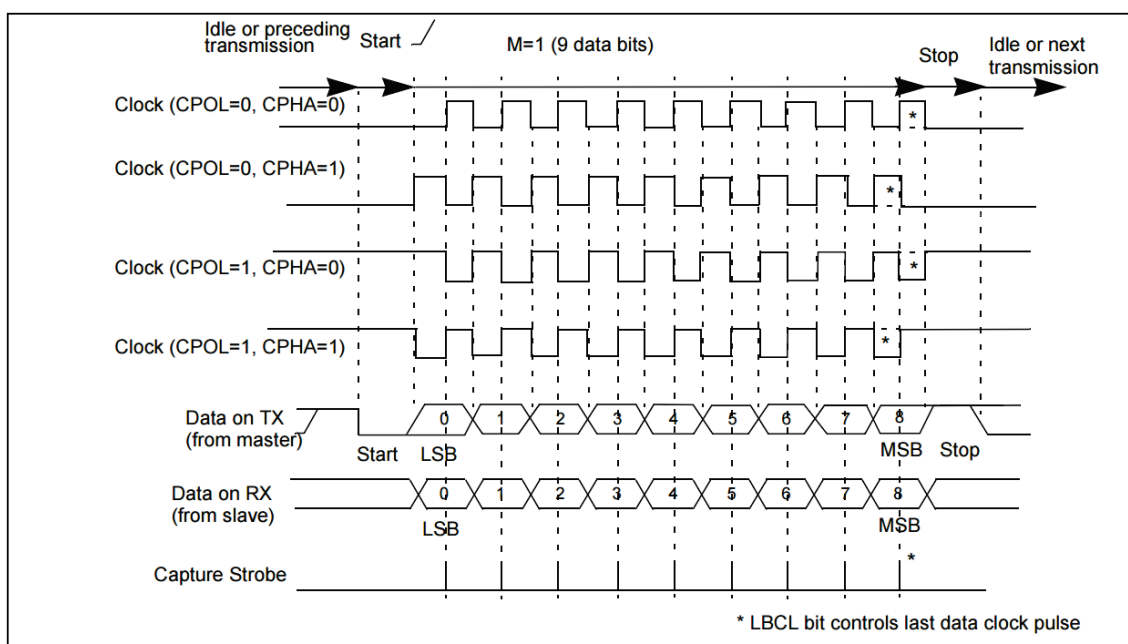


Figure260 USART Data Clock Timing Example (M=1)

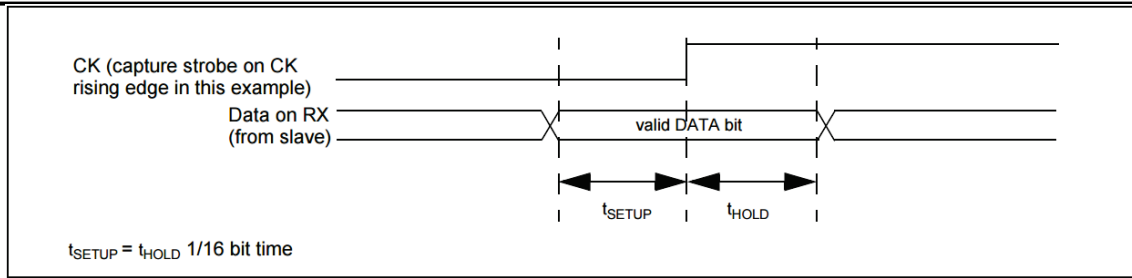


Figure261 RX Data Sample/Hold Time

Notes: CK functions differently in Smart Card Mode, refer to the Smart Card Mode section for details.

### 28.3.10 Single-Line Half-Duplex Communication

Single-wire, half-dual mode is selected by setting the HDSEL bit in the USART\_CR3 register. In this mode, the following bits must remain clear:

- LINEN and CLKEN bits of the USART\_CR2 register
- SCEN and IREN bits of the USART\_CR3 register

The USART can be configured to follow a single-wire half-duplex protocol. In single-wire half-duplex mode, the TX and RX pins are interconnected within the chip. Half-duplex and full-duplex communication is selected using the control bit "HALF DUPLEX SEL" (HDSEL bit in USART\_CR3). When HDSEL is '1'

- RX is no longer used
- TX is always released when there is no data transfer. Therefore, it behaves as a standard I/O port in the idle or receive state. This means that this I/O must be configured as a dangling input (or open-drain output high) when not being driven by the USART.

Other than this, communication is similar to normal USART mode. Conflicts on the line are managed by software (e.g., through the use of a central arbiter). In particular, transmissions are never blocked by hardware. When the TE bit is set, the transmission continues as soon as the data is written to the data register.

### 28.3.11 Smart Card

Setting the SCEN bit of the USART\_CR3 register selects smart card mode. In smart card mode, the lower column bits must be kept clear:

- LINEN bit of the USART\_CR2 register
- HDSEL bit and IREN bit of the USART\_CR3 register

In addition, the CLKEN bit can be set to provide a clock to the smart card.

The interface is compliant with the ISO7816-3 standard and supports the smart card asynchronous protocol. the USART should be set to:

- 8-bit data bit plus parity bit: M=1, PCE=1 in USART\_CR1 register at this time
- 1.5 stop bits for transmit and receive: i.e., STOP=11 in the USART\_CR2 register

Notes: It is also possible to select 0.5 stop bits on receive, but to avoid switching between the 2 configurations, it is recommended to use 1.5 stop bits on transmit and receive.

The example given below illustrates the signaling on the data line, both with and without a checksum error.



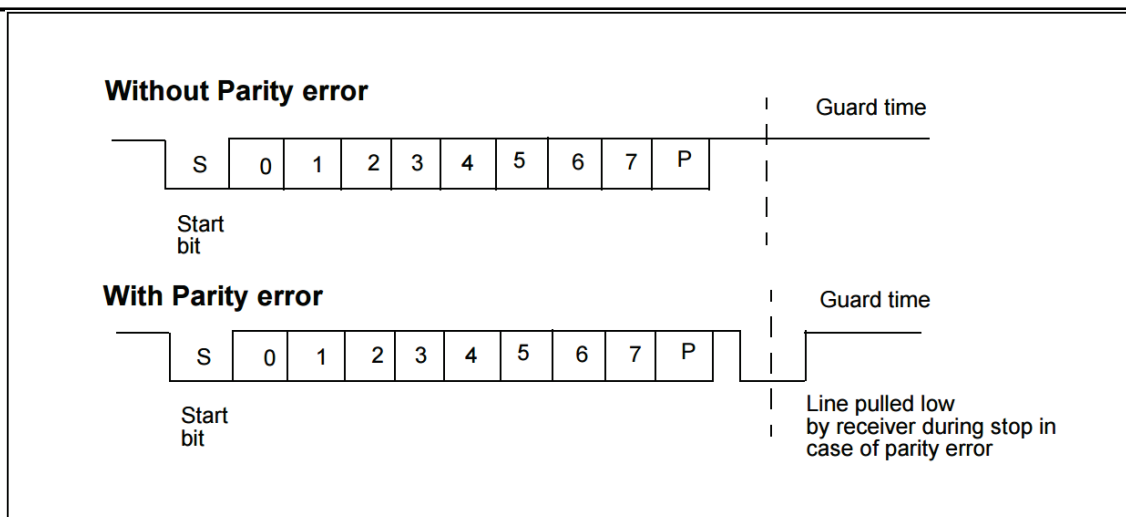


Figure262 ISO7816-3 asynchronous protocols

When connected to a smart card, the TX of the USART drives a bi-directional wire that the smart card also drives. To do this, SW\_RX must be connected to the same I/O port as TX. The transmitter's output enable bit, TX\_EN, is set during the sending of the start bit and the data byte, and is released (weakly pulled up) during the sending of the stop bit, so that the receiver can pull the data line down if a checksum error is detected. If TX\_EN is not used, TX is pulled high during the stop bit: in that case the receiver can also drive the line as long as TX is configured as open drain.

Smart card is a single-wire, half-duplex communication protocol

- Sending data out from the transmit shift register is delayed by a minimum of 1/2 baud clock. In normal operation, a full transmit shift register will begin shifting data out on the next baud clock edge. In smart card mode, this send is delayed by 1/2 baud clock.
- If a parity error is detected during the reception of a data frame set to 0.5 or 1.5 stop bits, the transmit line is pulled down by one baud clock cycle after completing the reception of the frame (i.e., at the end of the stop bits). This is to tell the smart card that the data sent to the USART was not received correctly. This NACK signal (pulling the transmit line down one baud clock cycle) will generate a frame error on the transmit side (the transmit side is configured for 1.5 stop bits). The application program can handle resending the data according to the protocol. If the NACK control bit is set, the receiver will give a NACK signal when a checksum error occurs; otherwise no NACK is sent.
- The setting of the TC flag can be delayed by programming the protection time register. In normal operation, the TC is raised when the transmit shift register is empty and no new transmit requests are made. In smart card mode, an empty transmit shift register triggers the protection time counter to start counting up until the value in the protection time register is reached, during which time the TC is forced low. When the protection time counter reaches the value in the protection time register, TC is set high.
- The revocation of the TC flag is not affected by the smart card mode.
- If the transmitter detects a framing error (receives a NACK signal from the receiver), the transmitter's receive function module does not detect the NACK as a start bit. The duration of the received NACK can be 1 or 2 baud clock cycles according to the ISO protocol.
- On the receiver side, if a checksum error is detected and a NACK is sent, the receiver does not detect the NACK as a start bit.

**Notes:**

1. The break symbol has no meaning in smart card mode. A 00h data with a framing error will be treated as data instead of a break symbol.
2. no IDLE frames are sent when switching the TE bits back and forth. the ISO protocol does not define IDLE frames.

The following figure details how the USART samples the NACK signal. In this example, the USART is sending data and is configured for 1.5 stop bits. In order to check the integrity of the data and the NACK signal, the USART's receive function block is activated.



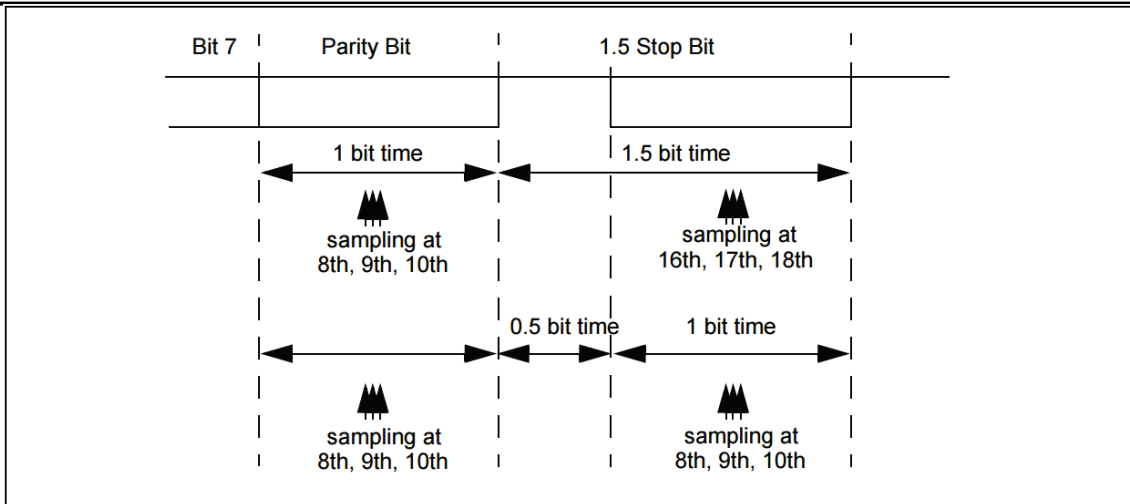


Figure263 Detecting parity check errors using 1.5 stop bits

The USART can provide a clock to the smart card via the CK output. In smart card mode, CK is not directly associated with communication, but first simply drives the smart card clock with an internal peripheral input clock via a 5-bit prescaler. The divider frequency 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.

### 28.3.12 IrDA SIR ENDEC Function Module

IrDA mode is selected by setting the IREN bit in the USART\_CR3 register. In IrDA mode, the lower column bits must be kept clear:

- LINEN, STOP, and CLKEN bits of the USART\_CR2 register
- SCEN and HDSEL bits of the USART\_CR3 register.

The IrDA SIR physical layer specifies the use of an inverted-zero modulation scheme (RZI), which uses an infrared light pulse to represent a logic '0' (seeFigure264 ). The SIR transmits an encoder to modulate the NRZ (non-zeroing) bit stream output from the USART. The output pulse stream is transmitted to a

The USART for SIR ENDEC only supports a maximum rate of 115.2 Kbps. In normal mode, the pulse width is specified as 3/16 of a bit period.

The SIR receive decoder demodulates the nulled bit stream from the IR receiver and outputs the received NRZ serial bit stream to the USART. In the idle state, the decoder input is normally high (marking state). The transmit encoder output has the opposite polarity of the decoder input. When the decoder input is low, a start bit is detected.

- IrDA is a half-duplex communication protocol. If the transmitter is busy (that is, 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 (that is, the USART is receiving decoded data from the IrDA decoder), data on TX from USART to IrDA will not be encoded by IrDA. When receiving data, you should avoid sending it because the data that will be sent may be corrupted.

- The SIR transmit logic sends '0' as a high pulse and '1' as a low level. The width of the pulse is specified as 3/16 of the bit period in normal mode (seeFigure265 ).

- The SIR receive logic interprets high level states as '1' and low pulses as '0'.

- The transmit encoder output has opposite polarity to the decoder input. When idle, the SIR output is low.

- The SIR decoder converts the IrDA-compatible receive signals into a bitstream to the USART.

- The IrDA specification requires pulses to be wider than 1.41us. pulse width is programmable. The spike detection logic at the receiver side filters out pulses that are less than 2 PSC cycles wide (PSC is a prescaled value programmed in the IrDA low power baud rate register USART\_GTPR). Pulses less than 1 PSC cycle in width must be filtered out, but those with a width greater than 1 and less than 2 PSC cycles may be received or filtered out, and those with a width greater than 2 cycles will be considered a valid pulse. When PSC=0, the IrDA encoder/decoder does not operate.

- The receiver can communicate with a low-power transmitter.

- In IrDA mode, the STOP bit on the USART\_CR2 register must be configured as a 1 stop bit.

## IrDA Low Power Mode

### transmitters

In low power mode, the pulse width no longer lasts for 3/16 of a bit period. Instead, the width of the pulse 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 frequency divider divides the system clock to achieve this value.

### refraction

Low power mode reception is similar to normal mode reception. To filter out spiky interference pulses, the USART should filter out pulses shorter than 1 PSC in width. Only low level signals from the IrDA low power baud rate clock (PSC in USART\_GTPR) with a duration greater than 2 cycles are accepted as valid.

Notes: 1. pulses with widths less than 2 greater than 1 PSC cycle may or may not be filtered.  
2. The receiver setup time should be managed by software. the IrDA physical layer specification specifies a minimum delay of 10ms between transmission and reception (IrDA is a half-duplex protocol).

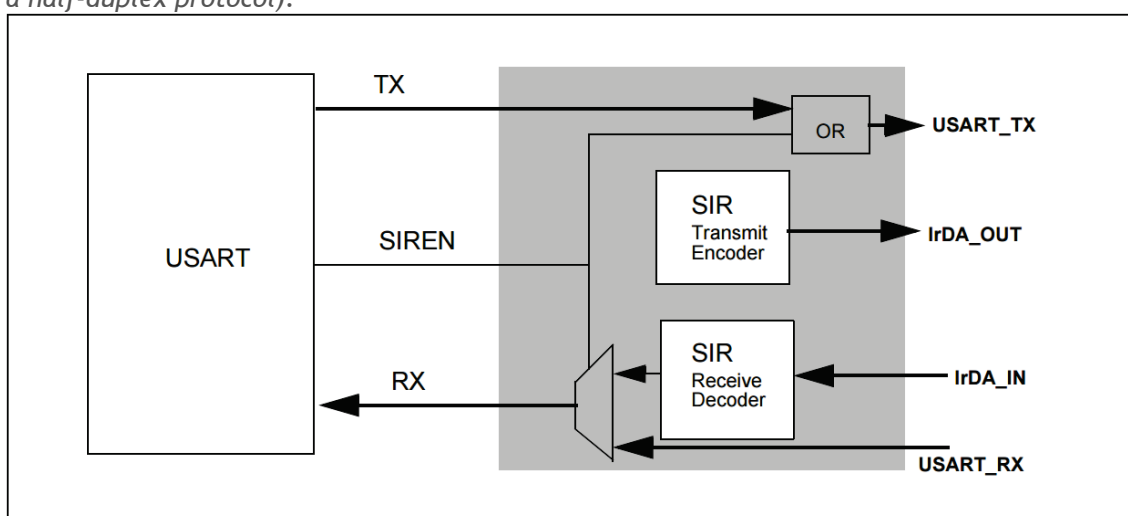


Figure 264 IrDA SIR ENDEC Block Diagram

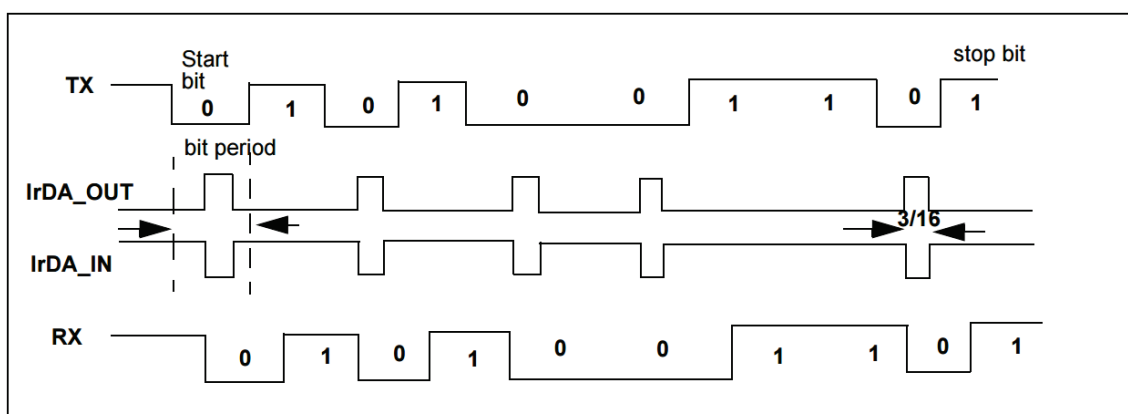


Figure 265 IrDA Data Modulation (3/16) Normal Mode

## 28.3.13 Continuous Communication using DMA

The USART can communicate continuously using DMA. the DMA requests for the Rx buffer and the Tx buffer are generated separately.

### Sending with DMA

Transmission using DMA can be activated by setting the DMAT bit on 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. To assign a DMA channel for USART transmissions, proceed as follows (x indicates the channel number):

1. Configure the USART\_DR register address on the DMA control register as the destination address for DMA transfers. After each TXE event, data will be transferred to this address.
2. The memory address is configured on the DMA control register as the source address for DMA transfers. 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 transferred in the DMA control register.
4. Configure the channel priority on the DMA registers.
5. Depending on the requirements of the application program, configure whether to generate a DMA interrupt when the transfer is half or all the way complete.
6. Activate the channel on the DMA register.

When the transfer has completed the amount of data specified by the DMA controller, the DMA controller generates an interrupt on the interrupt vector for that DMA channel.

In transmit mode, the DMA controller sets the TCIF flag of the DMA\_ISR register when the DMA has finished transmitting all the data to be sent; monitoring the TC flag of the USART\_SR register confirms the end of the USART communication, which avoids corrupting the last transmitted data before shutting down the USART or entering the shutdown mode; the software needs to first wait for TXE= 1 and then wait for TC=1.

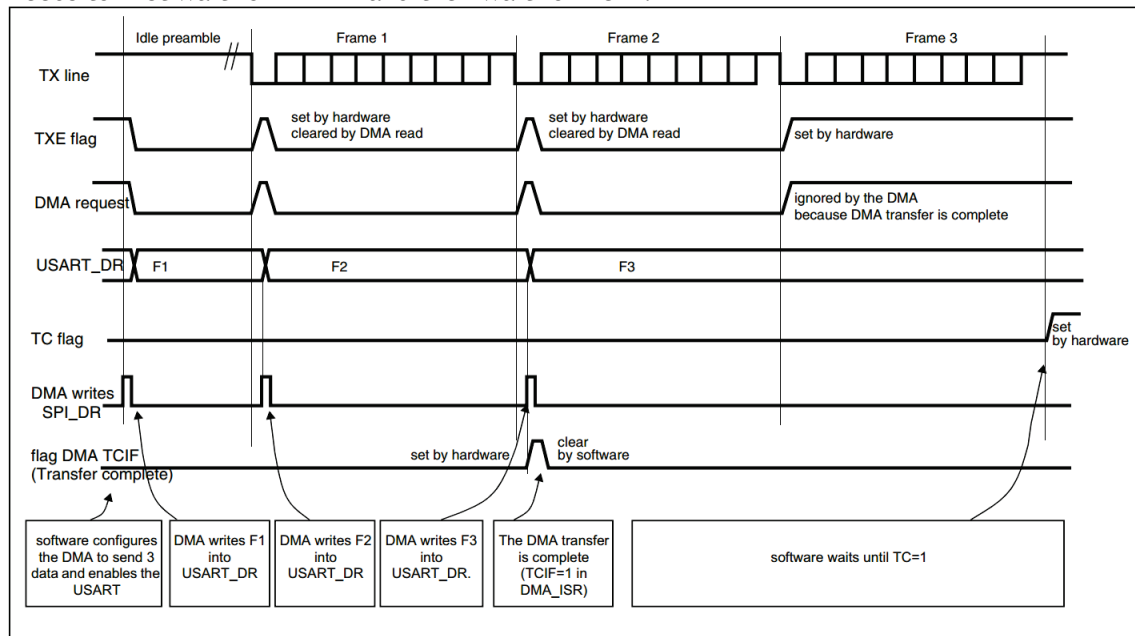


Figure266 Send using DMA

### Receive using DMA

Reception using DMA 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). To assign a DMA channel for USART reception, proceed as follows (x indicates the channel number):

1. The USART\_DR register address is configured 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. The memory address is configured 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 transferred in the DMA control register.
4. Configure the channel priority on the DMA registers.
5. Configure whether the DMA interrupt is generated when the transfer is half or fully completed, depending on the requirements of the application program.
6. Activate the channel on the DMA control register.

When reception of the amount of transmission specified by the DMA controller is completed, the DMA controller generates an interrupt on the interrupt vector for that DMA channel.

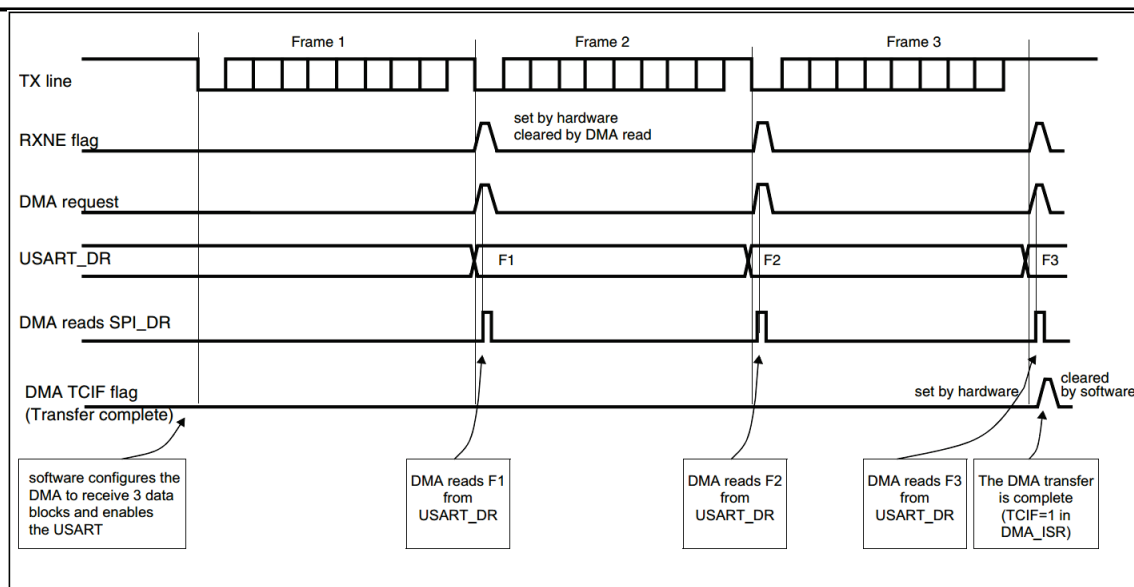


Figure267 Receiving with DMA

### 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 is transmitted. If the interrupt enable bit is set, an interrupt is generated. In the case of single byte reception, the frame error, overflow error, and noise flags, which are set together with RXNE, have separate error flag interrupt enable bits; if set, an interrupt is generated after the current byte has been transmitted.

## 28.3.14 Hardware Flow Control

The nCTS input and nRTS output can be used to control the serial data flow between two devices. The following diagram shows how to connect two devices in this mode.

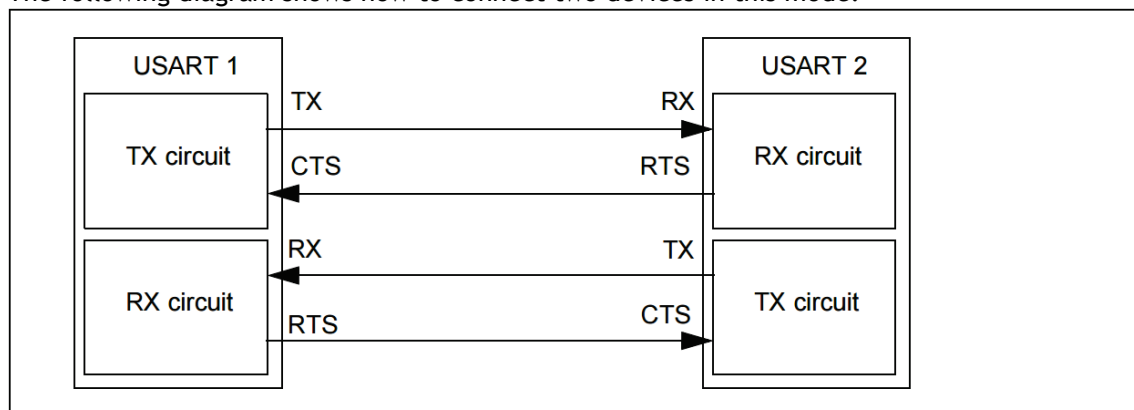


Figure268 Hardware flow control between two USARTs

RTS and CTS flow control can be enabled independently by setting RTSE and CTSE in UASRT\_CR3, respectively.

### RTS Flow Control

If RTS flow control is enabled (RTSE=1), nRTS becomes active (connected low) as long as the USART receiver is ready to receive new data. When data arrives in the receive register, nRTS is released, thus indicating that it is desired to stop data transmission at the end of the current frame. The following figure shows an example of communication with RTS flow control enabled.

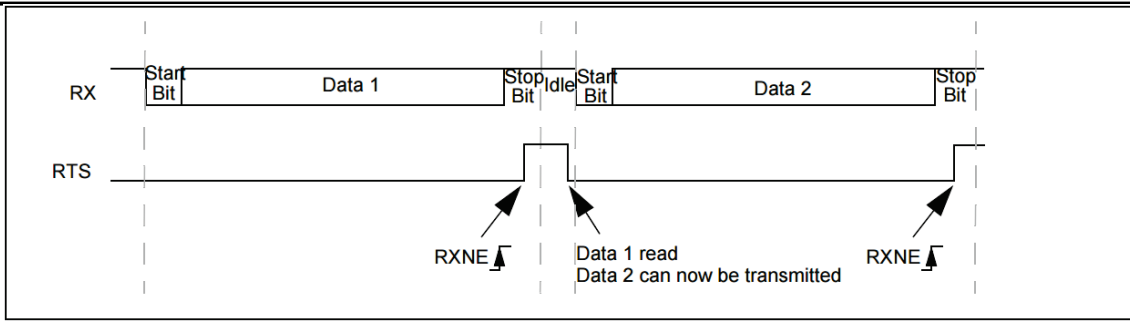


Figure269 RTS Flow Control

### 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 valid (pulled low), the next data is sent (assuming that that data is ready to be sent, i.e. TXE=0), otherwise the next frame is not sent. If nCTS is made invalid during a transmission, sending stops when the current transmission is complete.

When CTSE=1, the hardware automatically sets the CTSIF status bit as soon as the nCTS input changes state. It indicates whether the receiver is ready to communicate. If the CTSIE bit of the USART\_CT3 register is set, an interrupt is generated. The following figure shows an example of enabling CTS flow control communication.

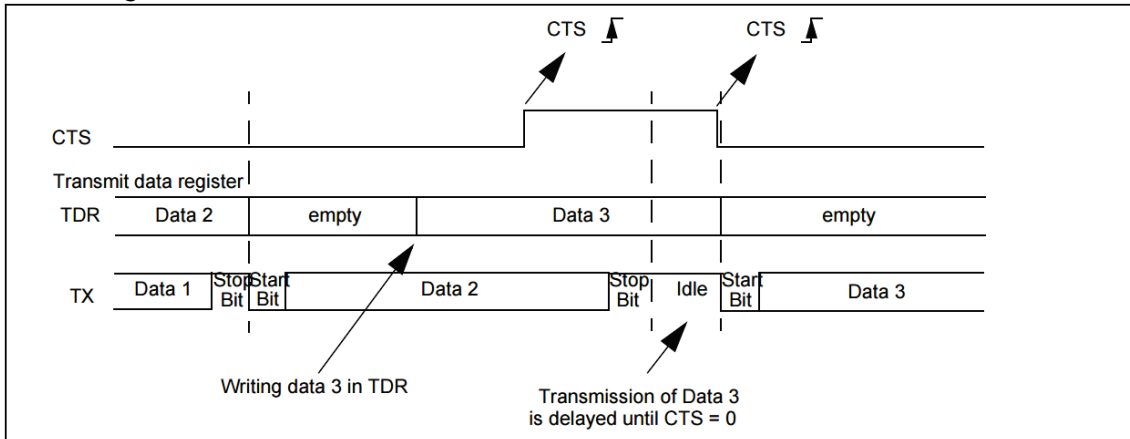


Figure270 CTS Flow Control

## 28.4 USART Interrupt Request

disruption event	event marker	enable bit
Send Data Register Empty	TXE	TXEIE
CTS logo	CTS	CTSIE
Send complete	TC	TCIE
Receive data ready to read	TXNE	TXNEIE
Data overflow detected	ORE	
Idle line detected	IDLE	IDLEIE
The parity test is wrong.	PE	PEIE
Disconnect symbol	LBD	LBDIE
Noise flags, overflow errors and framing errors in multi-buffered communications	NE or ORT or FE	EIE <sup>(1)</sup>

(1). This flag bit is used only when receiving data using DMA.

The various interrupt events of the USART are connected to the same interrupt vector (see below) with the following various interrupt events:

- During transmit: transmit complete, clear transmit, transmit data register empty.
- During reception: idle bus detection, overflow error, receive data register non-empty, checksum error, LIN break symbol detection, noise flag (only in multi-buffer communication) and frame error (only in multi-buffer communication).

These events can generate their own interrupts if the corresponding enable control bits are set.

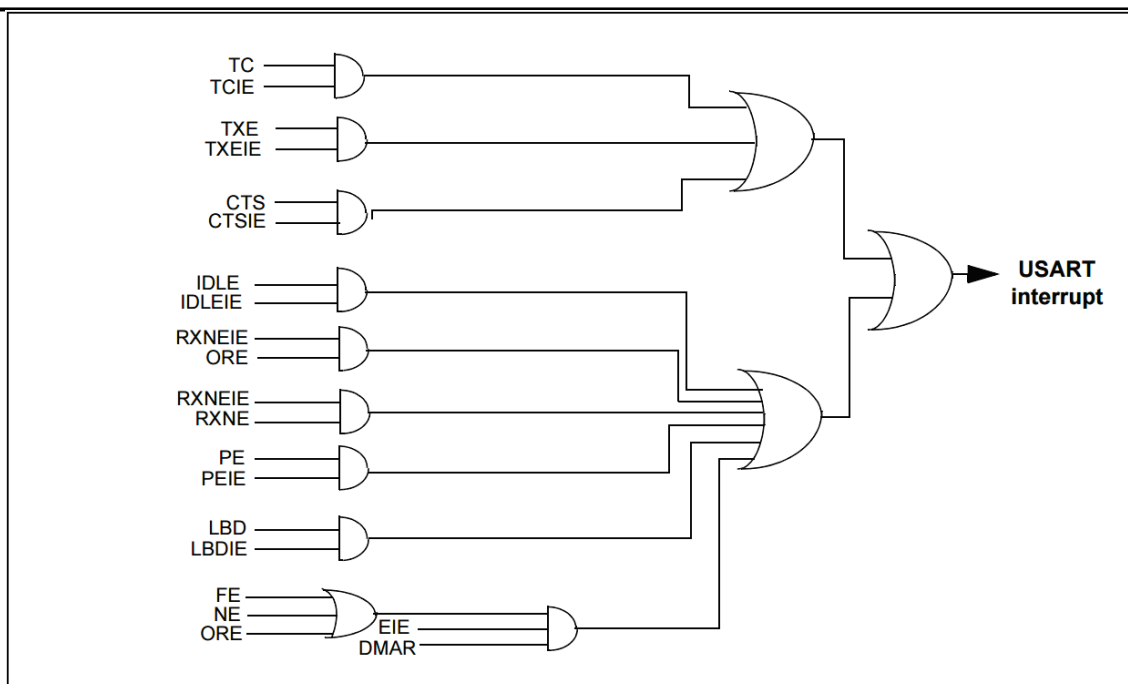


Figure271 USART interrupt image

## 28.5 USART Mode Configuration

Table150 USART Mode Settings <sup>(1)</sup>

USART mode	USART1	USART2	USART3	UART4	UART5
asynchronous mode	⊙	⊙	⊙	⊙	⊙
hardware flow control	⊙	⊙	⊙	●	●
Multi-cache communication (DMA)	⊙	⊙	⊙	⊙	●
multiprocessor communication	⊙	⊙	⊙	⊙	⊙
synchronization	⊙	⊙	⊙	●	●
smart card	⊙	⊙	⊙	●	●
Half-duplex (single-wire mode)	⊙	⊙	⊙	⊙	⊙
IrDA	⊙	⊙	⊙	⊙	⊙
LIN	⊙	⊙	⊙	⊙	⊙

(1) ⊙ = supported, ● = does not support this application

## 28.6 USART Register Description

Refer to Section1 for the abbreviations used in the register descriptions.  
These peripheral registers can be operated in half-word (16-bit) or word (32-bit) format.

## 28.6.1 Status Register (USART\_SR)

Address offset: 0x00

Reset value: 0x00C0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CTS	LBD	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
						rc_w0	rc_w0	r	rc_w0	rc_w0	r	r	r	r	r

Bit	notation	clarification
31:10	Reserved	Reserved, always reads 0.
9	CTS	<p><b>CTS:</b>CTS flag If the CTSE bit is set, it is set high by hardware when the nCTS input changes state. It is cleared by software. If CTSIE in USART_CR3 is '1', an interrupt is generated. 0: No change on the nCTS status line; 1: Changes occur on the nCTS status line. Note: This bit does not exist on UART4 and UART5.</p>
8	LBD	<p><b>LBD:</b>LIN break detection flag When a LIN disconnect is detected, this bit is set '1' by hardware and cleared '0' by software (write 0 to this bit). If LBDIE=1 in USART_CR3, an interrupt is generated. 0: No LIN disconnection detected; 1: LIN disconnection detected. Note: If LBDIE=1, an interrupt is to be generated when LBD is '1'.</p>
7	TXE	<p><b>TXE:</b> transmit data register empty (Transmit data register empty) This bit is set by hardware when the data in the TDR register is transferred to the shift register by hardware. If TXEIE in the USART_CR1 register is 1, an interrupt is generated. A write operation to USART_DR clears this bit to zero. 0: Data has not been transferred to the shift register; 1: Data has been transferred to the shift register. Note: This bit is used in single buffer transfers.</p>
6	TC	<p><b>TC:</b> Transmission complete When a frame containing data has been sent and TXE=1, the position is '1' by hardware. If TCIE in USART_CR1 is '1', an interrupt is generated. The bit is cleared by a software sequence (read USART_SR, then write USART_DR).The TC bit can also be cleared by writing '0', a clearing procedure recommended only for multi-buffer communications. 0: Sending is not yet complete; 1: Sending completed.</p>
5	RXNE	<p><b>RXNE:</b> Read data register not empty (Read data register not empty) This bit is set by hardware when the data in the RDR shift register is transferred to the USART_DR register. This bit is set by hardware if the An interrupt is generated when RXNEIE in the USART_CR1 register is 1. A read operation of USART_DR clears this bit to zero.The RXNE bit can also be cleared by writing a 0. This clearing procedure is recommended only for multi-buffer communications. 0: Data not received; 1: Data received and can be read out.</p>
4	IDLE	<p><b>IDLE:</b> Bus idle monitored (IDLE line detected) This bit is set by hardware when bus idle is detected. If IDLEIE in USART_CR1 is '1', an interrupt is generated. The bit is cleared by a software sequence (USART_SR is read first, then USART_DR). 0: No idle bus detected; 1: Idle bus detected. Note: The IDLE bit will not be set high again until the RXNE bit is set (i.e., another idle bus is detected)</p>
3	ORE	<p><b>ORE:</b> Overrun error When RXNE is still '1', the hardware bits the data currently being received in the shift register that needs to be transferred to the RDR register. If RXNEIE in USART_CR1 is '1', an interrupt is generated. It is cleared by the software sequence (read USART_SR first, then USART_CR). 0: No overload error; 1: Overload error detected. Note: When this bit is set, the value in the RDR register is not lost, but the data in the shift register is overwritten. If the EIE bit is set, the ORE flag set in multi-buffer communication mode will generate an interrupt.</p>
2	NE	<p><b>NE:</b>Noise error flag By hardware to the position bit when noise is detected in the received frame. By software sequence to its clearing ling (read USART_SR first, then USART_DR). 0: No noise detected; 1: Noise detected. Note: This bit does not generate an interrupt because it appears with RXNE and the</p>

		hardware generates an interrupt when the RXNE flag is set. In multi-buffer communication mode, if the EIE bit is set, an interrupt is generated when the NE flag is set.
1	FE	<b>FE:Framing error</b> This bit is set by hardware when a synchronization mismatch is detected, excessive noise is detected, or a disconnect symbol is detected. It is cleared by a software sequence (USART_SR is read first, then USART_DR). 0: No frame error detected; 1: Frame error or break character detected. Note: This bit does not generate an interrupt because it appears with RXNE and the hardware generates an interrupt when the RXNE flag is set. If the currently transmitted data generates both a framing error and an overload error, the hardware continues that data transmission anyway and only sets the ORE flag bit. In multi-buffer communication mode, if the EIE bit is set, an interrupt is generated when the FE flag is set.
0	PE	<b>PE:Parity error</b> In receive mode, if there is a parity error, hardware to this location bit. It is cleared by the software sequence (reading USART_SR and USART_DR in sequence). Before clearing the PE bit, software must wait for the RXNE flag bit to be set to '1'. If the PEIE in USART_CR1 is '1', an interrupt is generated. 0: No parity error; 1: Parity error.

## 28.6.2 Data Register (USART\_DR)

Address offset: 0x04

Reset value: Uncertain

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

Bit	notation	clarification
31:9	Reserved	Reserved, always reads 0.
8:0	DR[8:0]	<b>DR[8:0]: Data value</b> Contains the data to be sent or received. Since it is composed of two registers, one for transmitting (TDR) and one for receiving (RDR), this register has both read and write functions. The TDR register provides a parallel interface between the internal bus and the output shift registers (see Figure 248). The RDR register provides a parallel interface between the input shift registers and the internal bus. When enabling the parity bit (PCE bit in USART_CR1 is set) for transmission, the value written to the MSB (which is bit 7 or 8 depending on the length of the data) is replaced by the later parity bit that. When the parity bit is enabled for reception, the MSB bit read is the received parity bit.

## 28.6.3 Baud Rate Register (USART\_BRR)

Notes: The baud counter stops counting if TE or RE is disabled respectively

Address offset: 0x08

Reset value: 0x0000

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

Bit	notation	clarification
31:16	Reserved	Reserved, always reads 0.
15:4	DIV_Mantissa [11:0]	<b>DIV_Mantissa[11:0]: integer part of USARTDIV</b> These 12 bits define the integer portion of the USART divider division factor (USARTDIV).
3:0	DIV_Fraction [3:0]	<b>DIV_Fraction[3:0]: decimal fraction of USARTDIV</b> These 4 bits define the fractional part of the USART divider division factor (USARTDIV).



## 28.6.4 Control Register 1 (USART\_CR1)

Address offset: 0x0C

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	UE	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	RWU	SBK	
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	notation	clarification
31:14	Reserved	Reserved, always reads 0.
13	UE	<b>UE:</b> USART enable When this bit is cleared, the divider and outputs of the USART stop working after the current byte transfer is complete to minimize power consumption. This bit is set and cleared by software. 0: USART divider and outputs are disabled; 1: USART module enable.
12	M	<b>M:</b> Word length This bit defines the length of the data word, which is set and cleared by software 0: one start bit, 8 data bits, n stop bits; 1: One start bit, 9 data bits, n stop bits. Note: This bit cannot be modified during data transmission (when sending or receiving).
11	WAKE	<b>WAKE:</b> Wakeup method This bit determines the method of waking the USART up and is set and cleared by software. 0: Wake up by idle bus; 1: Awakened by an address marker.
10	PCE	<b>PCE:</b> Parity control enable Use this bit to select whether or not to perform hardware parity control (parity bit generation for transmit; parity bit detection for receive). When this bit is enabled, a parity bit is inserted in the highest bit of the transmitted data (if M=1, the highest bit is bit 9; e.g. if M=0, the highest bit is bit 8); the received data is checked for its parity bit. The software sets '1' or clears '0' to it. Once the bit is set, the parity control takes effect only after the current byte transmission is complete. 0: Checksum control is disabled; 1: Enable checksum control.
9	PS	<b>PS:</b> Parity selection When the checksum control is enabled, this bit is used to select whether to use even or odd checksum. It is set '1' or cleared '0' by software. The selection takes effect when the current byte transfer is complete. 0: Even check; 1: Odd calibration.
8	PEIE	<b>PEIE:</b> PE interrupt enable This bit is set or cleared by software. 0: Interrupt generation is disabled; 1: When the PE in USART_SR is '1', a USART interrupt is generated.
7	TXEIE	<b>TXEIE:</b> TXE interrupt enable This bit is set or cleared by software. 0: Interrupt generation is disabled; 1: When TXE in USART_SR is '1', a USART interrupt is generated.
6	TCIE	<b>TCIE:</b> Transmission complete interrupt enable This bit is set or cleared by software. 0: Interrupt generation is disabled; 1: When TC in USART_SR is '1', a USART interrupt is generated.
5	RXNEIE	<b>RXNEIE:</b> Receive Buffer Non-Air Interrupt Enable (RXNE interrupt enable) This bit is set or cleared by software. 0: Interrupt generation is disabled; 1: When ORE or RXNE in USART_SR is '1', a USART interrupt is generated.
4	IDLEIE	<b>IDLEIE:</b> IDLE interrupt enable This bit is set or cleared by software. 0: Interrupt generation is disabled; 1: When IDLE in USART_SR is '1', a USART interrupt is generated.
3	TE	<b>TE:</b> Transmitter enable This bit enables the transmitter. This bit is set or cleared by software. 0: Disable sending; 1: Enable Transmit. Note: 1. During data transmission, except in smart card mode, if there is a 0 pulse on the TE bit (i.e., set to '0' and then set to '1'), a "lead" (idle bus) will be sent after the current data word has been transmitted. 2. When the TE is set, there is a delay of one bit time before the actual transmission starts.
2	RE	<b>RE:</b> Receiver enable This bit is set or cleared by software. 0: Receiving is prohibited; 1: Enables reception and starts searching for the start bit on the RX pin.

1	RWU	<b>RWU:</b> Receiver wakeup This bit is used to determine whether or not to place the USART in silent mode. This bit is set or cleared by software. It is also cleared by hardware when a wake-up sequence arrives. 0: The receiver is in normal operating mode; 1: The receiver is in silent mode. Note: 1. The USART must have received a data byte before putting it into silent mode (setting the RWU bit). Otherwise, it cannot be woken up by idle bus detection in silent mode. 2. When configured for address mark detection wake-up (WAKE bit = 1), the RWU bit cannot be modified by software while the RXNE bit is set.
0	SBK	<b>SBK:</b> Send break frame (Send break) Use this bit to send a break character. This bit can be set or cleared by software. The procedure should be for software to set the bit it, and then for hardware to reset the bit when the stop bit of the break frame. 0: No disconnect character is sent; 1: The break character will be sent.

## 28.6.5 Control Register 2 (USART\_CR2)

Address offset: 0x10

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserv ed	LINEN	STOP[1:0]	CLKEN	CPOL	CPHA	LBCL	Reserv ed	LBDIE	LBDL	Reserv ed	ADD[3:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	notation	clarification
31:15	Reserved	Reserved, always reads 0.
14	LINEN	<b>LINEN:</b> LIN mode enable This bit is set or cleared by software. 0: Disable LIN mode; 1: Enable LIN mode. In LIN mode, you can use the SBK bit in the USART_CR1 register to send the LIN Synchronization Disconnect Token (low 13 bits), as well as to detect the LIN Synchronization Disconnect Token.
13:12	STOP	<b>STOP:</b> STOP bits These 2 bits are used to set the number of stop bits. 00: 1 stop bit; 01: 0.5 stop bits; 10: 2 stop bits; 11: 1.5 stop bits; Note: UART4 and UART5 cannot use 0.5 stop bit and 1.5 stop bit.
11	CLKEN	<b>CLKEN:</b> Clock enable This bit is used to enable the CK pin. 0: Disable CK pin; 1: Enable CK pin. Note: This bit does not exist on UART4 and UART5.
10	CPOL	<b>CPOL:</b> Clock polarity In synchronization mode, this bit can be used to select the polarity of the clock output on the SLCK pin. Works in conjunction with the CPHA bit to generate the desired clock/data sampling relationship 0: Hold low on the CK pin when the bus is idle; 1: Hold high on the CK pin while the bus is idle. Note: This bit does not exist on UART4 and UART5.
9	CPHA	<b>CPHA:</b> Clock phase In synchronization mode, this bit can be used to select the phase of the clock output on the SLCK pin. Work with the CPOL bit to generate the desired clock/data sampling relationship (seeFigure259 andFigure260 ). 0: Data capture at the first edge of the clock; 1: Data capture on the second edge of the clock. Note: This bit does not exist on UART4 and UART5.
8	LBCL	<b>LBCL:</b> Last bi tclock pulse In synchronization mode, this bit is used to control whether or not the clock pulse corresponding to the last data byte (MSB) sent is output on the CK pin 0: The clock pulse for the last bit of data is not output from CK; 1: The clock pulse for the last bit of data will be output from CK. Note: 1. The last data bit is the 8th or 9th bit sent (according to the 8- or 9-bit data frame format defined by the M bit in the USART_CR1 register). 2. This bit does not exist on UART4 and UART5.
7	Reserved	Reserved bit, hardware forced to 0
6	LBDIE	<b>LBDIE:</b> LIN break detection interrupt enable (LIN break detection interrupt enable) break detection interrupt mask (use break separator to detect breaks)

		0: Interrupts are disabled; 1: An interrupt is generated whenever the LBD in the USART_SR register is '1'.
5	LBDL	LBDL: LIN break detection length (LIN break detection length) This bit is used to select whether the break detection is 11-bit or 10-bit. 0: 10-bit break character detection; 1: 11-bit break character detection.
4	Reserved	Reserved bit, hardware forced to 0
3:0	ADD[3:0]	ADD[3:0]: the USART node address of this device This bit field gives the address of the USART node of this device. This is used in silent mode under multiprocessor communication, using an address tag to wake up a particular USART device.

Notes: These three bits (CPOL, CPHA, LBCL) cannot be rewritten after enabling transmission.

## 28.6.6 Control Register 3 (USART\_CR3)

Address offset: 0x14

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	notation	clarification
31:11	Reserved	Reserved, always reads 0.
10	CTSIE	CTSIE: CTS interrupt enable 0: Interrupts are disabled; 1: An interrupt is generated when CTS in the USART_SR register is '1'. Note: This bit does not exist on UART4 and UART5.
9	CTSE	CTSE: CTS enable 0: Disable CTS hardware flow control; 1: CTS mode enable, data can be sent only when the nCTS input signal is valid (pulled low). If the nCTS signal becomes invalid during data transmission, the transmission stops after sending this data. If data is written to the data register when nCTS is invalid, this data is not sent until nCTS is valid. Note: This bit does not exist on UART4 and UART5.
8	RTSE	RTSE: RTS enable 0: Disable RTS hardware flow control; 1: RTS interrupt enable to request the next data only when there is free space in the receive buffer. Once the current data has been sent, the send operation needs to be paused. If it is ready to receive data, enable (pull low) the nRTS output. Note: This bit does not exist on UART4 and UART5.
7	DMAT	DMAT: DMA enable transmitter This bit is set or cleared by software. 0: DMA mode when sending is disabled. 1: Enable DMA mode on transmit; Note: This bit does not exist on UART4 and UART5.
6	DMAR	DMAR: DMA enable receiver (DMA enable receiver) This bit is set or cleared by software. 0: Disable DMA mode on receive. 1: Enable DMA mode on receive; Note: This bit does not exist on UART4 and UART5.
5	SCEN	SCEN: Smartcard mode enable This bit is used to enable smartcard mode. 0: Disable smart card mode; 1: Enable smart card mode. Note: This bit does not exist on UART4 and UART5.
4	NACK	NACK: Smartcard NACK enable 0: No NACK is sent when a checksum error occurs; 1: A NACK is sent when a checksum error occurs. Note: This bit does not exist on UART4 and UART5.
3	HDSEL	HDSEL: Half-duplex selection (Half-duplex selection) Selects single-line half-duplex mode. 0: Half-duplex mode is not selected; 1: Select half-duplex mode.
2	IRLP	IRLP: Infrared low-power (IrDA low-power) This bit is used to select normal mode or low power IR mode 0: Usual mode; 1: Low power mode.
1	IREN	IREN: Infrared mode enable (IrDA mode enable) This bit is set or cleared by software. 0: Do not enable IR mode; 1: Enable IR mode.
0	EIE	EIE: Error interrupt enable

		In multi-buffer communication mode, an interrupt is generated when there is a frame error, overload, or noise error (FE=1, or ORE=1, or NE=1 in USART_SR). 0: Interrupts are disabled; 1: Generate an interrupt whenever DMAR=1 in USART_CR3 and FE=1, or ORE=1, or NE=1 in USART_SR
--	--	--

## 28.6.7 Guard Time and Prescaler Register (USART\_GTPR)

Address offset: 0x18

Reset value: 0x0000

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

Bit	notation	clarification
31:11	Reserved	Reserved, always reads 0.
15:8	GT[7:0]	GT[7:0]: Guard time value This bit field specifies the protection time in baud clocks. This is required in smart card mode. The transmit complete flag is set when the protection time has elapsed. Note: This bit does not exist on UART4 and UART5.
7:0	PSC[7:0]	PSC[7:0]: Prescaler value -In infrared (IrDA) low power mode: PSC[7:0] = infrared low power baud rate Frequency division of the system clock to obtain the frequency in low-power mode: the source clock is divided by the value in the register (only 8 bits are valid) 00000000: Reserved-Do not write this value; 00000001: Frequency division of source clock 1; 00000010: Divide the source clock by 2; ..... -In normal mode with infrared (IrDA): PSC can only be set to 00000001 -In smart card mode: PSC[4:0]: prescaler value The system clock is divided to provide a clock to the smart card. The value given in the register (lower 5 bits are valid) multiplied by 2 is used as the dividing factor for the source clock 00000: Reserved-Do not write this value; 00001: 2-division frequency of the source clock; 00010: 4-division frequency of the source clock; 00011: 6-division frequency of the source clock; ..... Note: 1. Bits [7:5] have no meaning in smart card mode. 2. This bit does not exist on UART4 and UART5.

## 28.6.8 USART Register Address Map

Table151 List of USART registers and their Reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h	USART_SR	Reserved																						CTS	LBD	TXEIE	TC	RXNE	IDLE	ORE	NE	FE	PE
	Reset value																							0	0	1	1	0	0	0	0	0	0
004h	USART_DR	Reserved																						DR[8:0]									
	Reset value																							0	0	0	0	0	0	0	0	0	0
008h	USART_BRR	Reserved												DIV_Mantissa [15:4]								DIV_Fraction [3:0]											
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
00Ch	USART_CR1	Reserved												UE	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	RWU	SBK						
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
010h	USART_CR2	Reserved												LIEN	STOP [1:0]	CLKEN	CPOL	CPHA	LBCL	Reserved	LBDIE	LBDL	Reserved	ADD[3:0]									
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
014h	USART_CR3	Reserved												CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE									
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
018h	USART_GTPR	Reserved												GT[7:0]				PSC[7:0]															
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

SeeTable1 for register start addresses.

## 29 USB On-The-Go Full-Speed (OTG\_FS)

### 29.1 OTG Module Introduction

Partially copyrighted by Synopsys (2004, 2005). All rights reserved. Permission to use has been granted. This chapter describes the architecture of the OTG\_FS controller and how to program it for use.

Terminology used in this chapter:

FS: Full Speed  
 LS: Low Speed  
 USB: Universal Serial Bus  
 OTG: On-the-Go  
 PHY: Physical layer  
 USB: Universal Serial Bus  
 UTMI: USB 2.0 Transceiver Macrocell Interface (UTMI)

Refer to the following documents:

- USBOn-The-Go Supplement, Revision 1.3
- Universal Serial Bus Revision 2.0 Specification

OTG\_FS is a Dual Role Device (DRD) controller that supports both host-side and device-side functionality and is fully compliant with the On-The-Go Supplement to the USB 2.0 specification. The controller can also be configured to support host-only or device-only functionality in compliance with the USB 2.0 specification. In host mode, the OTG\_FS supports full-speed (FS, 12Mbps/s) and low-speed (LS, 1.5Mbps/s) communication, while in device mode, full-speed (FS, 12Mbps/s) communication is supported. The OTG\_FS controller supports both HNP and SRP protocols. The peripheral only needs to be configured with a charge pump for VBUS in host mode to complete the design.

### 29.2 OTG\_FS Main Functions

The following section describes the characteristics of the OTG\_FS controller in three areas: general, host mode and device mode.

#### 29.2.1 Common Functionality

OTG\_FS controller interface:

- Certified by USB-IF to comply with Universal Serial Bus Specification, Revision 2.0 standard
- Fully supports the OTG protocol defined as an optional item in the (OTG\_FS Controller's Physical Layer (PHY)) USBOn-The-Go Supplement, Revision 1.3 specification.
  - Identification of inserted Class A-B devices (ID lines)
  - Support for Host Negotiation Protocol (HNP) and Session Request Protocol (SRP)
  - In OTG applications, allow the host to turn off VBUS to save power consumption
  - The OTG controller uses an internal comparator to monitor the VBUS level
  - Can dynamically switch host/device roles
- The following design can be accomplished through software configuration:
  - USB full-speed devices that support the SRP protocol (Class B devices)
  - USB full-speed/low-speed hosts supporting SRP protocol (Class A devices)
  - USB on-the-go full-speed Dual Role Device
- Supports SOF signaling for full-speed communication and hold-valid signaling for low-speed communication
  - SOF pulses can be output to pins
  - SOF is internally connected to Timer 2 (TIM2)
  - Configurable frame period
  - Configurable end-of-frame interrupt
- Provides power saving features: such as stopping the system when USB hangs, shutting down the internal clock system of the digital section, PHY and DFIFO power management sections.
- Provides 1.25K bytes of dedicated RAM and advanced FIFO management
  - Configure different RAM areas for different FIFOs via software for flexible and efficient RAM usage
  - Each FIFO can store multiple packets
  - Allows dynamic allocation of storage areas

- The length of the FIFO is not limited to a power of 2 so that the storage area can be used continuously.
- Maximum data traffic is guaranteed for one frame (1ms) without system intervention.

## 29.2.2 Host Mode Functions

OTG\_FS controller interface:

- Requires an external charge pump to power the VBUS
- Supports up to 8 host channels, each of which can be dynamically configured for any transmission type
- Built-in hardware scheduling controller:
  - Supports up to 8 interrupt and synchronized transfer requests in the periodic hardware transfer request queue
  - Supports up to eight transfer requests for control and high-capacity transfers in a queue of non-periodic hardware transfer requests.
- For efficient use of RAM space, the USB data RAM area is divided into a shared receive FIFO, a cyclic transmit FIFO, and an acyclic transmit FIFO.

### Device Mode Functions

OTG\_FS controller interface:

- Provides 1 bi-directional control endpoint 0
- Provides 3 IN endpoints to support high-capacity, interrupted or synchronized transmission
- Provides 3 OUT endpoints to support high-capacity, interrupted or synchronized transmission
- Manages a shared receive FIFO and a transmit OUTFIFO for efficient use of the USB data RAM area.
- Manages up to 4 dedicated transmit INFIFOs (one FIFO for each IN endpoint) to minimize application loads
- Supports the disconnect function of the software.

## 29.3 OTG\_FS Functional Description

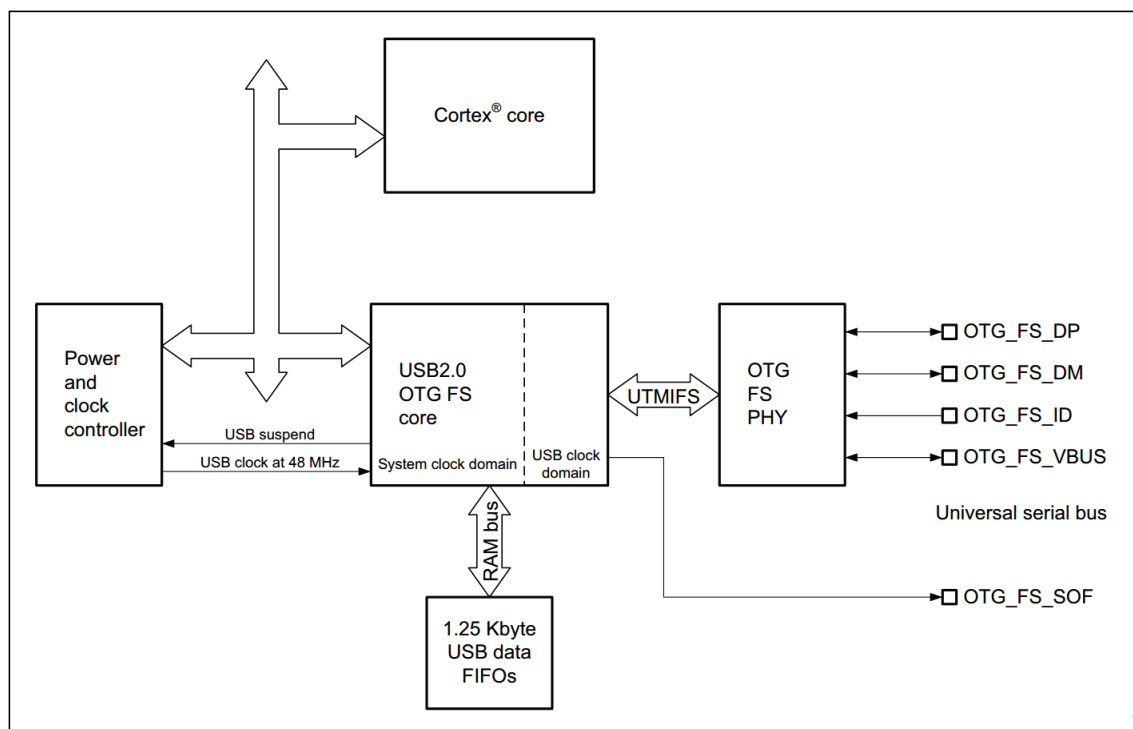


Figure272 Block Diagram

### 29.3.1 OTG Pin

Table152 OTG Pins

Signal Name	Signal Type	descriptive
OTG_FS_DP	Digital inputs/outputs	USB OTG D+ Cable
OTG_FS_DM	Digital inputs/outputs	USB OTG D-Cable
OTG_FS_ID	digital input	USB OTG ID
OTG_FS_VBUS	analog input	USB OTG VBUS
OTG_FS_SOF	digital output	USB OTG frame start (visibility)

### 29.3.2 OTG Full Speed Controller

The USB OTG Full Speed Controller obtains from the Reset and Clock Control Module (RCC) a 48MHz  $\pm 0.25\%$  clock generated by an external crystal, which is used to drive the 48MHz control clock of the USB Full Speed Module (12Mbit/s), which must be enabled before the OTG Full Speed Controller can be configured.

The microcontroller core (CPU) accesses the OTG Full Speed Controller registers via the AHB peripheral bus, and USB events are managed by a separate USBOTG interrupt control line (see Section 29.15 : OTG\_FS Interrupt) that notifies the microcontroller core.

The microcontroller transfers data to the USB controller in the form of 32-bit data writes to the OTG\_FS dedicated address (PUSH register), which are automatically deposited into the configured transmit FIFOs in the USB data RAM. Each send FIFO is configured with such a PUSH register for each IN endpoint (device mode) or OUT channel (host mode).

The microcontroller obtains 32-bit data from the USB bus by reading the dedicated address of the OTG\_FS (POP register), which is automatically loaded from the shared receive FIFO. The receive FIFO is located in a total of 1.25K bytes of USB data RAM area. Each OUT endpoint or IN channel is served by such a POP register.

The USB protocol layer is driven by the Serial Interface Controller (SIE) and connects to the USB full speed/low speed transceiver modules supported by the built-in physical layer (PHY).

### 29.3.3 Full-Speed OTGPHY (Physical Interface)

The built-in full-speed OTGPHY is controlled by the OTG full-speed controller and sends and receives via the full-speed subset of the UTMI+ bus (UTMIFS) control and data signals. the PHY provides physical layer support for USB communication. The full-speed OTGPHY consists of the following components:

- Full speed/low speed transceiver module for use in host and device modes. This module drives transmit and receive directly on a single-ended USB cable.
- A pull-up resistor for the ID line is built in to differentiate between Class A/B devices.
- The DP/DM lines have built-in pull-up and pull-down resistors that are controlled by the OTG\_FS controller to meet the needs of the current device type. In device mode, when a valid level (Class B active) is present on the VBUS line, the controller enables the pull-up resistor on the DP line, announcing to the host that a USB full-speed device is being accessed. In host mode, the controller enables the pull-down resistors on both the DP and DM lines. The pull-up and pull-down resistors can be switched dynamically as the controller switches role types via Host Negotiation Protocol (HNP).
- ECN circuit with pull-up/down resistors. The DP line has two pull-up resistors separately controlled by the OTG\_FS controller, which is compliant with the ECN requirements of USB version 2.0 (Engineering Change Notice). Supports dynamic adjustment of the pull-up strength of the DP line to better combat noise and improve transmit and receive signal quality.
- A built-in VBUS detection comparator with hysteresis is used to detect valid VBUS levels, valid A-B session levels, and end-of-session levels. These levels are used to drive the Session Request Protocol (SRP), to detect valid session start and end conditions, and to continuously detect the VBUS line status during USB communication.
- The VBUS pulse circuit is used for charging/discharging operation of the VBUS through a resistor during SRP (weak drive).



## 29.4 OTG Dual Role Device (DRD)

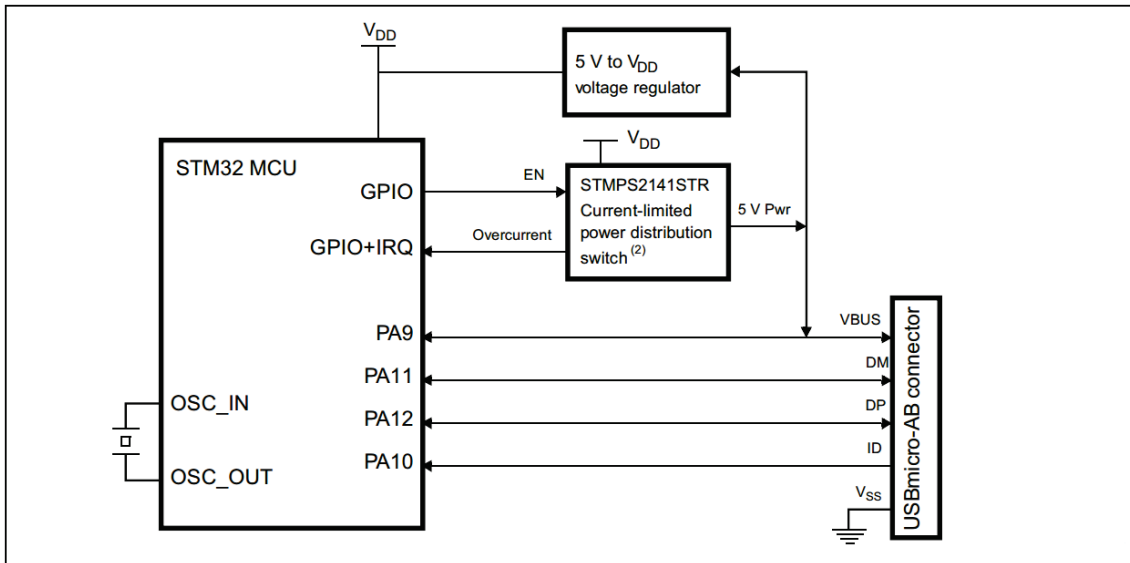


Figure 273 OTG A-B Device Connections

1. The use of an external voltage regulator is only necessary when designing equipment to be powered by VBUS.
2. The ST20x2 is required only when designing devices that use VBUS power supply. If a 5V supply is available on the application board, a basic power switch can be used.
3. VDD ranges from 2V to 3.6V.

### 29.4.1 ID Signal Detection

The roles of host and device (default) are defined by the state of the ID line. The state of the ID line is determined the moment it is connected to the USB bus and depends on the USB cable plugged into the micro-AB socket.

- If the USB cable is plugged into the B end of the USB cable and the ID line is floating, the built-in pull-up resistor will detect a high level on the ID line, which puts the controller in the default device mode. In this mode, the OTG\_FS controller adheres to the USB 2.0 Specification Supplement to the OTG 1.3 Specification, Section 6.8.2: OTGB Class Device Description of the Standard FSM.
- If the USB cable is plugged into end A and the ID line is grounded. At this point, the OTG\_FS controller will execute an ID line state change interrupt (CIDSCHG bit of the OTG\_FS\_GINTSTS register), automatically switching to host mode and requiring software to initialize host mode. In this mode, the OTG\_FS controller complies with Section 6.8.1: Description of OTGA Class Devices to Standard FSM of the OTG1.3 Specification, Supplement to the USB 2.0 Specification.

### 29.4.2 HNP Dual Role Devices

The HNP enable bit of the USB Global Configuration Register (HNPCAP bit of the OTG\_FS\_GUSBCFG register) allows the OTG\_FS controller to dynamically toggle between Class A hosts and Class A devices, and Class B hosts and Class B devices via the Host Negotiation Protocol (HNP). The current device state can be obtained by reading a combination of the ID status bit of the OTG global control and status register (CIDSTS bit of the OTG\_FS\_GOTGCTL register) and the current operating mode bit of the global interrupt and status register (CMOD bit of the OTG\_FS\_GINTSTS register).

Refer to Section 29.17 : OTG\_FS Programming for HNP programming rules.

### 29.4.3 SRP Dual Role Devices

The SRP capability bit in the Global USB Configuration Register (SRPCAP bit in OTG\_FS\_GUSBCFG) enables the OTG\_FS core to turn off VBus generation for device A to save power. Note that the A device is always responsible for driving VBus, regardless of the host or peripheral role of OTGFS.

The SRP A/B device programming model is described in detail in Section 26.17: OTG FS Programming Model.

## 29.5 USB Device Mode

This chapter describes the function of the OTG\_FS controller when it is in USB device mode. The OTG\_FS controller operates in device mode in the following cases:

- OTGB-type equipment
  - When the USB cable is plugged into the B end, the default is OTGB class device mode.
- OTGA type equipment
  - OTGA class devices switch to the device role via HNP protocols
- Category B equipment
  - When the ID line is present, the insertion is at the B end of the USB cable and the HNP bit of the USB Global Configuration Register (HNPCAP bit of the OTG\_FS\_GUSBCFG register) is '0' (refer to section 6.8.3 of OTG version 1.3)
- As a USB device only (please refer to the figure below)
  - A '1' to the Forced Device Mode bit of the USB Global Configuration Register (FDMOD bit of the OTG\_FS\_GUSBCFG register) will force the OTG\_FS controller to operate in USB Device Mode (refer to Section 6.8.3 of the OTG version 1.3). In this case the state of the ID line will be ignored regardless of whether it is present or not.

*Notes: When designing a bus-powered Class B device that implements only device mode, an external voltage converter is required to take power from VBUS and supply the chip's VDD.*

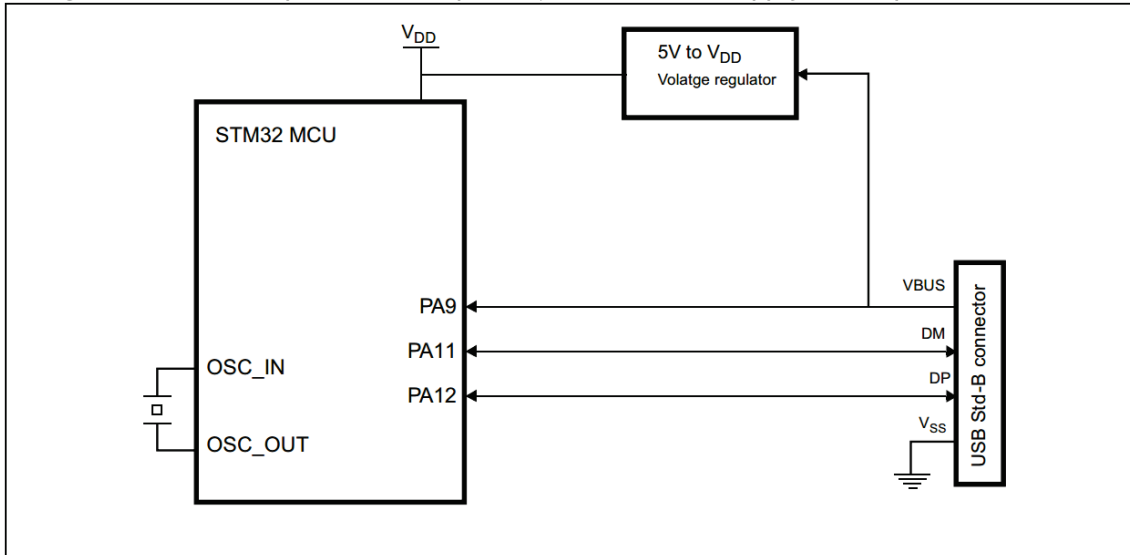


Figure 274 Simple USB peripheral connection

1. Designing a bus-powered device using a voltage converter
2.  $V_{DD}$  ranges from 2V to 3.6V.

### 29.5.1 SRP-Capable Devices

The SRP enable bit in the USB Global Configuration Register (SRPCAP bit in the OTG\_FS\_GUSBCFG register) allows the OTG\_FS controller to implement the Session Request Protocol (SRP). In this case, Class A devices can program the SRP protocol in the USB session. Refer to the "Session Request Protocol for Class B Devices" subsection of Section 29.17.8 for programming rules.

### 29.5.2 Device Status

#### Power-On State

The USB device enters the power-up state after a valid level for a Class B device is detected on the VBUS line (refer to Chapter 9.1 of USB 2.0). The OTG\_FS controller will automatically enable the pull-up resistor on the DP line to inform the host of the access to a full-speed device and generate a session request interrupt (SRQINT bit in the OTG\_FS\_GINTSTS register) to flag the entry into the power-up state.

The host is required to provide a valid VBUS level throughout the USB communication phase. If the level on the VBUS line is detected to be lower than the valid Class B level (e.g., a power supply perturbation has occurred, or the host side has turned off the power supply), the OTG\_FS

controller will automatically disconnect the USB connection and generate an end-of-session interrupt (the SEDET bit in the OTG\_FS\_GOTGINT register), signaling that the controller is exiting from the power-up state.

During the power-up state, the OTG\_FS controller expects to receive a reset signal from the host and all other USB operations are disabled. When a reset signal is received, a reset interrupt is generated (USBRST bit of the OTG\_FS\_GINTSTS register). When the reset is complete, an enumeration interrupt is generated (ENUMDNE bit of the OTG\_FS\_GINTSTS register), signaling the OTG\_FS controller to enter its default state.

#### Software Disconnection

The exit power-up state can be configured using software. Setting the software disconnect bit of the Device Control Register (SDIS bit of the OTG\_FS\_DCTL register) removes the pull-up resistor on the DP cable, allowing the host to recognize a device disconnect event even when the USB cable is still connected.

#### Default State

In the default state, the OTG\_FS controller expects to receive a SET\_ADDRESS command, at which point all other operations are disabled. Upon receiving a valid SET\_ADDRESS command, the application program is required to write the associated address into the Device Address bit of the Device Configuration Register (the DAD bit of the OTG\_FS\_DCFG register.) The OTG\_FS controller enters the address state and is ready to respond to transmissions sent from the host to this address.

#### Pending

The OTG\_FS controller continuously detects activity on the USB line. An early hang interrupt (ESUSP bit of the OTG\_FS\_GINTSTS register) is generated after 3ms of USB idle status is detected, and a hang interrupt (USBSUSP bit of the OTG\_FS\_GINTSTS register) is generated after 3ms to confirm the hang. The device hang bit of the Device Status Register (SUSPSTS bit of the OTG\_FS\_DSTS register) will be set automatically and the OTG\_FS controller enters the hang state.

The device can spontaneously come out of the hung state, which can be accomplished by setting the Remote Wake-Up Signal bit of the Device Control Register (the WKUPINT bit of the OTG\_FS\_DCTL register) and clearing the bit between 1 and 15ms.

When a recovery signal is detected from the host, a recovery interrupt is generated (RWUSIG bit in the OTG\_FS\_GINTSTS register) and the device's pending bit is automatically cleared.

## 29.5.3 Device Endpoint

The OTG\_FS controller supports the following USB endpoints:

- Control Endpoint 0
  - Bi-directional endpoint for processing control information only
  - For each of the IN and OUT directions, there is a separate set of registers to control the
  - This includes the control (DIEPCTL0/DOEPCTL0) registers, the transmission configuration (DIEPTSIZ0/DOEPTSIZ0) registers, and the status interrupt (DIEPINT0/DOEPINT0) registers. Some bits in the control and transfer length registers are different from those in the other endpoints.
- 3 IN endpoints
  - Each endpoint can be configured for synchronous, block or interrupt transmission
  - Each endpoint has control (DIEPCTLx) registers, transmission configuration (DIEPTSIZx) registers, and status interrupt (DIEPINTx) registers
  - Device IN Endpoint General Interrupt Mask Register (DIEPMSK) is used to enable/disable either type of endpoint interrupt source on all IN endpoints (including endpoint 0)
  - Supports an unfinished synchronous IN transmission interrupt (IISOIXFR bit of the OTG\_FS\_GINTSTS register), which is generated when there is at least one unfinished synchronous IN transmission in the current frame. This interrupt is generated with the Synchronized Frame Interrupt (EOPF of the OTG\_FS\_GINTSTS register)
- 3 OUT endpoints
  - Each endpoint can be configured for synchronous, block or interrupt transmission
  - Each endpoint has control (DOEPCTLx) registers, transmission configuration (DOEPTSIZx) registers, and status interrupt (DOEPINTx) registers

- Device OUT Endpoint General Interrupt Mask Register (DOEPMSK) is used to enable/disable either type of endpoint interrupt source on all OUT endpoints (including endpoint 0)
- The Outstanding Synchronized OUT Transmission Interrupt (INCOMPISOOUT bit of the OTG\_FS\_GINTSTS register) is supported and is generated when there is at least one synchronized OUT transmission outstanding for the current frame. This interrupt is generated with the cycle frame interrupt (EOPF of the OTG\_FS\_GINTSTS register)

### Endpoint Configuration

- The following configurations of the endpoints are realized by setting the endpoint IN/OUT control registers (DIEPCTLx/DOEPCTLx):
  - Endpoint Enable/Disable
  - Activate the endpoint in the current configuration
  - Configure USB transfer type (synchronous, block transfer, interrupt)
  - Configure maximum USB packet length
  - Configure the transmit FIFO number associated with an IN endpoint
  - Configure the PID packet DATA0/DATA1 that is expected to be received or will be sent (valid only for block transfers and interrupt transfers)
  - Configure the odd/even frames of transmission frames to be sent or received (valid for synchronous transmission only)
  - Optional configuration: set the NAK bit to respond to host NAK regardless of FIFO status
  - Optional configuration: set the STALL bit to respond to any command sent by the host with a STALL
  - Optional configuration: set the OUT endpoint SNOOP mode to not examine CRC for received data

### endpoint transmission

The Device Endpoint Transmission Length Register (DIEPTSIZx/DOEPTSIZx) is used to configure the transmission length and read the transmission status. Configuration of this register must be done before setting the enable bit of the port control register. Once the endpoint is enabled, the register is read-only and only the OTG\_FS controller can update the transfer status bits of the register.

- The transmission parameters to be configured are as follows:
  - Length of a single transmission in bytes
  - Number of USB packets that complete the entire transfer

### Endpoint Status/Interrupt

The device endpoint interrupt registers (DIEPINTx/DOEPINTx) indicate the status of USB and AHB-related events at the endpoint. When the OUT endpoint interrupt bit of the controller interrupt register or the IN endpoint interrupt bit (OEPINT bit and IEPINT bit of the OTG\_FS\_GINTSTS register) is set, the application program needs to read the Device All Endpoint Interrupt Register (DAINT) to determine the endpoint number where the interrupt has occurred, and then read the Device Endpoint Interrupt Register to get the detailed information about the interrupt. The application program must then clear the corresponding interrupt bit of this register to clear the corresponding interrupt bits of the DAINT register and the GINTSTS register.

- The device controller provides the following status bits and interrupt events:
  - The Transfer Complete interrupt indicates that the data transfer is complete on both the AHB and USB sides.
  - SETUP phase complete (valid for control endpoints only)
  - The associated transmission FIFO is half-empty or fully empty (IN endpoint)
  - NAK response has been sent to the host (valid only for synchronized IN transmissions)
  - When the transmit FIFO is empty, the host has issued an IN request (valid only for block transfer IN/interrupt IN transfers)
  - The host issued an OUT request when the endpoint was not enabled
  - Crosstalk error detected
  - The software disables endpoints
  - Software sets endpoint state to NAK (valid only for synchronized IN transmission)
  - More than 3 consecutive SETUP packets received (valid for control OUT transmissions only)
  - Timeout error detected (valid only for control IN transmission)
  - Synchronization OUT packet lost, no interrupt generated

## 29.6 USB Host

This chapter describes the functions of the OTG\_FS controller when it is operating in USB host mode. The OTG\_FS controller operates in host mode under the following conditions:

- OTGA Class Host
  - Plugged into the A end of the USB cable, the default is OTGA class host mode
- OTGB Class Host
  - OTGB class device switches to host role via HNP protocol
- Category A equipment
  - In the presence of the ID line, the insertion is at the A end of the USB cable, and at the same time the HNP bit of the USB Global Configuration Register is '0' (HNPCAP bit of the OTG\_FS\_GUSBCFG register). At this point, the built-in pull-up resistor on the DP/DM line is automatically valid.
- As a host only (please refer to the picture below)
  - The Forced Host Mode bit of the USB Global Configuration Register (FHMOD bit of the OTG\_FS\_GUSBCFG register) can force the USB host mode in which the OTG\_FS controller operates. In this case, even if the ID line on the USB connector is present, it is still invalid. The built-in pull-up resistor on the DP/DM line is automatically valid.

**Notes:**

1. The controller cannot supply 5V VBUS, so an external charge pump is required. If the application board can supply 5V, a basic switching power supply can be used to drive the 5V VBUS line. The external charge pump can be controlled with any of the general purpose I/O ports. This is required when designing OTGA class hosts, class A devices and USB hosts.
2. During USB communication, the charge pump needs to keep the power supply on the VBUS line at all times. The overcurrent output signal should be connected to an I/O port configured as an interrupt input, which can generate an interrupt to cut off the power supply to the VBUS line in time in case of an overcurrent event.

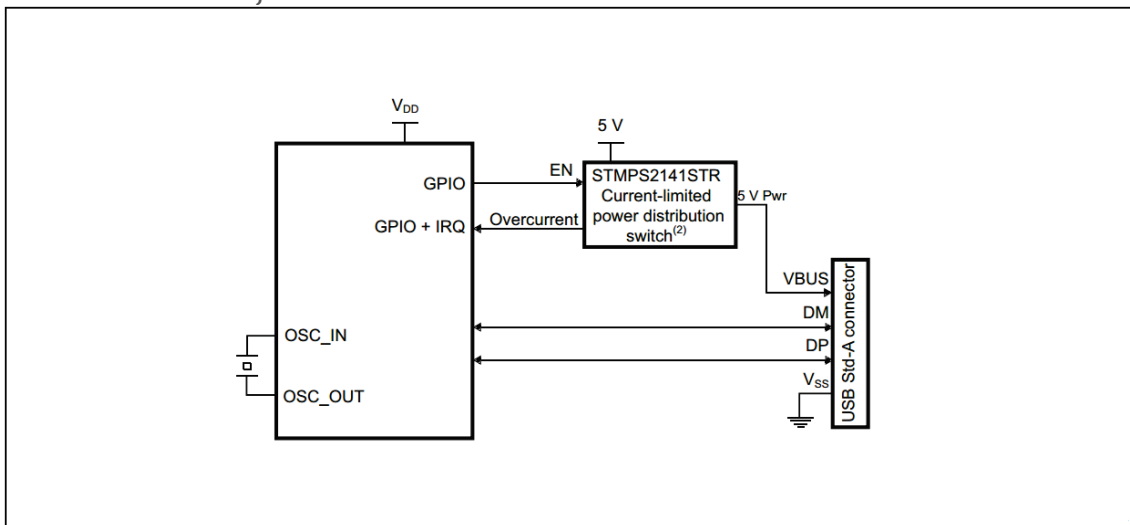


Figure 275 Simple USB Host Connection

1. If the designed USB host supports VBUS bus-powered devices, an external ST20x2 is required. If the application board can provide 5V power supply, an ordinary power switch can also be used.
2. VDD ranges from 2V to 3.6V.

### 29.6.1 SRP-Capable Hosts

The SRP function can be supported by configuring the SRP enable bit of the USB global configuration register (SRPCAP bit of the OTG\_FS\_GUSBCFG register). With the SRP function enabled, the host can turn off the power supply to the VBUS line to save power consumption when the USB session hangs.

Refer to the "Session Request Protocol for Class A Devices" subsection of Section 29.17.8 for the programmed implementation of the SRP feature.

## 29.6.2 USB Host Status

### Host Port Power

The controller does not provide a 5V supply for VBUS internally, so an external charge pump is required to power the VBUS line, or an ordinary power switch can be used if a 5V supply is available on the application board. This external charge pump can be controlled through a common I/O port. When the application program needs to set this ordinary I/O port to supply power to the VBUS line, it also needs to set the power supply bit of the host port control and status register (PPWR bit of the OTG\_FS\_HPRT register).

### VBUS Line Enable

The VBUS line needs to be guaranteed to be at a valid level at all times throughout USB communication. Any event that causes the level of the VBUS line to drop below the threshold (4.25V) will result in an OTG interrupt triggered by the session abort bit (SEDET bit in the OTG\_FS\_GOTGINT register). At this point, the application needs to turn off power to the VBUS and clear the port power supply bit. An overcurrent signal from the charge pump can also be used to prevent electrical damage. Connect this overcurrent signal to an I/O port set as an interrupt input, and as soon as an overcurrent event is generated, the application program needs to turn off power to the VBUS and clear the port power supply bit in the interrupt.

### Host Monitoring of Device Access

If the VBUS line never has a valid level (4.75V), the OTG\_FS controller will not be able to monitor the insertion of a USB device or a Class B device whenever it is inserted.

When the VBUS line is within the valid level range, the OTG\_FS controller generates a host port interrupt triggered by the device insertion bit (PCDET bit of the OTG\_FS\_HPRT register) as soon as a Class B device is inserted.

### Host Monitoring of Device Disconnections

Disconnecting an inserted device will generate a disconnect interrupt triggered by a device disconnect event (DISCINT bit in the OTG\_FS\_GINTSTS register).

### Host Enumeration

Once the host detects that a device has been inserted, it needs to enter the enumeration process and send USB reset and configuration commands to the newly inserted device.

Since the inserted device has pull-up resistors on the DP (full-speed device) or DM (low-speed device) lines, this can cause the bus to become unstable. When the bus returns to a stable state again, the end-of-backtrack OTG interrupt is triggered (DBCDNE bit of the OTG\_FS\_GOTGINT register), and only then can the host send a USB reset command to the device.

The application sends a USB reset signal (single-ended 0) to the USB bus by setting the port reset bit of the host port control and status register (PRST bit of the OTG\_FS\_HPRT register).

The application program needs to ensure that this reset signal is maintained between 10ms and 20ms and that the port reset bit is cleared in a timely manner.

Once the USB reset sequence is complete, a host port interrupt triggered by the port enable/disable change bit is generated (PENCHNG bit of the OTG\_FS\_HPRT register). This interrupt informs the application that it can obtain speed information for the enumerated device from the host port control and status registers (PSPD bit of the OTG\_FS\_HPRT register), and with that, the host will either send a SOF (full speed device) signal or remain active (low speed state). At this point, the host can send configuration commands to the device to complete the device enumeration.

### Host Hangs

An application program can suspend USB activity by setting the port suspend bit of the host control and status register (the PSUSP bit of the OTG\_FS\_HPRT register). At this point, the OTG\_FS controller will stop sending the SOF signal and enter the hang state.

The controller may optionally support a remote device to wake up the host to exit the hung state. In this case, the controller will generate a remote wake-up interrupt (WKUPINT bit in the OTG\_FS\_GINTSTS register) when it detects a remote wake-up signal, and then the wake-up bit in the host port control and status register (PRES bit in the OTG\_FS\_HPRT register) will be automatically set, and the controller will automatically send a wake-up signal to the USB bus.



The application program needs to control the duration of the wake-up signal and clear the port wake-up bit in time to exit the suspend state and restart sending SOF signals.

If it is necessary for the host to spontaneously exit the pending state, the application program can set the port wake-up bit, at which time the wake-up signal will be sent to the USB bus, and the application program needs to control the time this wake-up signal is maintained and clear the port wake-up bit in a timely manner.

### 29.6.3 Host Channel

The OTG\_FS controller provides 8 host channels, each corresponding to one USB host transfer (USBPIPE). The host does not support initiating more than 8 transfer requests at the same time. If the application program has more than 8 unprocessed transfer requests, the Host Controller (HCD) must reschedule the channels after the channels become valid again, i.e., after it receives an interrupt for transfer completion and channel abort.

Each channel can be configured for input/output mode or for any periodic/non-periodic transmission type. Each channel has control registers (HCCHARx), transmission configuration registers (HCTSIZx) and status/interrupt registers (HCINTx) and corresponding mask registers (HCINTMSKx).

#### Host Channel Control

- The Channel Control Register (HCCHARx) can provide the following operations:
  - Channel Enable/Disable
  - Configure transfer speed for connected USB devices: full speed/low speed
  - Configure the address for the connected USB device
  - Configure endpoint numbers for connected USB devices
  - Configure transmission direction: input/output
  - Configure transfer types: control, block transfer, interrupt, synchronization
  - Configure the maximum packet length (MPS)
  - Configure periodic transmission in odd/even frames

#### Host Channel Transmission

The application program configures the transmission length through the host channel transmission length register (HCTSIZx) and reads back the transmission status. It is necessary to configure the transfer length before enabling a transfer channel. Once the channel is enabled, the transfer length register becomes read-only and only the OTG\_FS controller can update the register with the current transfer status.

- The following transmission parameters can be configured:
  - Transmission length of a single packet in bytes
  - The number of packets in the entire transmission
  - Starting data PID

#### Host Channel Status/Interrupts

The host channel x interrupt register (HCINTx) indicates channel events related to USB and AHB. When the host channel interrupt bit of the host controller interrupt register (HCINT bit of the OTG\_FS\_GINTSTS register) is set, the application program needs to read the host all channel interrupt register (HCAINT) first to get the channel number of the channel that generated the host channel interrupt, and then read the host channel x interrupt register (HCINTx) based on the channel number to get the information about the interrupt. The application program needs to clear the corresponding bits of the host channel x interrupt register (HCINTx) in order to clear the corresponding bits of the HCAINT and GINTSTS registers. The OTG\_FS\_HCINTMSKx register can be used to mask the interrupt source for each channel.

- The host controller provides the following status queries and interrupts:
  - The Transfer Complete interrupt indicates that the data transfer has been completed on both the application program side (AHB) and the USB side.
  - Channel abort due to transfer completion, USB communication error or application program inhibit command
  - The corresponding transmission FIFO is half-empty or fully empty (valid only for IN channels)
  - ACK response received
  - NAK response received

- STALL response received
- USB transfer errors caused by CRC errors, timeouts, bit-fill errors, or incorrect EOPs
- crosstalk error
- frame overflow
- Data PID (DATA0/DATA1) rollover error

## 29.6.4 Host Scheduler

The host controller has a built-in hardware scheduler for spontaneously managing and driving application-initiated transfer requests. At the beginning of each frame, the host will process periodic (synchronous and interruptive transfers) data transfers before non-periodic (control and bulk transfers) data transfers in order to comply with the USB specification's high-priority guarantee for synchronous and interruptive transfers.

The host manages the USB transfer channel through request queues (a periodic queue and an acyclic queue). Each request queue can hold up to 8 requests, and each request represents an unprocessed transfer made by the application program. Each request in the request queue carries the IN/OUT direction, channel number, and other information required to perform a USB transfer. The order in which requests are written to the request queue determines the sequence of transfers on the USB bus. At the beginning of each frame, the host will process the periodic request queue first and then the non-periodic request queue. If at the end of a frame there are still unprocessed requests for synchronization or interrupt transfers, an incomplete periodic transfer interrupt will be generated (IPXFR bit in the OTG\_FS\_GINTSTS register).

The management of the periodic and non-periodic queues is done entirely by the OTG\_FS controller. The application program can obtain status information for each request queue through read-only registers.

- The Periodic Transfer FIFO and Queue Status Register (HPTXSTS) and the Non-Periodic Transfer FIFO and Queue Status Register (GNPTXSTS) are included:

- Number of requests that can also be inserted in the periodic and non-periodic queues (maximum 8)
- Periodic (non-periodic) sending of the remaining free space in the FIFO (for OUT transfers)
- IN/OUT commands, host channel count and other status information

Since each request queue can hold up to 8 requests, applications can utilize this feature as much as possible by submitting up to 8 periodic transfer requests and 8 non-periodic transfer requests to improve transfer efficiency. For example, for block transmission of OUT/IN data, the application can be configured to transmit up to 64 (maximum number of bytes per packet) x 8 (maximum number of requests) = 512 bytes of data, realizing the maximum data transmission rate for a full-speed device. This data will be automatically scheduled by the host without the need for the application to insert management.

- The application program initiates a periodic (non-periodic) OUT transfer request to the host scheduler by following these steps:

- Configure the transmission parameters of the host channel
- Enable configured channels
- Read the HPTXSTS bit of the OTG\_FS\_GNPTXSTS register to ensure that the periodic (non-periodic) queue can still hold at least 1 request.
- Read the HPTXSTS (GNPTXSTS) register to ensure that there is enough space remaining in the periodic (non-periodic) transmit FIFO (see Section 26.11.2: Transmit FIFO in Host Mode). This step can be omitted if the application submits the transfer request after receiving a half-empty or full-empty interrupt in the periodic (non-periodic) transmit FIFO.
- Load data into the appropriate FIFO (PUSH register). Each channel is configured with a PUSH register. The data will be automatically loaded into the appropriate periodic or non-periodic transmit FIFO depending on the EPTYPE bit of the OTG\_FS\_HCCHARx register. When the last 32 bits of data have been written into the FIFO, a request will be inserted into the end of the request queue, waiting to be scheduled for execution.

- The application program initiates a periodic (non-periodic) IN transfer request to host scheduling by following these steps:

- Configure the transmission parameters of the host channel
- Set the channel enable bit of the host channel control register (CHENA bit of the OTG\_FS\_HCCHARx register) to enable the configured channel. This operation inserts a



transfer request at the end of the periodic (non-periodic) request queue and waits for scheduling to execute.

## 29.7 SOF Trigger

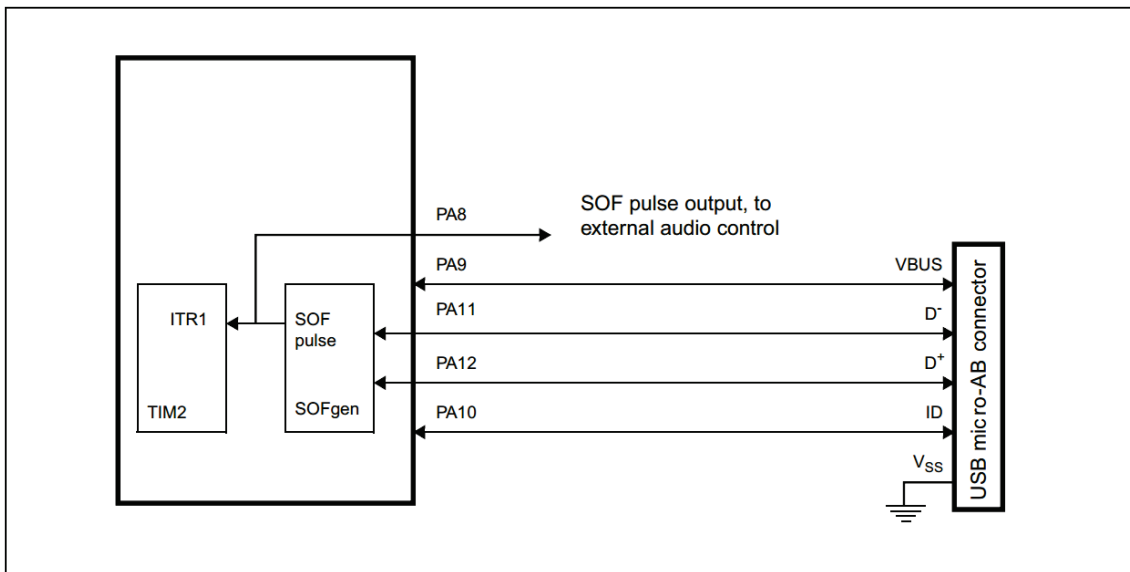


Figure276 SOF Connections

OTG\_FS controller provides:

- Monitoring, Tracking and Configuration of SOF
- SOF pulse output

These features are useful for clocking out audio applications as the audio device needs to be synchronized with the PC's data stream, or the host needs to adjust the frame rate to the needs of the audio device.

### 29.7.1 Host SOFs

In host mode, the number of PHY clocks between two consecutive SOFs (full-speed devices) or remain active (low-speed devices) can be set by configuring the Host Frame Interval Register (HFIR), which can therefore be used by the application program to control the SOF frame period. An interrupt can be generated at each frame capital (SOF bit in the OTG\_FS\_GINTSTS register). Both the current frame number and time will remain unchanged until the next frame number appears in the Host Frame Number Register (HFNUM).

Each SOF capital will generate a SOF pulse signal with a width of 12 system clock cycles, and this pulse can be configured to be output on the SOF pin via the SOFOUTEN bit in the Global Control and Configuration register. The SOF pulse is also internally connected to the trigger input of Timer 2 (TIM2), so input capture, output compare and timer can be triggered by the SOF pulse. The connection of the SOF pulse to TIM2 can be enabled via 'bit 29' of the REMAP\_DBGAFR register.

### 29.7.2 Device SOFs

In device mode, a frame header interrupt is generated each time a SOF is received on the USB line (SOF bit of the OTG\_FS\_GINTSTS register). The current frame number can be read from the device status register (FNSOF bit of the OTG\_FS\_DSTS register). At this point, a SOF pulse signal with a width of 12 system clock cycles is generated, and this pulse signal can be output on the SOF pin by enabling the SOF output enable bit of the Global Control and Configuration Register (SOFOUTEN bit of the OTG\_FS\_GCCFG register). The SOF pulse signal is also internally connected to the trigger input of Timer 2 (TIM2), and this The connection is enabled by 'bit 29' of the REMAP\_DBGAFR register, at which point the input capture, output compare and timer can all be triggered by the SOF pulse.

The periodic end-of-frame interrupt (EOPF bit of the GINTSTS register) is used to inform the application program that 80%, 85%, 90%, or 95% of the frame has been completed at intervals that depend on the Frame Interval bit of the Device Configuration Register (PFIVL bit of the

OTG\_FS\_DCFG register). This function is used to determine if all synchronization transmissions within a frame have been completed.

## 29.8 Power Supply Options

Table153 W55MH32 Low Power and OTG Compatibility

paradigm	descriptive	USB compatibility
(of a computer) run	MCU fully activated	Required when the USB is not in the suspended state.
sleep	USB Suspend Exit causes the device to exit sleep mode. The peripheral register contents will be retained.	Available when USB is suspended.
cessation	USB pause exit causes the device to exit stop mode. Peripheral register contents reserved <sup>(1)</sup>	Available when USB is suspended.
pragmatic	Turn off the power. After exiting standby mode, the peripheral devices must be reinitialized.	Not compatible with USB applications.

(1). There are different possible settings in stop mode. There may also be limitations, see Section 5: Power Control (PWR) to see which (if any) limitations apply when using OTG.

The power consumption of the OTGPHY is determined by the following three bits of the General Purpose Controller Configuration Register:

- PHY power supply bit (PWRDWN bit of the GCCFG register)
  - Toggles the PHY full speed transceiver module on/off. Must be pre-set to allow USB communication.
- Class A VBUS monitor enable bit (VBUSASEN bit of the GCCFG register)
  - Toggles the Class A device VBUS comparator on/off. When in Class A device mode (USB host), this bit must be enabled in the HNP stage.
- Class B VBUS monitor enable bit (VBUSASEN bit of the GCCFG register)
  - Toggles the Class B device VBUS comparator switch. When in Class B device mode (USB device), this bit must be enabled in the HNP stage.

Suspending USB when there is no USB communication or no USB device is connected saves power.

- Stop PHY clock (STPPCLK bit of OTG\_FS\_PCGCCTL register)
  - When the Stop PHY Clock bit of the Clock Gating Control Register is set, most of the 48MHz clock that is internally connected to the OTG Full Speed Controller is turned off via gating, at which point the dynamic power consumption caused by the USB clock is greatly reduced, even though the application program still enables the 48MHz input clock.
  - Most of the transceivers will be disabled, but the circuitry used to detect asynchronous reset signals and remote wake-up events will remain active.
- Gating HCLK (GATEHCLK bit of the OTG\_FS\_PCGCCTL register)
  - Most of the system clocks internally connected to the OTG\_FS controller are turned off by the gating circuitry when the Gating HCLK bit of the Clock Gating Register is set. Only the register access interface remains active. At this point, even though the application program still enables the system clock, the dynamic power consumption caused by the USB clock is greatly reduced.
- USB system stop
  - When the OTG\_FS controller is in the USB suspend state, the application needs to completely shut down all system clock sources to completely reduce all power consumption. Setting the PHY clock stop bit first and then setting the power supply control system module (PWR) into deep system sleep mode will stop the USB system.
  - Upon detecting a remote wake-up (as host) or a wake-up signal from the USB bus (as device), the OTG\_FS controller will automatically resume system activity and clocking of the USB.

In order to reduce power consumption, the clock is provided to the USB data FIFO only when the OTG\_FS controller needs to access the FIFO.

## 29.9 Dynamic Update of OTG FS HFIR Registers

The USB core embeds dynamic pruning of the SOF frame period in host mode, allowing external devices to be synchronized with the SOF frame. When the OTG\_FS\_HFIR register is changed within the current SOF frame, the SOF period correction is applied in the next frame, as shown in Figure 308

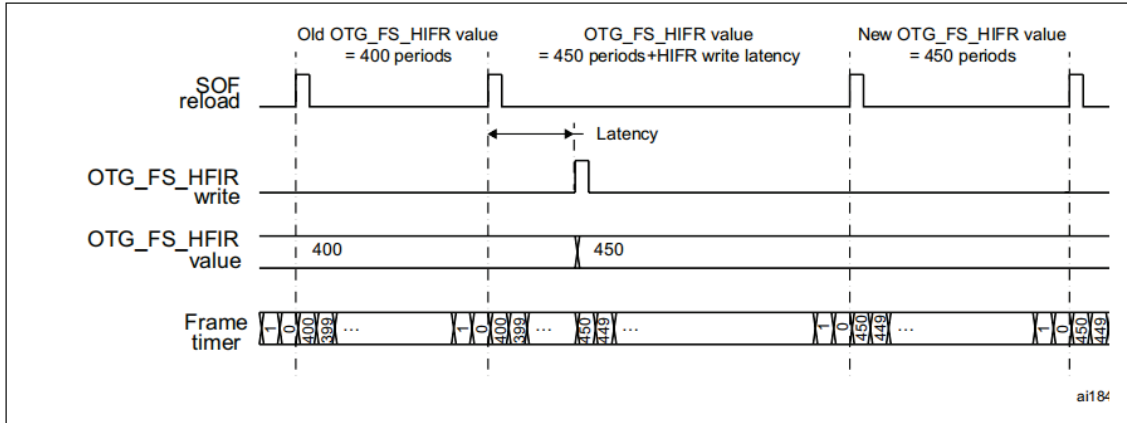


Figure277 Dynamic Update OTG FS HFIR

## 29.10 USB Data FIFOs

The USB system plans 1.25K bytes of dedicated RAM for FIFO management. The Packet FIFO Controller (PFC) module of the OTG\_FS controller divides this RAM into a send FIFO area and a receive FIFO area. The application program temporarily caches the data into the send FIFO and waits for it to be sent, while the data received from the USB bus is temporarily cached in the receive FIFO, waiting to be read by the application program. The actual number of FIFOs and how they are programmed in the dedicated RAM is determined by the actual application. For example, in device mode, each IN endpoint is configured with a transmit FIFO, and the sizes of these FIFOs can be specified by the software so that the dedicated RAM can be optimized to meet the needs of the application.

## 29.11 FIFO Structure in Device Mode

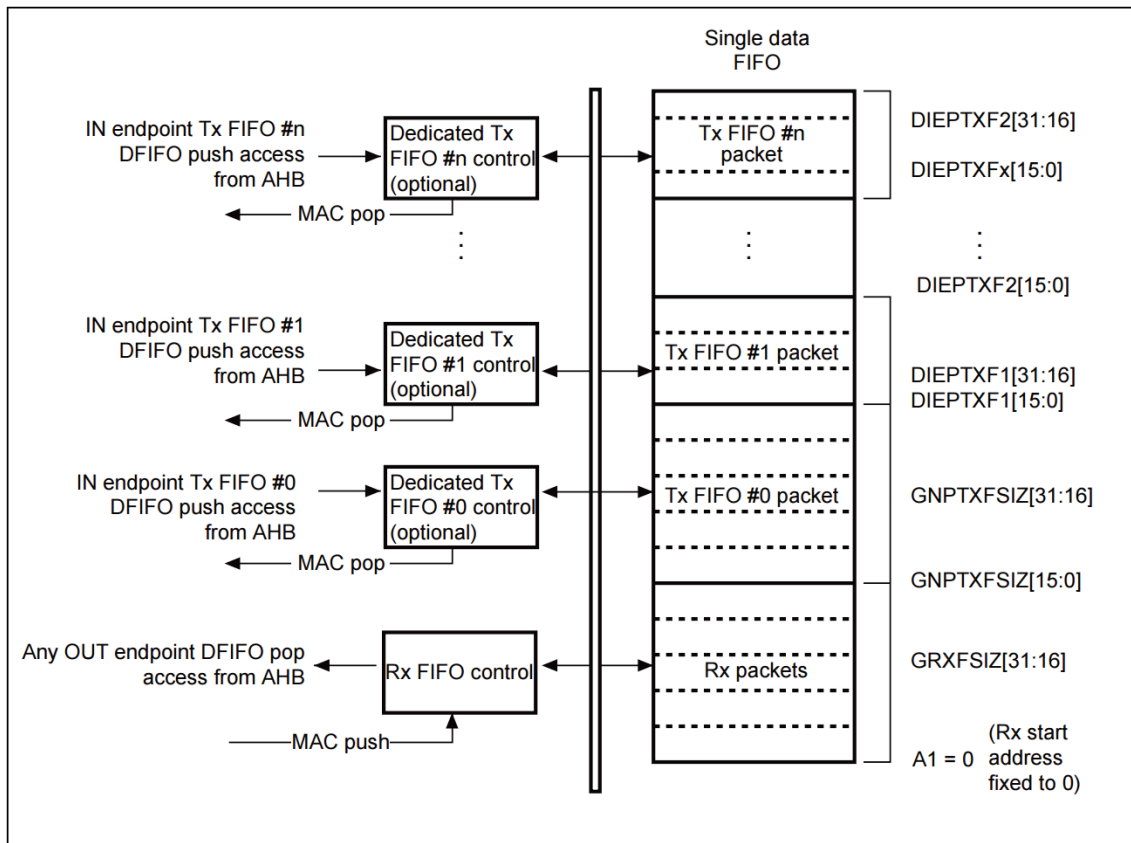


Figure 278 FIFO address image in device mode and FIFO operation image in AHB

### 29.11.1 Receive FIFO in Device Mode

The FS\_OTG controller caches data from all OUT endpoints in device mode using a separate FIFO. Received packets are sequentially cached in the available space of the RXFIFO. The status of received packets (including OUT endpoint number, byte count, data PID number and validity of received data) is written by the PFC module before the received data. If there is no space remaining in the receive FIFO, the controller responds to the host with a NACK and generates an interrupt for the appropriate endpoint. The size of the receive FIFO is configurable by the receive FIFO length register (GRXFSIZ).

The architecture using a single RXFIFO allows USB devices to utilize the entire receive RAM space more efficiently.

- All OUT endpoints share the entire RAM space (shared FIFO)
- The OTG\_FS controller can maximize the utilization of the receive FIFO in the order in which the host sends OUT data.

The application continuously receives receive receive FIFO out-of-air breaks (RXFLVL bit of the OTG\_FS\_GINTSTS register) as long as there is at least one packet still waiting to be fetched by the application in the receive FIFO. The application program can obtain packet information by reading the receive status read and pop registers (GRXSTSP) and fetch packets from the receive FIFO by reading the POP registers of the corresponding endpoints.

### 29.11.2 Transmit FIFO in Device Mode

The mode of shared FIFOs is not applicable to IN transmissions. the only way to transfer packets from all endpoints in order into the same transmit FIFO is to know the order of host requests in advance or to be able to predict the processing of the process. Therefore, in device mode, the controller configures a dedicated FIFO for each IN endpoint. the application program can configure the FIFO length for IN endpoint 0 through the Non-Cyclic Transmission FIFO Length Register (GNPTXFSIZ), and the FIFO length for IN endpoint x through the Device IN Endpoint Transmission FIFO Register (DIEPTXFx).

Dedicated FIFOs have a very flexible architecture that greatly reduces application load. These FIFOs have no request sequence and there is no need to predict the order of accesses when a USB host accesses a non-periodic endpoint.

Depending on the value of the Non-Cyclic Transmit FIFO Empty Level bit (TXFELVL bit of the OTG\_FS\_GAHBCFG register) configured in the AHB Configuration Register, the OTG\_FS controller generates a Transmit FIFO Empty Break (NPTXFE bit of the OTG\_FS\_GINTSTS register), which signals that the Transmit FIFO corresponding to the IN endpoint has been either half empty or The application program first reads all the endpoints of the device. The application program first reads the device all endpoint interrupt register (DAINT) to get the number of the IN endpoint that needs to be serviced, then reads the transmit FIFO status register (DTXFSTSx) of the IN endpoint x of the device to check if there is enough space left in the FIFO, and then finally writes the PUSH register of the corresponding endpoint to transfer data to the transmit FIFO x. The application program then writes the transmit FIFO status register of the corresponding endpoint (DTXFSTSx) to check if there is enough space left in the FIFO.

## 29.12 FIFO Structure in Host Mode

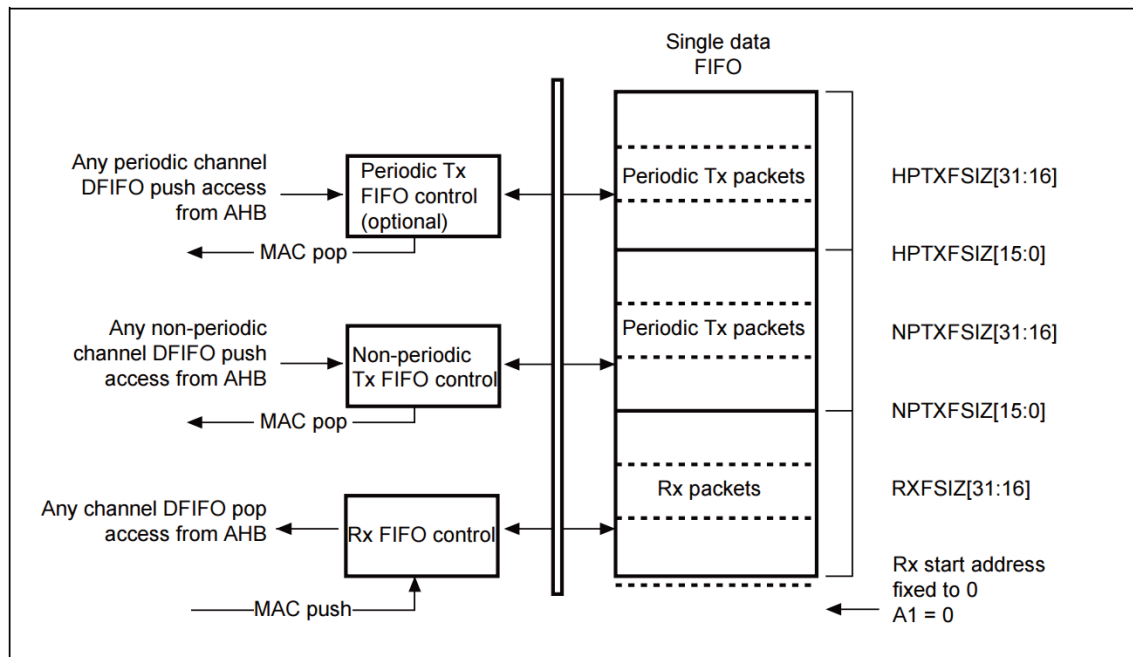


Figure 279 Host mode FIFO address image and FIFO operation image for AHBs

### 29.12.1 Receive FIFO in Host Mode

All periodic and non-periodic transfers are managed in host mode using a receive FIFO that is used to temporarily store data received from the USB bus that has not yet been transferred to system memory (received packets). Packets from any of the remote IN endpoints are sequentially staged in the FIFO. The status of the received packet, including the host channel number, number of bytes, data PID, and the validity of the received data are also stored together in the FIFO. The application program can configure the length of this receive FIFO through the receive FIFO length register (GRXFSIZ).

The architecture of using a single receive FIFO allows the USB host to utilize the entire receive RAM space more efficiently.

- All configured IN channels share the entire RAM space (shared FIFO)
- The OTG\_FS controller can maximize the utilization of the receive FIFOs in the sequence in which the host sends the IN command

The application receives a receive FIFO out-of-air disconnect as long as the receive FIFO still has at least one packet waiting to be fetched by the application. The application program can obtain packet information by reading the receive status read and pop registers and fetch packets from the receive FIFO by reading the POP registers of the corresponding endpoints.

## 29.12.2 Transmit FIFO in Host Mode

In host mode, the controller uses one transmit FIFO to manage all non-periodic (control and block transfers) OUT transfers and another transmit FIFO to manage all periodic (synchronization and interrupt) OUT transfers. These two FIFOs are used to temporarily store data that needs to be sent to the USB bus (sent packets). The length of the periodic (non-periodic) transmit FIFOs can be configured through the Host Periodic (Non-Periodic) Transmit FIFO Length Register (HPTXFSIZ/GNPTXFSIZ).

The reason for using two transmit FIFOs is to ensure high priority for periodic transmissions in a USB frame. At the beginning of each frame, the built-in host scheduler processes the periodic request queue before the non-periodic request queue.

The use of two transmit FIFOs allows the USB host to easily separate and optimize the management of cyclic and non-cyclic data caches.

- All host channels used for periodic (non-periodic) OUT transfers share the same RAM buffer (shared FIFO)
- The OTG\_FS controller can maximize the use of the periodic (non-periodic) transmit FIFO based on host software scheduling of OUT commands.

Depending on the value of the Periodic Transmit FIFO Empty Flag bit (PTXFELVL bit of the OTG\_FS\_GAHBCFG register) configured in the AHB Configuration Register, the OTG\_FS controller generates a Periodic Transmit FIFO Empty Break (PTXFE bit of the OTG\_FS\_GINTSTS register) to indicate that the Periodic Transmit FIFO has been half-empty or fully empty. The application program needs to read the periodic transmit FIFO first. The application program needs to read the Periodic Transmit FIFO and Queue Status Registers (HPTXSTS) first to get the information whether there is any space left in the Periodic Transmit FIFO and the Periodic Request Queue, and the application program can write data into the FIFO only if there is space left in both the Transmit FIFO and the Request Queue.

Depending on the value of the Non-Cyclic Transmit FIFO Empty Level bit (TXFELVL bit of the OTG\_FS\_GAHBCFG register) configured in the AHB Configuration Register, the OTG\_FS controller generates a Non-Cyclic Transmit FIFO Empty Break (NPTXFE bit of the OTG\_FS\_GINTSTS register) to indicate that the Non-Cyclic Transmit FIFO has been half-empty or fully empty. The application program needs to read the nonperiodic transmit FIFO and queue status registers (GNPTXSTS) first to get information about whether there is any space left in the nonperiodic transmit FIFO and the nonperiodic request queue, and the application program can write data to the FIFOs only if there is space left in both the transmit FIFO and the request queue.

## 29.13 FIFO RAM Allocation

### 29.13.1 Device Mode

**Receive FIFO RAM Allocation:** The application should allocate RAM for SETUP packets: Ten locations must be reserved in the receive FIFO to receive SETUP packets at the control endpoint. The core will not use these reserved locations for SETUP packets to write any other data. One location should be allocated for global OUTNAK. status information is written to the FIFO for each received packet. therefore at least  $(\text{maximum packet size}/4)+1$  space must be allocated to receive the packet. If multiple isochronous endpoints are enabled, at least two  $(\text{maximum packet size}/4)+1$  spaces must be allocated to receive consecutive packets. Typically, it is recommended that two  $(\text{maximum packet size}/4)+1$  spaces be allocated so that the USB can receive subsequent packets while the previous packet is being transmitted to the CPU.

Along with the last packet from each endpoint. transmission-complete status information is also pushed to the FIFO. typically. it is recommended that each OUT endpoint have a location.

**Transmit FIFO RAM Allocation:** Minimum RAM space required for each IN endpoint. The transmit FIFO is the maximum packet size for a particular IN endpoint.

*Notes: More space allocated in the transfer IN endpoint FIFO will result in better USB performance.*

## 29.14 USB System Performance

By configuring a large-capacity RAM buffer, high-freedom FIFO length configuration, 32-bit fast access to the AHBPU/POP registers, and an advanced FIFO control mechanism, the OTG\_FS controller is able to maximize the use of the RAM space without having to worry about the order of the USB data, and the USB system achieves the optimal performance.

These advantages are listed below:

- Reduced load on applications, no need to intervene in USB transfers, optimized CPU bandwidth usage
  - When data is transferred efficiently via the USB system, applications can prepare large amounts of transferred data in advance.
  - The application gets a lot of time to fetch data from a single receive FIFO
- USB controllers can control the full efficiency of their own work, meaning that today's architectures provide the highest full-speed bandwidth and the greatest degree of autonomy compared to requiring application intervention.
  - The USB controller manages a large data buffer and can autonomously manage the transfer of data on the USB bus.
  - The USB controller manages a large data receiving buffer, which can be autonomously received from the USB bus and filled into the buffer.

Since the OTG\_FS controller can use the 1.25K bytes of RAM buffer very efficiently, and since 1.25K bytes is more than enough for a full-speed frame to manage the sent/received data, the USB system is able to provide the maximum full-speed data transfer rate within each USB frame (1ms).

## 29.15 OTG\_FS Interrupt

Regardless of whether the OTG\_FS controller is operating in device mode or host mode, the application program cannot access register groups in the other mode. If the application program has an illegal access, a mode mismatch interrupt is generated and affects the corresponding bit in the controller interrupt register (MMIS bit in the OTG\_FS\_GINTSTS register). When the controller is switched from one mode to another, the register groups in the new mode all need to be re-initialized in the same way as during a power-on reset.



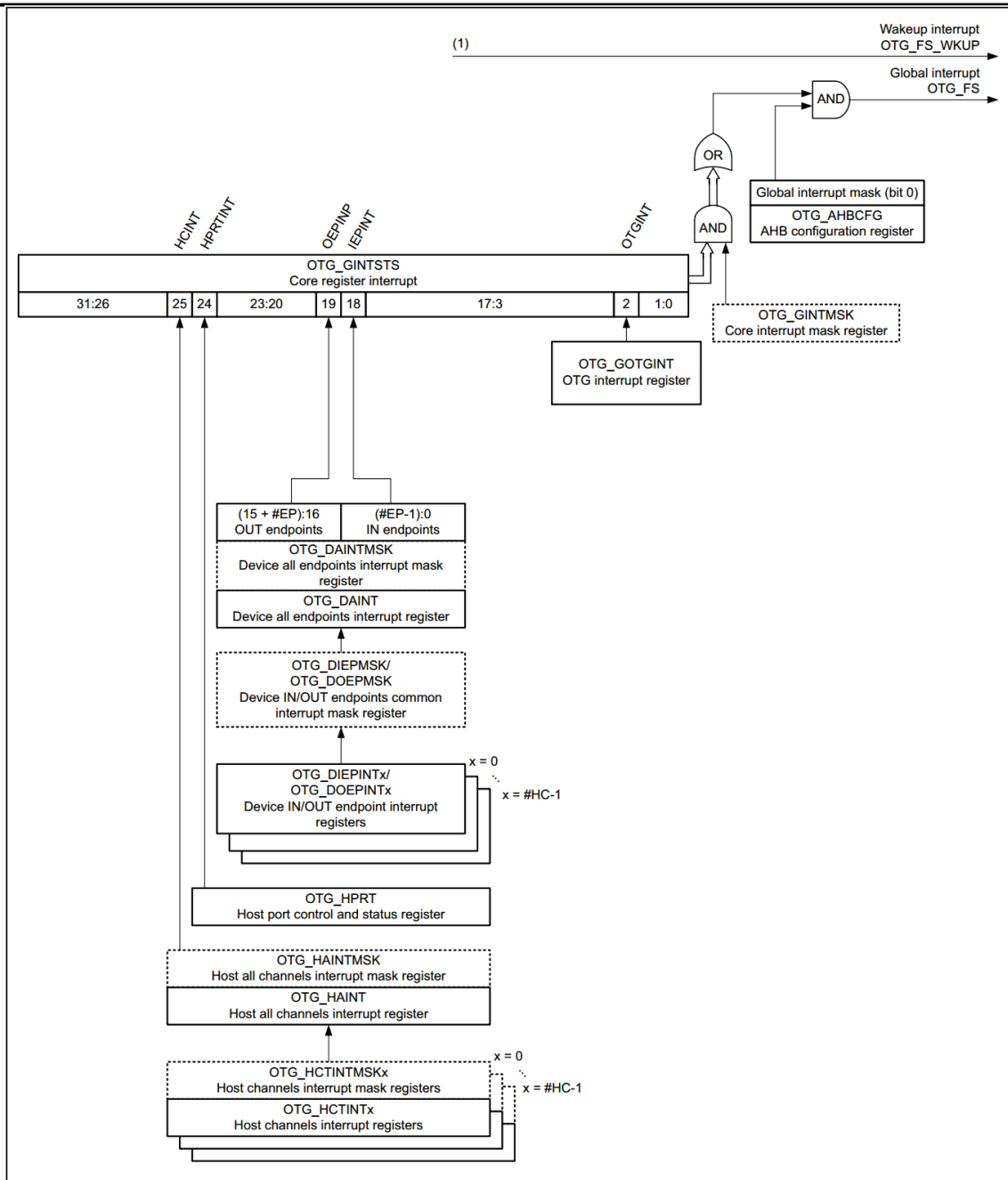


Figure280 Interrupt Architecture

Refer to the relevant bits of the controller interrupt registers

## 29.16 OTG\_FS Control and Status Registers

The application program reads and writes the control and status registers (CSR<sub>x</sub>) through the AHB slave interface to enable control of the OTG\_FS controller. These registers are 32-bit accessible and 32-bit address aligned. The following register groups are included:

- Controller Global Register Set
- Host Mode Register Group
- Host Global Register Set
- Host Port CSR Register Group
- Host Channel Related Register Groups
- Device Mode Register Group
- Device Global Register Set
- Device Endpoint Related Register Groups
- Power Supply and Clock Control Register Group
- Data FIFO (DFIFO) access register set



Only the controller global registers, power supply and clock control registers, data FIFO access registers, and host port control and status registers are valid in both host and device modes. Regardless of whether the OTG\_FS controller is operating in host mode or device mode, the application program cannot access register groups in the other mode. If an illegal access occurs by the application program, a mode mismatch interrupt is generated and affects the corresponding bit of the controller interrupt register (the MMIS bit of the OTG\_FS\_GINTSTS register). When the controller is switched from one mode to another, the register groups in the new mode all need to be reinitialized in the same way as during a power-on reset. These peripheral registers must be operated in word (32-bit) format.

## 29.16.1 CSR Memory Image

The host mode registers and device mode registers occupy different addresses. All registers are driven by the AHB clock.

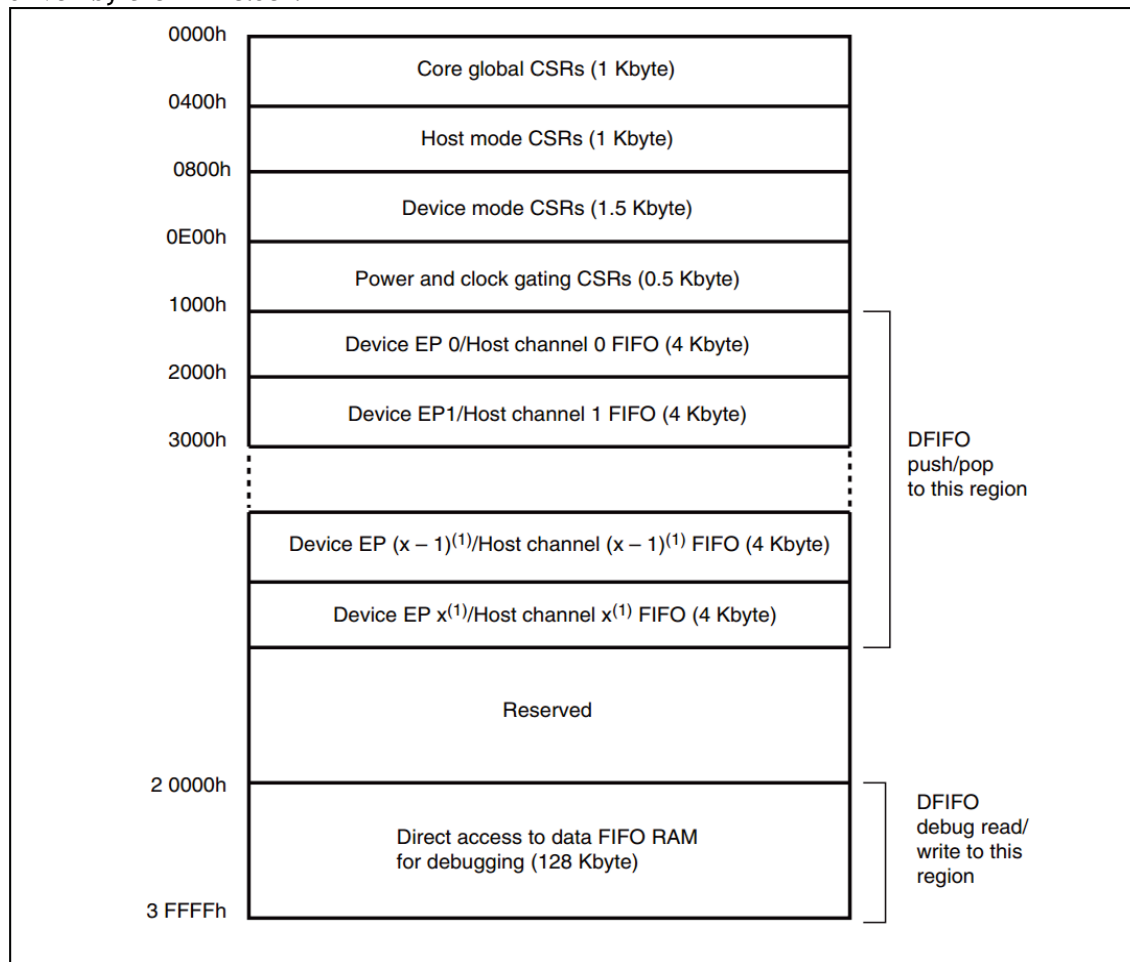


Figure 281 CSR memory image

In device mode x is 3, in host mode x is 7

### Global CSR Address Image

These registers are valid in both host mode and device mode.

Table 154 Controller Global Control and Status Registers (CSRs)

Register Designator	offset address	register name
OTG_FS_OTGCTL	0x000	OTG_FS control and status registers
OTG_FS_GOTGINT	0x004	OTG_FS interrupt registers
OTG_FS_GAHBCFG	0x008	OTG_FSAHB Configuration Registers
OTG_FS_GUSBCFG	0x00C	OTG_FSUSB Configuration Registers
OTG_FS_GRSTCTL	0x010	OTG_FS Reset Register
OTG_FS_GINTSTS	0x014	OTG_FS controller interrupt registers
OTG_FS_GINTMSK	0x018	OTG_FS Interrupt Mask Register

OTG_FS_GRXSTSR	0x01C	OTG_FS receive status debug read/OTG status read and pop registers
OTG_FS_GRXSTSP	0x020	
OTG_FS_GRXFSIZ	0x024	OTG_FS Receive FIFO Length Register
otg_fs_gnptxfsize	0x028	OTG_FS Non-Cyclic Transmit FIFO Length Register
OTG_FS_GNPTXSTS	0x02C	OTG_FS Off-cycle Transmit FIFO/Queue Status Register
OTG_FS_GCCFG	0x038	OTG_FS General Control Configuration Registers
OTG_FS_CID	0x03C	OTG_FS controller ID registers
otg_fs_hptxfsize	0x100	OTG_FS Host Mode Periodic Transmit FIFO Length Register
OTG_FS_DIEPTXFx	0x104 0x108 0x10C	OTG_FS device mode IN endpoint TXFIFO length register (x=1..4, specifies the number of the FIFO)

1. The general rule is to use OTG\_FS\_HNPTXFSIZ for host mode and OTG\_FS\_DIEPTXF0 for device mode.

### Host Mode CSR Address Image

These registers need to be configured each time you switch to host mode.

Table155 Control and Status Registers (CSRs) in Host Mode

Register Designator	offset address	register name
OTG_FS_HCFG	0x400	OTG_FS host mode configuration registers
OTG_FS_HFIR	0x404	OTG_FS Host Mode Frame Interval Register
OTG_FS_HFNUM	0x408	OTG_FS host mode frame number/remaining frame time registers
OTG_FS_HPTXSTS	0x410	OTG_FS host mode periodic TXFIFO/queue status registers
OTG_FS_HAINT	0x414	OTG_FS host mode all channel interrupt registers
OTG_FS_HAINTMSK	0x418	OTG_FS host mode all channel interrupt mask registers
OTG_FS_HPRT	0x440	OTG_FS host mode port control and status registers
OTG_FS_HCCHARx	0x500 0x520 ... 0x5E0	OTG_FS Host Mode Channel x Characteristics Register (x=0..7, x indicates channel number)
OTG_FS_HCINTx	0x508	OTG_FS host mode channel x interrupt register (x=0..7, x indicates channel number)
OTG_FS_HCINTMSKx	0x50C	OTG_FS host mode channel x interrupt mask register (x=0..7, x indicates channel number)
OTG_FS_HCTSIZx	0x510	OTG_FS host mode channel x transmission length register (x=0..7, x indicates channel number)

### Device Mode CSR Address Image

These registers must be configured each time you switch to device mode.

Table156 Device Mode Control and Status Registers

Register Designator	offset address	register name
OTG_FS_DCFG	0x800	OTG_FS Device Mode Configuration Registers
OTG_FS_DCTL	0x804	OTG_FS device mode control registers
OTG_FS_DSTS	0x808	OTG_FS Device Mode Status Register
OTG_FS_DIEPMSK	0x810	OTG_FS device mode IN endpoint common interrupt mask registers
OTG_FS_DOEPMSK	0x814	OTG_FS Device Mode OUT Endpoints Common Interrupt Mask Registers
OTG_FS_DAIN	0x818	OTG_FS device mode all endpoint interrupt registers
OTG_FS_DAINMSK	0x81C	OTG_FS device mode all endpoint interrupt mask registers
OTG_FS_DVBUSDIS	0x828	OTG_FS Device Mode VBUS Power Down Time Register
OTG_FS_DVBUSPULSE	0x82C	OTG_FS Device Mode VBUS Pulse Time Register
otg_fs_diepempmsk	0x834	OTG_FS Device Mode IN Endpoint FIFO Air Break Mask Register
OTG_FS_DIEPCTL0	0x900	OTG_FS device mode IN control endpoint 0 control register

OTG_FS_DIEPCTLx	0x920 0x940 0x960	OTG_FS device mode IN endpoint x control register (x=1..3, x indicates endpoint number)
OTG_FS_DIEPINTx	0x908	OTG_FS device mode IN endpoint x interrupt register (x=1..3, x indicates endpoint number)
OTG_FS_DIEPTSIZ0	0x910	OTG_FS Device Mode IN Endpoint 0 Transmit Length Register
OTG_FS_DTXFSTSx	0x918	OTG_FS device mode IN endpoint TXFIFO status register (x=1..3, x indicates endpoint number)
OTG_FS_DIEPTSIZx	0x930 0x950 0x970	OTG_FS device mode IN endpoint x transmission length register (x=1..3, x indicates endpoint number)
otg_fs_oeepctl0	0xB00	OTG_FS device mode OUT control endpoint 0 control register
OTG_FS_DOEPTCTLx	0xB20 0xB40 0xB60	OTG_FS device mode OUT endpoint x control register (x=1..3, x indicates endpoint number)
OTG_FS_DOEPTINTx	0xB08	OTG_FS device mode OUT endpoint x interrupt register (x=1..3, x indicates endpoint number)
OTG_FS_DOEPTSIZx	0xB10	OTG_FS device mode OUT endpoint x transmission length register (x=1..3, x indicates endpoint number)
OTG_FS_DOEPTSIZx	0xB30 0xB50 0xB70	OTG_FS device mode OUT endpoint x transfer size register (x=1..3, x indicates endpoint number)

### Data FIFO (DFIFO) Access Register Address Mapping

This set of register columns is valid in both host and device modes and is used to read or write the FIFO of a special endpoint or channel in the specified direction. If a channel in host mode is of type IN, the corresponding FIFO can only be read. Similarly, if a channel in host mode is of type OUT, the corresponding FIFO can only perform write operations.

Table157 Data FIFO (DFIFO) Access Registers

Data FIFO (DFIFO) access to register segments	address range	access method
IN endpoint 0 in device mode / OUT channel 0 in host mode: DFIFO write only	0x1000-0x1FFC	write operation
OUT endpoint 0 in device mode/IN channel 0 in host mode: DFIFO read-only		read operation
IN endpoint 1 in device mode/OUT channel 1 in host mode: DFIFO write-only	0x2000-0x2FFC	write operation
OUT endpoint 1 in device mode/IN channel 1 in host mode: DFIFO read-only		read operation
.....	.....	.....
IN endpoint x in device mode/OUT channel x in host mode: DFIFO write-only	0xX000-0xXFFC	write operation
OUT endpoint x in device mode/IN channel x in host mode: DFIFO read-only		read operation

The x in the table is 3 in device mode and 7 in host mode

### Power Supply and Clock Control CSR Register Image

Only one register is used to control power supply and clock control, and this register is valid in both device mode and host mode.

Table158 Power Supply and Gating Control and Status Registers

register name	abridge	offset address
Power supply and gated clock control registers	PCGCR	0xE00-0xE04
Reserved		0xE05-0xFFFF

## 29.16.2 OTG\_FS Global Registers

These registers are valid in both device and host modes and do not need to be re-initialized when the mode is switched. The value of each bit in the registers is represented by a binary number if not otherwise noted.

### OTG\_FS control and status register (OTG\_FS\_GOTGCTL)

Offset address: 0x000

Reset value: 0x0001 0000

The OTG control and status registers control the operation of the OTG\_FS controller and react to the status of the controller.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												BSVLD	ASVLD	DBCT	CIDSTS	Reserved					DHNPEN	HSHPEN	HNPRQ	HNGSCS	Reserved					SRQ	SRQSCS
												r	r	r	r						rw	rw	rw	r	res					rw	r

Bit	notation	clarification
31:20	Reserved	Reserved bit, hardware forced to 0
19	BSVLD	<b>BSVLD:</b> Class B session valid (B-session valid) indicates the status of the device mode transceiver 0: Class B session is invalid; 1: Class B sessions are valid. In OTG mode, the user can use this bit to determine whether the device is connected to the host or not Note: Valid only in device mode.
18	ASVLD	<b>ASVLD:</b> Class A session valid (A-session valid) indicates the status of the host mode transceiver 0: Class A session is invalid; 1: Class A sessions are valid. Note: Valid only in host mode.
17	DBCT	<b>DBCT:</b> Long/short debounce time indicates the debounce time of the detected connection. 0: Long reflection time for physical connections (100ms + 2.5us); 1: Short reflection time for software connection (2.5us). Note: Valid only in host mode.
16	CIDSTS	<b>CIDSTS:</b> Connector ID status indicates the status of the ID line on the connector. 0: OTG_FS is in Class A device mode; 1: OTG_FS is in Class B device mode. Note: Valid in both device mode and host mode.
15:12	Reserved	Reserved
11	DHNPEN	<b>DHNPEN:</b> Device HNP enabled The application sets this bit when it successfully receives a SetFeature.SetHNPEnable command from the USB host. 0: Disable device mode HNP; 1: Enable device mode HNP. Note: Valid only in device mode.
10	HSHPEN	<b>HSHPEN:</b> Host mode HNP enable (Host set HNP enable) The application sets this bit when it successfully enables HNP for the connected device (via the SetFeature.SetHNPEnable command). 0: Disable host mode HNP; 1: Enable host mode HNP. Note: Valid only in host mode.
9	HNPRQ	<b>HNPRQ:</b> HNP request (HNP request) The application program sets this bit when it needs to send an HNP request to the USB host. The application program can clear this bit by writing '0' when the Host Negotiation Successful Status Transition bit of the OTG Interrupt Register (HNSSCHG bit of the OTG_FS_GOTGINT register) is set. The controller clears this bit when the HNSSCHG bit is cleared. 0: No HNP request; 1: HNP request. Note: Valid only in device mode.
8	HNGSCS	<b>HNGSCS:</b> Host negotiation success The controller sets this bit when host negotiation is successful. The controller clears this bit when the application sets the HNP request bit (HNPRQ). 0: Host negotiation failed; 1: Host negotiation was successful. Note: Valid only in device mode.
7:2	Reserved	Reserved
1	SRQ	<b>SRQ:</b> Session request The application program initiates a session request on the USB bus by setting this bit. The application program can clear this bit by writing '0' when the Host Negotiation Successful Status Change bit of the OTG Interrupt Register (HNSSCHG bit of the OTG_FS_GOTGINT register) is set. The controller clears this bit when the HNSSCHG bit is cleared. If a user uses the USB 1.1 Full Speed Serial Transceiver interface to initiate a session request, the application must wait for the VBUS line to drop to 0.2 V after the Class B Session Valid bit (the BSVLD bit in the OTG_FS_GOTGCTL

		register) is cleared. Different PHYs use different wait times, controlled by the vendor of the PHY. 0: no session request; 1: session request. Note: Valid only in device mode.
0	SRQSCS	<b>SRQSCS:</b> Session request success (Session request success) The controller sets this bit after a successful session request. 0: Session request failed; 1: The session request was successful. Note: Valid only in device mode.

#### OTG\_FS interrupt register (OTG\_FS\_GOTGINT)

Offset address: 0x04

Reset value: 0x0000 0000

The application program reads this register when an OTG interrupt occurs and clears the OTG interrupt flag by clearing the corresponding bit.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												DBCNE	ADTOCHG	HNGDET	Reserved								HNSSCHG	SRSSCHG	Reserved				SEDET	Reserved	
												rc_w1	rc_w1	rc_w1									rc_w1	rc_w1					rc_w1		

Bit	notation	clarification
31:20	Reserved	Reserved bit, hardware forced to 0
19	DBCNE	<b>DBCNE:</b> Debounce done The controller sets this bit at the end of the device connection line reflection. The application program can initiate a USB reset signal after this interrupt is detected. This bit is only valid when the HNP Valid or SRP Valid bit of the controller's USB Configuration Register (HNPCAP bit or SRPCAP bit of the OTG_FS_GUSBCFG register) is set. Note: Valid only in host mode.
18	ADTOCHG	<b>ADTOCHG:</b> Class A device timeout (A-device timeout change) The controller sets this bit when a Class A device waits for a Class B device insertion timeout. Note: Valid in both host mode and device mode.
17	HNGDET	<b>HNGDET:</b> Host negotiation detected The controller sets this bit when it detects a host negotiation request on the USB bus. Note: Valid in both host mode and device mode.
16:10	Reserved	Reserved
9	HNSSCHG	<b>HNSSCHG:</b> Host negotiation success status change The controller sets this bit when the USB host negotiation request succeeds or fails. The application program needs to read the Host Negotiation Success bit of the OTG Control and Status Register (HNGSCS bit of the OTG_FS_GOTGCTL register) to get the success or failure information. Note: Valid in both host mode and device mode.
8	SRSSCHG	<b>SRSSCHG:</b> Session request success status change The controller sets this bit on session request success or failure. The application program needs to get the success or failure information by reading the session request success bit of the OTG control and status register (SRQSCS bit of the OTG_FS_GOTGCTL register). Note: Valid in both host mode and device mode.
7:3	Reserved	Reserved
2	SEDET	<b>SEDET:</b> session end detected The controller sets this bit when it detects VBUS < 0.8V to indicate that the level of the VBUS line of the Class B device is invalid.
1:0	Reserved	Reserved

### OTG\_FSAHB configuration register (OTG\_FS\_GAHBCFG)

Offset address: 0x008

Reset value: 0x0000 0000

This register is used to configure the controller when powering up or changing the controller mode. This register mainly configures some parameters related to AHB. Do not modify this register after the initial configuration is complete. The application program must configure this register before starting to transfer data to the AHB or USB.

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

Bit	notation	instructions
31:9	Reserved	Reserved bit, hardware forced to 0
8	PTXFELVL	<b>PTXFELVL</b> : Periodic TxFIFO empty level Indicates the circumstances under which a periodic transmit FIFO air interrupt of the controller interrupt register needs to be triggered (PTXFE bit of the OTG_FS_GINTSTS register): 0: PTXFE (in the OTG_FS_GINTSTS register) interrupt indicates that the periodic transmit FIFO is half empty; 1: PTXFE (in the OTG_FS_GINTSTS register) interrupt indicates that the periodic transmit FIFO is fully empty. Note: Valid only in host mode.
7	TXFELVL	<b>TXFELVL</b> : Transmit FIFO empty level (TxFIFO empty level) In device mode, this bit indicates the conditions under which the IN endpoint needs to be triggered to send a FIFO air break (TXFE bit of the OTG_FS_DIEPINTx register): 0: TXFE (in the OTG_FS_DIEPINTx register) interrupt indicates that the IN endpoint TXFIFO is half empty; 1: TXFE (in the OTG_FS_DIEPINTx register) interrupt indicates that the IN endpoint TXFIFO is fully empty. In host mode, this bit indicates the circumstances under which a non-periodic transmit FIFO air break (NPTXFE bit in the OTG_FS_GINTSTS register) needs to be triggered: 0: NPTXFE (in the OTG_FS_GINTSTS register) interrupt indicates that the non-periodic transmit FIFO is half empty; 1: The NPTXFE (in the OTG_FS_GINTSTS register) interrupt indicates that the non-periodic transmit FIFO is fully empty.
6:1	Reserved	Reserved
0	GINTMSK	<b>GINTMSK</b> : Global interrupt mask The application program can choose to mask or unmask interrupts with this bit. However, the controller still updates the interrupt status register whether this bit is set or not. 0: Mask interrupt; 1: Do not mask interrupts. Note: Valid in both device mode and host mode.

### OTG\_FS\_USB configuration register ( OTG\_FS\_GUSBCFG )

Offset address: 0x00C

Reset value: 0x0000 0A00

This register is used to configure the controller during power-up or mode switching. This register is mostly used to configure parameters related to USB and USBPHY. The application program must configure this register before transferring data to AHB or USB. Do not modify this register after the initial configuration is complete.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTXPKT	FDWOK	FHMOD	Reserved															TRDT				HNPCAP	SRPCAP	Reserved	PHYSEL	Reserved			TOCAL		
rw	rw	rw																rw	rw	rw	rw	rw	rw		r				rw		

Bit	notation	clarification
31	CTXPKT	<b>CTXPKT:</b> Erroneous Send Packet (Corrupt Txp acket) This bit is for debugging only. You cannot set this bit to '1'. Note: Valid in both device mode and host mode.
30	FDMOD	<b>FDMOD:</b> Forced device mode (FDMOD) Setting this bit to '1' will force the controller into device mode regardless of the state of the OTG_FS_ID input pin. 0: Normal mode; 1: Forced device mode. After setting this bit, the application needs to wait at least 25ms for the setting to take effect. Note: Valid in both host mode and device mode.
29	FHMOD	<b>FHMOD:</b> Forcehostmode Setting this bit to '1' will force the controller into host mode regardless of the state of the OTG_FS_ID input pin. 0: Normal mode 1: Forced host mode After setting this bit, the application needs to wait at least 25ms for the setting to take effect. Note: Valid in both host mode and device mode.
28:14	Reserved	Reserved
13:10	TRDT	<b>TRDT:</b> USB bus turnaround time (USB turnaround time) Set the turnaround time in terms of the clock period of the PHY layer. Sets the response time for a MAC request to the Packet FIFO Controller (PFC) to request data from the DFIFO (SPRAM). These bits must be set: 0101: When the MAC interface is 16-bit UTMIFS; 1001: When the MAC interface is 8-bit UTMIFS. Note: Valid only in device mode.
9	HNPCAP	<b>HNPCAP:</b> HNP enable (HNP-capable) The application program uses this bit to enable the HNP function of the OTG_FS controller. 0: Disable the HNP function; 1: Enable the HNP function. Note: Valid in both device mode and host mode.
8	SRPCAP	<b>SRPCAP:</b> SRP enable (SRP-capable) The application program uses this bit to enable the SRP function of the OTG_FS controller. If the controller is operating in Class B device mode without SRP functionality, it cannot ask the connected Class A device (host) to enable the VBUS line and initiate a session. 0: Disable the SRP function; 1: Enable the SRP function. Note: Valid in both device mode and host mode.
7:3	Reserved	Reserved
2:0	TOTAL	<b>TOTAL:</b> FS timeout calibration Taking into account the additional latency introduced by the PHY, the application program can adjust the controller's full-speed packet timeout judgment latency; these bits give the length of the adjustment latency in terms of the number of PHY clocks. The additional delay introduced by different PHYs in controlling the bus state is different, and different bus delays can be accommodated with these bits. The USB standard specifies a timeout judgment of 16 to 18 BIT time for full speed packets. The application program needs to configure this bit according to the enumeration speed. The BIT time added per PHY clock is 0.25 BIT time.

Table159 TRDT values

AHB Frequency Range (MHz)		TRDT Min.
minimal	greatest	
14.2	15	0xF
15	16	0xE
16	17.2	0xD
17.2	18.5	0xC
18.5	20	0xB
20	21.8	0xA
21.8	24	0x9
24	27.5	0x8
27.5	32	0x7
32	-	0x6



## OTG\_FS reset register (OTG\_FS\_GRSTCTL)

Offset address: 0x10

Reset value: 0x2000 0000

The application program can use this register to reset each hardware module of the controller.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AHBIDL	Reserved										TXFNUM					TXFFLSH		RXFFLSH		Reserved		FCRST		HSRST		CSRST					
r											rw					rs		rs		res		rs		rs		rs					

Bit	notation	clarification
31	AHBIDL	AHBIDL: AHB master idle (AHB master idle) Indicates whether the AHB master module is in idle state. Note: Valid in both device mode and host mode.
30:11	Reserved	Reserved
10:6	TXFNUM	TXFNUM: Transmit FIFO number (TxFIFO number) This bit indicates which FIFO needs to be refreshed by sending the FIFO refresh bit. The application program can modify this bit only if the controller clears the Send FIFO Refresh bit. l00000: Host mode indicates non-periodic sending of FIFOs Device mode indicates sending of FIFOs0 l00001: Host mode indicates periodic sending of FIFOs Device mode indicates sending of FIFOs1 l00010: Device mode indicates sending FIFO2 ..... l00101: Device mode indicates sending FIFO5 l10,000: Flush all transmit FIFOs in host mode or device mode Note: Valid in both host mode and device mode.
5	TXFFLSH	TXFFLSH: Send FIFO flush (TxFIFO flush) This bit is used to refresh individual or all transmit FIFOs and cannot be set when the controller is transmitting. The application program can set this bit only if it detects that the controller is neither writing data to nor reading data from the sending FIFO. This can be detected in the following ways: Read: NAK active interrupt to ensure that the controller is not reading data from the FIFO. Write: the AHBIDL bit of the OTG_FS_GRSTCTL register ensures that the controller is not writing the FIFO. Note: Valid in both device mode and host mode.
4	RXFFLSH	RXFFLSH: Receive FIFO flush (RxFIFO flush) The application can use this bit to refresh the entire receive FIFO, but it is necessary to ensure that the controller is not transmitting data. The application program can set this bit only if it detects that the controller is neither writing nor reading the receive FIFO. The application program can do nothing else until it waits for this bit to be cleared by the controller again. This bit takes 8 clock cycles (whichever is the slower of the PHY or AHB clocks) to clear. Note: Valid in both host mode and device mode.
3	Reserved	Reserved
2	FCRST	FCRST: Host frame counter reset The application program resets the controller's internal frame counter by writing this bit. After the frame counter is reset, the frame number of the first SOF sent by the controller is 0. Note: Valid only in host mode.
1	HSRST	HSRST: HCLK software reset (HCLK soft reset) Applications can use this bit to refresh the control logic of the AHB clock domain. Only modules driven by the AHB clock domain are reset. the FIFO is not affected by this bit. According to the protocol, all state machines driven by the AHB clock domain are reset to the idle state after completing the current transmission of AHB. Clear the CSR control bit in the state machines driven by the AHB clock domain. To clear this interrupt, the status mask bits used to control the interrupt status and generated by the state machine driven by the AHB clock domain are



		cleared. Since the interrupt status bit is not cleared, the application program can still obtain the status of any controller events generated after this bit is set. The controller will automatically clear this bit after all necessary logic module resets. This operation will be maintained for a number of clock cycles, as determined by the state the controller is currently in. Note: Valid in both device mode and host mode.
0	CSRST	<b>CSRST:</b> Controller software reset (Core soft reset) resets the HCLK and PCLK driver domains: Clears all interrupt and CSR registers except the following bits: the RSTPDMODL bit of the OTG_FS_PCGCCTL register the GAYEHCLK bit of the OTG_FS_PCGCCTL register the PWRCLMP bit of the OTG_FS_PCGCCTL register STPPCLK bit of the OTG_FS_PCGCCTL register FSLSPCS bit of the OTG_FS_HCFG register DSPD bit of the OTG_FS_DCFG register The state machines of all modules (except the AHB slave unit) are reset to an idle state and all transmit FIFOs and receive FIFOs are flushed. All data transfers on the AHB master module end immediately upon completion of the last data phase of the AHB transfer. All transfers on USB end immediately. The application program can set this bit any time a reset of the controller is required. The controller will automatically clear this bit after all required logic modules have been reset, and this operation will last for a number of clock cycles, depending on the current state of the controller. Once this bit is cleared, software must wait at least 3 PHY clock cycles (the delay of the synchronization operation) before activating the PHY domain. The application program must ensure that bit 31 of this register is '1' (i.e., the AHB master module is idle) before starting any other operation. Normally, it is only necessary to enter a software reset operation during the software development phase and when dynamically modifying the PHY select bits in the USB configuration registers listed above. If the user changes the PHY configuration, the corresponding clock will be selected and applied to the PHY domain. Once the new clock is selected, the PHY domain needs to be reset for further operation. Note: Valid in both device mode and host mode.

### OTG\_FS controller interrupt register (OTG\_FS\_GINTSTS)

Offset address: 0x014

Reset value: 0x0400 0020

This register interrupts the application when a system event occurs that matches the current mode (device mode or host mode).

Some bits of the register are only valid in host mode and others are only valid in device mode.

The register can also be used to indicate the current mode. The application program needs to clear the interrupt status bit of type rc\_w1 by writing a 1 to the corresponding bit.

The FIFO status interrupt bit is read-only and is automatically cleared once the application program reads or writes to the FIFO in the interrupt service program.

During initialization, the application needs to clear the corresponding bit of the OTG\_FS\_GINTSTS register before enabling an interrupt bit to avoid unnecessary interrupts due to the original value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WKUJINT	SRQJINT	DISCINT	CIDSCHG	Reserved	PTXFE	HCINT	HPRTINT	Reserved		IPXFR/INCOMPI SOUT	IISOXFR	OEPIINT	IEPIINT	Reserved	EOPF	ISOODRP	ENUMDNE	USBRST	USBSUSP	ESUSP	Reserved		BOUTNAKEFF	GINAKEFF	NPTXFE	RXFLVL	SOF	OTGINT	MMIS	CMOD	
rc_w1	rc_w1	rc_w1	rc_w1		r	r	r		rc_w1	rc_w1		r	r		rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1		r	r	r	r		rc_w1		rc_w1		r

Bit	notation	clarification
31	WKUPINT	<b>WKUPINT:</b> Wakeup/remote wakeup detected interrupt (Resume/remote wakeup detected interrupt) In device mode, this interrupt is generated when a wakeup signal is detected on the USB bus; In host mode, this interrupt is generated when a remote wake-up signal is detected on the USB bus. Note: Valid in both device mode and host mode.
30	SRQINT	<b>SRQINT:</b> Session request/new session detected interrupt (Session request/new session detected interrupt) In host mode, this interrupt is generated when a session request is detected from the device; In device mode, this interrupt is generated when VBUS is within the valid level range for a Class B device.

		Note: Valid in both device mode and host mode.
29	DISCINT	<b>DISCINT:</b> Disconnect detected interrupt This interrupt is generated when the device is detected to be disconnected. Note: Valid only in host mode.
28	CIDSCHG	<b>CIDSCHG:</b> Connector ID status change The controller sets this bit when it detects a change in the status of the ID line on the connection. Note: Valid in both host mode and device mode
27	Reserved	Reserved
26	PTXFE	<b>PTXFE:</b> Periodic TxFIFO empty (Periodic TxFIFO empty) This interrupt is generated when the periodic send FIFO is half-empty or fully empty and at least one request can be written in the periodic request queue. The half-empty or full-empty state is determined by the Periodic Send FIFO Empty Level bit in the controller's AHB Configuration Register (PTXFELVL bit in the OTG_FS_GAHBCFG register). Note: Valid only in host mode.
25	HCINT	<b>HCINT:</b> Host channels interrupt The controller sets this bit to indicate that there is an unhandled host channel event (in host mode). The application needs to read the host all channel interrupt register (OTG_FS_HAINT register) to get the channel number of the channel that generated the interrupt, and then read the corresponding host channel x interrupt register (OTG_FS_HCINTx register) to get the specific information about the interrupt. The application program needs to clear this bit by clearing the corresponding bit in the OTG_FS_HCINTx register. Note: Valid only in host mode.
24	HPRTINT	<b>HPRTINT:</b> Host port interrupt The controller sets this bit to indicate that the status of one of the OTG_FS controller's ports has changed. The application program needs to read the Host Port Control and Status Register (OTG_FS_HPRT) to obtain the specific information that caused this interrupt. The application program must clear this bit by clearing the corresponding bit in the Host Port Control and Status Register. Note: Valid only in host mode.
23:22	Reserved	Reserved
21	IPXFR	<b>IPXFR:</b> Incomplete periodic transfer (IPXFR). In host mode, the controller generates this interrupt at the end of the frame when there is still an unfinished periodic transmission belonging to the current frame. Note: Valid only in host mode. <b>INCOMPIROUT:</b> Incomplete isochronous OUT transfer In device mode, the controller generates this interrupt at the end of a frame when there is still at least one unfinished synchronized OUT transmission belonging to the current frame. This interrupt is generated along with the periodic end-of-frame interrupt (EOPF) in this register. Note: Valid only in device mode.
20	IISOXFR	<b>IISOXFR:</b> Incomplete isochronous IN transfer In device mode, the controller generates this interrupt at the end of a frame when there is still at least one unfinished synchronization IN transmission belonging to the current frame. This interrupt is generated along with the periodic end-of-frame interrupt (EOPF) in this register. Note: Valid only in device mode.
19	OEPINT	<b>OEPINT:</b> OUT end point interrupt In device mode, the controller sets this bit to indicate that there is an OUT endpoint event that has not been processed by the controller. The application program needs to read the device all endpoint interrupt register (OTG_FS_DAINTE) to get the endpoint number that generated the interrupt event, and then read the corresponding device OUT endpoint x interrupt register (OTG_FS_DOEPINTx) to get the specific information that generated the interrupt. The application program needs to clear this bit by clearing the corresponding bit in the OTG_FS_DOEPINTx register. Note: Valid only in device mode.
18	IEPINT	<b>IEPINT:</b> IN end point interrupt (IN end point interrupt) In device mode, the controller sets this bit to indicate that there is an IN endpoint event that has not been processed by the controller. The application program needs to read the Device All Endpoint Interrupt Register (OTG_FS_DAINTE) to get the endpoint number that generated the interrupt event, and then read the corresponding Device IN Endpoint x Interrupt Register (OTG_FS_DIEPINTx) to get the specific information that generated the interrupt. The application program needs to clear this bit by clearing the corresponding bit in the OTG_FS_DIEPINTx register. Note: Valid only in device mode.
17:16	Reserved	Reserved
15	EOPF	<b>EOPF:</b> End of periodic frame interrupt (End of periodic frame interrupt) This interrupt indicates that the periodic frame interval time set in the Device Configuration Register (PFIVL bit of the OTG_FS_DCFG register) has arrived in

		the current frame. Note: Valid only in device mode.
14	ISOODRP	<b>ISOODRP</b> : Isochronous OUT packet dropped interrupt When the receive FIFO does not have enough space to hold a maximum length packet for a synchronization OUT endpoint, it will cause the controller to fail to write the synchronization OUT packet and generate this interrupt. Note: Valid only in device mode.
13	ENUMDNE	<b>ENUMDNE</b> : Enumeration done The controller sets this bit to indicate that speed enumeration information has been obtained. The application program reads the Device Status Register (OTG_FS_DSTS) to obtain the speed information for the enumeration. Note: Valid only in device mode.
12	USBRST	<b>USBRST</b> : USB reset The controller sets this bit when it detects a USB reset signal. Note: Valid only in device mode.
11	USBSUSP	<b>USBSUSP</b> : USB suspend The controller sets this bit when it detects a hang signal on the USB line. The controller will enter the hang state if there is no activity on the data line for more than 3ms. Note: Valid only in device mode.
10	ESUSP	<b>ESUSP</b> : Early suspend The controller sets this bit when it detects that the USB bus is idle for 3ms. Note: Valid only in device mode.
9:8	Reserved	Reserved
7	GONAKEFF	<b>GONAKEFF</b> : Global OUT NAK effective Indicates to the application program that the global OUTNAK bit of the Device Control Register (SGONAK bit of the OTG_FS_DCTL register) is in effect. The application clears this bit by writing the clear global OUTNAK bit of the Device Control Register (CGONAK bit of the OTG_FS_DCTL register). Note: Valid only in device mode.
6	GINAKEFF	<b>GINAKEFF</b> : Global IN non-periodic NAK effective Indicates to the application program that the Set Global Non-Cyclic INNAK bit of the Device Control Register (SGINAK bit of the OTG_FS_DCTL register) has taken effect. That is, the controller has corresponded to the application program's operation of setting the global INNAK bit. The application program clears this bit by clearing the Clear Global Non-Cyclic INNAK bit (CGINAK bit of the OTG_FS_DCTL register) of the Device Control Register. This interrupt does not imply that a NAK handshake signal was sent to the USB bus; the STALL bit has a higher priority than the NAK bit. Note: Valid only in device mode.
5	NPTXFE	<b>NPTXFE</b> : Non-periodic transmit FIFO empty (Non-periodic Tx FIFO empty) The controller generates this interrupt when the nonperiodic transmit FIFO is either fully empty or half empty and at least one request can be written in the nonperiodic request queue. The state of the FIFO being half empty or fully empty depends on the nonperiodic transmit FIFO empty level bit of the controller's AHB Configuration Register (the TXFELVL bit of the OTG_FS_GAHBCFG register). Note: Valid only in host mode.
4	RXFLVL	<b>RXFLVL</b> : Receive FIFO non-empty (Rx FIFO non-empty) indicates that there is at least one packet in the receive FIFO waiting to be processed. Note: Valid in both device mode and host mode.
3	SOF	<b>SOF</b> : Star to frame In host mode, the controller sets this bit to indicate that a SOF (full speed device) or hold active (low speed device) signal has been sent to the USB bus. The application program must write a '1' to this bit to clear this interrupt. In device mode, the controller sets this bit to indicate that a SOF has been detected on the USB bus. the application needs to read the Device Status Register to get the frame number. This interrupt is generated only when the controller is at full speed. Note: Valid in both device mode and host mode.
2	OTGINT	<b>OTGINT</b> : OTG interrupt The controller sets this bit to indicate that an OTG protocol event has occurred. The application program must read the OTG interrupt register (OTG_FS_GOTGINT) to obtain the specific information that generated this interrupt. The application program must clear this bit by clearing the corresponding bit in the OTG_FS_GOTGINT register. Note: Valid in both device mode and host mode.
1	MMIS	<b>MMIS</b> : Mode mismatch interrupt (MMIS) The controller sets this bit when the application tries to perform the following actions: Attempts to access registers in host mode when the controller is running in device mode. Attempts to access registers in device mode when the controller is running in host mode. When operating the register, the correct response is received at the AHB side, but such operations are internally ignored by the controller and do not have any effect on the operation of the controller.

		Note: Valid in both host mode and device mode.
0	CMOD	<b>CMOD:</b> Current mode of operation (Current mode of operation) indicates the current mode. 0: Device mode; 1: Host mode. Note: Valid in both device mode and host mode.

### OTG\_FS Interrupt Mask Register (OTG\_FS\_GINTMSK)

Offset address: 0x018

Reset value: 0x0000 0000

This register is used in conjunction with the controller interrupt register to generate interrupts. When an interrupt bit is masked, no corresponding interrupt is generated, however the corresponding bit in the controller interrupt register (OTG\_FS\_GINTSTS) is still set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WUIM	SRQIM	DISCINT	CIDSCHGM	Reserved	PTXFEM	HCIM	PPTIM	Reserved	IPXFRM/IISOXFRM	IISOXFRM	OEPINT	IEPINT	Reserved	EOPFM	ISOODRPM	ENUNDNEM	USBRST	USBSUSM	ESUSPM	Reserved	GONAKEFFM	GINAKEFFM	NPTXFEM	RXFLVLM	SOFM	OTGINT	MMISM	Reserved			
rw	rw	rw	rw		rw	rw	r		rw	rw	rw	rw		rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	notation	clarification
31	WUIM	<b>WUIM:</b> Wakeup/remote wakeup detected interrupt mask (Resume/remote wakeup detected interrupt mask) 0: Mask this interrupt; 1: Do not mask this interrupt. Note: Valid in both device mode and host mode.
30	SRQIM	<b>SRQIM:</b> Session request/new session detected interrupt mask 0: Mask this interrupt; 1: Do not mask this interrupt. Note: Valid in both device mode and host mode.
29	DISCINT	<b>DISCINT:</b> Detected disconnect event interrupt mask (Disconnect detected interrupt mask) 0: Mask this interrupt; 1: Do not mask this interrupt. Note: Valid only in host mode.
28	CIDSCHGM	<b>CIDSCHGM:</b> Connector ID status change mask on connection 0: Mask this interrupt; 1: Do not mask this interrupt. Note: Valid in both host mode and device mode
27	Reserved	Reserved
26	PTXFEM	<b>PTXFEM:</b> Periodic TxFIFO empty mask 0: Mask this interrupt; 1: Do not mask this interrupt. Note: Valid only in host mode
25	HCIM	<b>HCIM:</b> Host channels interrupt mask 0: Mask this interrupt; 1: Do not mask this interrupt. Note: Valid only in host mode.
24	PRTIM	<b>PRTIM:</b> Host port interrupt mask 0: Mask this interrupt; 1: Do not mask this interrupt. Note: Valid only in host mode.
23:22	Reserved	Reserved
21	IPXFRM	<b>IPXFRM:</b> Incomplete periodic transfer mask 0: Mask this interrupt; 1: Do not mask this interrupt. Note: Valid only in host mode. <b>IISOXFRM:</b> Incomplete isochronous OUT transfer transfer mask 0: Mask this interrupt; 1: Do not mask this interrupt. Note: Valid only in device mode.
20	IISOXFRM	<b>IISOXFRM:</b> Incomplete isochronous IN transfer mask 0: Mask this interrupt; 1: Do not mask this interrupt. Note: Valid only in device mode.
19	OEPINT	<b>OEPINT:</b> OUT end points interrupt mask 0: Mask this interrupt; 1: Do not mask this interrupt. Note: Valid only in device mode.

18	IEPINT	<b>IEPINT:</b> IN end points interrupt mask 0: Mask this interrupt; 1: Do not mask this interrupt. Note: Valid only in device mode.
17:16	Reserved	Reserved
15	EOPFM	<b>EOPFM:</b> End of periodic frame interrupt mask (End of periodic frame interrupt mask) 0: Mask this interrupt; 1: Do not mask this interrupt. Note: Valid only in device mode.
14	ISOODRPM	<b>ISOODRPM:</b> Isochronous OUT packet dropped interrupt mask 0: Mask this interrupt; 1: Do not mask this interrupt. Note: Valid only in device mode.
13	ENUMDNEM	<b>ENUMDNEM:</b> Enumeration done mask 0: Mask this interrupt; 1: Do not mask this interrupt. Note: Valid only in device mode.
12	USBRST	<b>USBRST:</b> USB reset mask 0: Mask this interrupt; 1: Do not mask this interrupt. Note: Valid only in device mode.
11	USBSUSPM	<b>USBSUSPM:</b> USB suspend mask 0: Mask this interrupt; 1: Do not mask this interrupt. Note: Valid only in device mode.
10	ESUSPM	<b>ESUSPM:</b> Early suspend mask 0: Mask this interrupt; 1: Do not mask this interrupt. Note: Valid only in device mode.
9:8	Reserved	Reserved
7	GONAKEFFM	<b>GONAKEFFM:</b> Global OUT NAK effective mask 0: Mask this interrupt; 1: Do not mask this interrupt. Note: Valid only in device mode.
6	GINAKEFFM	<b>GINAKEFFM:</b> Global non-periodic INNAK status interrupt mask (Global non-periodic IN NAK effective mask) 0: Mask this interrupt; 1: Do not mask this interrupt. Note: Valid only in device mode.
5	NPTXFEM	<b>NPTXFEM:</b> Non-periodic Tx FIFO empty mask 0: Mask this interrupt; 1: Do not mask this interrupt. Note: Valid only in host mode.
4	RXFLVLM	<b>RXFLVLM:</b> Receive FIFO non-empty mask 0: Mask this interrupt; 1: Do not mask this interrupt. Note: Valid in both device mode and host mode.
3	SOFM	<b>SOFM:</b> Start of frame mask. 0: Mask this interrupt; 1: Do not mask this interrupt. Note: Valid in both device mode and host mode.
2	OTGINT	<b>OTGINT:</b> OTG interrupt mask 0: Mask this interrupt; 1: Do not mask this interrupt. Note: Valid in both device mode and host mode.
1	MMISM	<b>MMISM:</b> Mode mismatch interrupt mask 0: Mask this interrupt; 1: Do not mask this interrupt. Note: Valid in both host mode and device mode.
0	Reserved	Reserved

## OTG\_FS Receive Status Debug Read/OTG Status Read and POP Registers

### (OTG\_FS\_GRXSTSR/OTG\_FS\_GRXSTSP)

Address offset for read operation: 0x01C

Address offset for POP operation: 0x020

Reset value: 0x0000 0000

A read operation to the Receive Status Debug Read Register will return the data at the top of the receive FIFO. A read operation to the Receive Status Debug Read Register and the POP Register will top out the top data item in the receive FIFO.

The interpretation of receive status data is not the same in host mode and device mode. If the receive FIFO is empty, the controller ignores read/POP operations on the receive status and returns 0x0000 0000. the application program can only POP out the receive status FIFO if the Receive FIFO Non-Empty bit of the Controller Interrupt Register (the RXFLVL bit of the OTG\_FS\_GINTSTS register) is '1'.

#### Host Mode:

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

Bit	notation	clarification
31:21	Reserved	Reserved
20:17	PKTSTS	<b>PKTSTS:</b> Packet status indicates the status of the received packet. 0010: IN packet received; 0011: IN transmission completed (interrupt triggered); 0101: Data flip-flop bit error (triggered interrupt); 0111: Channel abort (trigger interrupt); Other: Reserved.
16:15	DPID	<b>DPID:</b> Data PID (Data PID) Indicates the data PID of the received packet 00: DATA0; 10: DATA1; 01: DATA2; 11: MDATA.
14:4	BCNT	<b>BCNT:</b> Byte count (Byte count) indicates the number of bytes in the received packet.
3:0	CHNUM	<b>CHNUM:</b> Channel number indicates which channel the currently received packet belongs to.

#### Device Mode:

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

Bit	notation	clarification
31:25	Reserved	Reserved
24:21	FRMNUM	<b>FRMNUM:</b> Frame number This 4 bits are the last 4 bits of the frame number to which the packet received from the USB belongs, and are valid only for synchronized OUT transmission.
20:17	PKTSTS	<b>PKTSTS:</b> Packet status indicates the status of the received packet. 0001: Global OUTNAK (trigger interrupt); 0010: OUT packet received; 0011: OUT transmission completed (interrupt triggered); 0100: SETUP transmission completed (triggered interrupt); 0110: SETUP packet received; Other: Reserved.
16:15	DPID	<b>DPID:</b> Data PID (Data PID) indicates the PID of the received OUT packet 00: DATA0; 10: DATA1; 01: DATA2; 11: MDATA.
14:4	BCNT	<b>BCNT:</b> Byte count (Byte count) indicates the number of bytes in the received packet.
3:0	EPNUM	<b>EPNUM:</b> endpoint number Indicates the endpoint number to which the currently received packet belongs.

## OTG\_FS Receive FIFO Length Register (OTG\_FS\_GRXFSIZ)

Offset address: 0x024

Reset value: 0x0000 0200

The application program can define the length of RAM allocated to the receive FIFO.

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

Bit	notation	clarification
31:16	Reserved	Reserved
15:0	RXFD	<b>RXFD:</b> Receive FIFO depth (RxFIFO depth) The unit of this value is 32-bit word. Minimum value is 16 Maximum value is 256 The power-on Reset value is the maximum depth value of the receive FIFO.

## OTG\_FS Non-Cyclic TXFIFO Size Register

### (OTG\_FS\_HNPTXFSIZ)/Endpoint 0 transfer FIFO size (OTG\_FS\_DIEPTXF0)

Address offset: 0x028

Reset value: 0x0000 0200

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NPTXFD/TX0FD																NPTXFSF/TX0FSA															
																rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw															

### Host Mode

Bit	notation	clarification
31:16	NPTXFD	<b>NPTXFD:</b> Non-periodic TxFIFO depth (Non-periodic TxFIFO depth) The unit of this value is a 32-bit word. Minimum value is 16 Maximum value is 256
15:0	NPTXFSF	<b>NPTXFSF:</b> Non-periodic receive FIFO start address in RAM (Non-periodic transmit RAM start address) The value of these bits indicates the starting address of the non-cyclic receive FIFO in RAM

### device mode

Bit	notation	clarification
31:16	TX0FD	<b>TX0FD:</b> Endpoint 0 Transmit FIFO depth (Endpoint 0 TxFIFO depth) The unit of this value is a 32-bit word. Minimum value is 16 Maximum value is 256
15:0	TX0FSA	<b>TX0FSA:</b> Start address of Endpoint 0 Transfer FIFO in RAM (Endpoint 0 TxFIFO depth) The value of these bits indicates the starting address of the non-cyclic receive FIFO in RAM

## OTG\_FS Non-Cyclic TXFIFO/Request Queue Status Register (OTG\_FS\_HNPTXSTS)

Offset address: 0x02C

Reset value: 0x0008 0200

*Notes: This register is invalid in device mode.*

This register is a read-only register that stores information about the remaining space in the non-periodic transmit FIFO and the non-periodic transfer request queue.

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

Bit	notation	clarification
31	Reserved	Reserved
30:24	NPTXQTOP	<b>NPTXQTOP:</b> Top of the non-periodic transmit request queue (Top of the non-periodic transmit request queue) Non-periodic send requests being processed by the MAC module: Bits [30:27]: channel/endpoint number; bits [26:25]: 00: IN/OUT command; 01: 0-length transmission packet (device IN/host OUT); 11: channel abort command;



		Bit 24: End (last request for the selected channel/endpoint).
23:16	NPTQXSAV	NPTQXSAV: Non-periodic transmit request queue space available Indicates the remaining space in the non-periodic transmission request queue. In host mode, this queue holds both IN and OUT transmission requests, and in device mode, only IN transmission requests. 00: Non-periodic transmission request queue full; 01: d× 1 request space remaining; 02: Remaining d× 2 request space; B× n: remaining d× n request space (0 n d≤× 8); other: reserved.
15:0	NPTXFSAV	NPTXFSAV: Non-periodic Tx FIFO space available Indicates the remaining space of the non-periodic Tx FIFO, this value is 32 bits. 00: Non-periodic transmit FIFO full; 01: 1 word of space remaining for d× ; 02: 2 words of space remaining for d× ; O× n: space for the remaining d× n words (0 n d≤× 256); other: reserved.

### OTG\_FS Generalized Controller Configuration Register (OTG\_FS\_GCCFG)

Offset address: 0x038

Reset value: 0x0000 XXXX

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

Bit	notation	clarification
31:21	Reserved	Reserved
20	SOFOUTEN	<b>SOFOUTEN</b> : SOF output enable 0: No SOF pulse is output; 1: Output SOF pulse to pin.
19	VBUSBSEN	<b>VBUSBSEN</b> : Enable the VBUS sensing "B" device to monitor the valid level of B class. 0: VBUS does not monitor the Class B active level; 1: VBUS monitors the Class B active level.
18	VBUSASEN	<b>VBUSASEN</b> : Enable the VBUS sensing "A" device to monitor the valid level of class "A". 0: VBUS does not monitor the Class A active level; 1: VBUS monitors the Class A active level.
17	Reserved	Reserved
16	PWRDWN	<b>PWRDWN</b> : Power down is used to activate the transceiver during transmission and reception. 0: Enable power down; 1: Disable power-down (transceiver activation).
15:0	Reserved	Reserved

### OTG\_FS controller ID register (OTG\_FS\_CID)

Offset address: 0x03C

Reset value: 0x0000 1200

This register is read-only and holds the product ID.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
PRODUCT_ID																																
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
Bit	notation								clarification																							
31:0	PRODUCT_ID								PRODUCT_ID: Product ID (Product ID field) Applications can write this ID bit.																							



### OTG\_FS Host Periodic Transmit FIFO Length Register (OTG\_FS\_HPTXFSIZ)

Offset address: 0x100

Reset value: 0x0200 0400

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

Bit	notation	clarification
31:16	PTXFSIZ	PTXFSIZ: Host periodic send FIFO depth (Host periodic TxFIFO depth) This value is a 32-bit word with a minimum value of 16 and a maximum value of 512.
15:0	PTXSA	PTXSA: Host periodic TxFIFO start address The Reset value of this register is the sum of the maximum receive FIFO depth and the maximum non-periodic transmit FIFO depth.

### OTG\_FS device IN endpoint transmit FIFO length register (OTG\_FS\_DIEPTXFx)

(where x is the number of the FIFO, x=1...3)

Offset address: 0x104+(x-1)×0x04

Reset value: 0x0200 0200

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

Bit	notation	clarification
31:16	INEPTXFD	INEPTXFD: IN endpoint transmit FIFO depth (IN endpoint TxFIFO depth) This value is a 32-bit word with a minimum value of 16 and a maximum value of 512. The Reset value is the maximum possible depth of the IN endpoint sending FIFOs
15:0	INEPTXSA	INEPTXSA: IN endpoint transmit FIFO in RAM start address (IN endpoint FIFOx transmit RAM start address) This value is the starting address of the IN endpoint transmit FIFO in RAM.

## 29.16.3 Registers in Host Mode

If not otherwise specified, the values in the registers are expressed in binary form.

The registers in host mode are valid only in host mode and cannot be accessed in device mode; illegal access results in undefined results. The registers in host mode are as follows:

### OTG\_FS host mode configuration register (OTG\_FS\_HCFG)

Offset address: 0x400

Reset value: 0x0000 0000

This register configures the operation of the controller after power-up; do not modify this register after initialization.

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

Bit	notation	clarification
31:3	Reserved	Reserved
2	FSLSS	FSLSS: support for full-speed and low-speed devices (FS- and LS-only support) The application uses this bit to configure the speed at which the controller enumerates devices. The application program can use this bit to cause the controller to enumerate a device that supports high-speed communication in full-speed mode. Do not modify this value after initialization. 1: Only full speed/low speed devices are supported, even if the inserted device supports high speed communication (read only).
1:0	FSLSPCS	FSLSPCS: Full Speed/Low Speed PHY clock select (FS/LS PHY clock select) When the controller is in full speed host mode: 01: PHY clock running at 48MHz; other values: reserved. When the controller is in low-speed host mode: 00: Reserved; 01: PHY clock running at 48MHz;

		10: The PHY clock runs at 6MHz, according to the USB1.1 Full Speed Mode definition, when UTMIFSPHY Low Power Mode is checked, the 6MHz clock is used in Low Speed Mode if the PHY supports 6MHz clock. Users need to perform a software reset operation once the 6MHz clock is checked in low-speed mode; 11: Reserved.
--	--	--

### OTG\_FS Host Frame Interval Register (OTG\_FS\_HFIR)

Offset address: 0x404

Reset value: 0x0000 EA60

This register sets the frame interval time for the speed selected by the OTG\_FS controller during enumeration.

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

Bit	notation	instructions
31:16	Reserved	Reserved
15:0	FRIVL	<b>FRIVL:</b> Frame interval The application program uses this bit to configure the time interval between two consecutive SOF (full speed) or hold active (low speed). This register represents the frame interval in terms of the number of PHY clocks. The application program can set this register only after setting the Port Enable bit of the Host Port Control and Status Register (the PENA bit of the OTG_FS_HPRT register). If this register is not set, the controller will follow the Full/Low Speed PHY Clock Selection bit of the Host Configuration Register (FSLSPCS bit of the OTG_FS_HCFG register) as defined by the PHY clock to calculate the value. Do not modify this register after initialization. $1ms \times (\text{PHY clock frequency for full/low speed})$

### OTG\_FS host frame number/frame time remaining register (OTG\_FS\_HFNUM)

Offset address: 0x408

Reset value: 0x0000 3FFF

This register indicates the current frame number, and likewise how much time is left in the current frame (indicated by the number of PHY clocks).

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

Bit	notation	clarification
31:16	FTREM	<b>FTREM:</b> Frame time remaining (FTREM) Indicates how much time remains in the current frame, expressed as a number of PHY clocks. For each PHY clock, this value is subtracted by 1. When this value is subtracted to 0, the value defined in the Frame Interval Register is automatically loaded into this register and a new SOF is sent to the USB bus.
15:0	FRNUM	<b>FRNUM:</b> Frame number For each SOF signal sent to the USB bus, this field is automatically incremented by 1. When 0x3FFF is reached, the field is automatically reset to zero and the accumulation starts again.

### OTG\_FS Host Periodic Send FIFO/Request Queue Register (OTG\_FS\_HPTXSTS)

Offset address: 0x410

Reset value: 0x0008 0100

This register is a read-only register that holds information about the remaining space in the periodic transmit FIFO and request queue.

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

Bit	notation	clarification
31:24	PTXQTOP	<b>PTXQTOP:</b> periodic transmission request queue top (Top of the periodic transmit request queue) Indicates the periodic transmit request entry being processed by the MAC module. This register is used only for module debugging. Bit 31: Odd/Even Frames 0: Even Frames sent; 1: Odd frames are sent.

		Bits [30:27]: channel/endpoint number; bits [26:25]: type 00: IN/OUT; 01: zero-length packet; 11: abort channel command. Bit 24: End (last request for the selected channel/endpoint).
23:16	PTXQSAV	<b>PTXQSAV</b> : Periodic transmit request queue space available Indicates the remaining space in the periodic transmit request queue, which includes both IN and OUT requests. 00: periodic transmission request queue full; 01: d × 1 request position remaining; 10: d × 2 request positions remaining; b × n: remaining d × n request positions (0 ≤ d ≤ n ≤ 8); others: reserved.
15:0	PTXFSAVL	<b>PTXFSAVL</b> : Periodic transmit data FIFO space available indicates the remaining space in the periodic transmit FIFO. This value is a 32-bit word. 0000: Periodic send FIFO full; 0001: Remaining d × 1 word; 0010: Remaining d × 2 words; b × n: remaining d × n words (0 ≤ d ≤ n ≤ 512); b × 200: remaining d × 512 words; Other: Reserved.

### OTG\_FS Host All Channel Interrupt Register (OTG\_FS\_HAINT)

Offset address: 0x414

Reset value: 0x0000 0000

When a channel generates a signature event, a host all-channel interrupt interrupts the application via the host channel interrupt bit of the controller interrupt register (HCINT bit of the OTG\_FS\_GINTSTS register). Refer to Figure 280 for details. Each channel has a corresponding channel interrupt bit, for a total of 16 control bits. The application sets and clears this bit by setting and clearing the corresponding bit in the corresponding host channel x interrupt register.

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

Bit	notation	clarification
31:16	Reserved	Reserved
15:0	HAINT	<b>HAINT</b> : Channel interrupts Each bit corresponds to a channel: bit 0 corresponds to channel 0 and bit 15 corresponds to channel 15.

### OTG\_FS host all channel interrupt mask register (OTG\_FS\_HAINTMSK)

Offset address: 0x418

Reset value: 0x0000 0000

The host all-channel interrupt mask register is used in configuration with the host all-channel interrupt register to interrupt the application program when an event is generated. Each channel has a corresponding interrupt mask bit with a total of 16 bits.

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

Bit	notation	clarification
31:16	Reserved	Reserved
15:0	HAINTM	<b>HAINTM</b> : Channel interrupt mask 0: Mask interrupt; 1: Do not mask interrupts. Each bit corresponds to a channel: bit 0 corresponds to channel 0 and bit 15 corresponds to channel 15.

## OTG\_FS Host Port Control and Status Register (OTG\_FS\_HPRT)

Offset address: 0x440

Reset value: 0x0000 0000

This register is valid only in host mode. An OTG host controller supports only one port.

This register holds information related to the host port, including USB reset, enable, suspend, wakeup, connection status and test mode information for each port. Refer to Figure 280 for details. The bit labeled rc\_w1 can be used to trigger an interrupt that interrupts the application program via the host port interrupt bit of the host interrupt register (HPRTINT bit of the OTG\_FS\_GINTSTS register). In the port interrupt service program, the application program must read this register and clear the bit that caused the interrupt. For the bit labeled rc\_w1, the application needs to clear the interrupt by writing '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													PSPD	PTCTL			PPWR	PLSTS	Reserved	PRST	PSUSP	PRES	POCCHNG	POCA	PENCHNG	PENA	PCDET	PCSTS			
													r	r	rw	rw	rw	rw	rw	r	r	rw	rs	rw	rc_w1	r	rc_w1	rc_w1	rc_w1	r	

Bit	notation	clarification
31:19	Reserved	Reserved
18:17	PSPD	<b>PSPD:</b> Port speed indicates the speed of the device connected to the port. 01: Full-speed equipment; 10: Low-speed equipment; 11: Reserved.
16:13	PTCTL	<b>PTCTL:</b> Port test control The application program enters the test mode by writing a non-zero value to this bit, and a corresponding signal appears at the port. 0000: Non-test mode; 0001: Test_J mode; 0010: Test_K mode; 0011: Test_SE0_NAK mode; 0100: Test_Packet mode; 0101: Test_Force_Enable; Other: Reserved.
12	PPWR	<b>PPWR:</b> Port power The application program supplies power to the port by setting this bit, and the controller clears this bit when a current overflow event occurs. 0: No power supply; 1: Power supply.
11:10	PLSTS	<b>PLSTS:</b> Port line status (Port line status) indicates the current logical status of the USB data line. Bit 10: Logical state of the OTG_FS_FS_DP line; Bit 11: Logical state of the OTG_FS_FS_DM line.
9	Reserved	Reserved
8	PRST	<b>PRST:</b> Port reset When the application program sets this bit, a reset sequence is generated on the port. The application program needs to control the timing of the reset and clear this bit when the reset is complete. 0: The port is not reset; 1: Port reset. The application needs to hold this bit in the '1' state for at least 10ms in order to initiate the reset sequence for the port. The application can also add 10ms of '1' state before clearing this bit, the USB specification does not define a maximum duration for the reset signal.
7	PSUSP	<b>PSUSP:</b> Port suspend The application program sets this bit to put the port into a hung state. In this state the controller stops sending only SOF signals. The application program needs to stop the PHY clock by setting the Port Clock Stop bit. A read operation of this bit will return the current hung state of the port, and when the application program sets the Port Reset bit or the Port Wakeup bit of this register, or the controller detects a Remote Wakeup signal, or the Wakeup/Remote Wakeup Detect Interrupt bit or the Port Detect Interrupt bit of the Controller Interrupt Register (the WKUINT bit or the DISCINT bit of the OTG_FS_GINTSTS register) is set, the The controller will clear this bit. 0: The port is not in suspend mode; 1: The port is in suspend mode.
6	PRES	<b>PRES:</b> Port resume The application program sets this bit to drive a wake-up signal from the port. The controller will output a drive wake-up signal until the application clears this bit. When the controller detects a USB Remote Wakeup sequence, as indicated by the Port Wakeup/Remote Wakeup Detect Interrupt bit in the Controller Interrupt Register, the controller automatically outputs the wakeup sequence without application program intervention. If the controller detects that the

		device is disconnected, it will automatically clear this bit. A read operation of this bit will return information on whether the controller is outputting a wake-up signal. 0: No wake-up call; 1: Output wake-up signal.
5	POCCHNG	<b>POCCHNG</b> : Port overcurrent change The controller sets this bit when the port overcurrent bit (bit 4) of this register changes.
4	POCA	<b>POCA</b> : Port overcurrent active bit (Port overcurrent active) indicates whether overcurrent has occurred on the port. 0: No overcurrent; 1: Overcurrent.
3	PENCHNG	<b>PENCHNG</b> : Port enable/disable change (Port enable/disable change) The controller sets this bit when the port enable bit (bit 2) of this register changes.
2	PENA	<b>PENA</b> : Port enable The port can only be enabled by the controller after a reset sequence, and the port is disabled in the event of an overcurrent, a device disconnect being cleared by the controller with this bit. This bit cannot be set by the program through a register write operation. This bit does not trigger an interrupt. 0: Port disabled; 1: Port Enable.
1	PCDET	<b>PCDET</b> : Port connect detected When the controller detects that a device is connected to the port, it triggers the host port interrupt bit of the controller interrupt register (the HPRTINT bit of the OTG_FS_GINTSTS register) to generate an interrupt and set this bit. The application program must clear the interrupt by writing a '1'.
0	PCSTS	<b>PCSTS</b> : Port connect status 0: No device is connected to the port; 1: There is a device connected to the port.

#### OTG\_FS Host Channel x Characteristics Register (OTG\_FS\_HCCHARx) (here x codes the

channel number, x=0.... 7)

Offset address: 0x500+0x20\*x

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHENA	CHDIS	ODDFRM	DAD								MCNT	EPTYP	LSDEV	Reserved	EPDIR	EPNUM				MPSIZ											
RS	RS	RS	RS	RS	RS	RS	RS	RS	RS	RS	RS	RS	RS	RS	RS	RS	RS	RS	RS	RS	RS	RS	RS	RS	RS	RS	RS	RS	RS	RS	RS

Bit	notation	clarification
31	CHENA	<b>CHENA</b> : Channel enable This bit is set by the application program and cleared by the FS_OTG host controller. 0: Channel prohibited; 1: Channel Enable.
30	CHDIS	<b>CHDIS</b> : Channel disable By setting this bit, the application program can immediately stop the receipt/transmission of data on that channel, even if the transmission of data on that channel has not been completed. The application program can only assume that the channel is disabled if it waits until the channel disable interrupt is generated.
29	ODDFRM	<b>ODDFRM</b> : Odd frame The application program tells the OTG host controller that it should transmit on odd/even frames by setting/clearing this bit. This bit is only valid for periodic transmissions (synchronized or interrupted). 0: Even frames; 1: Odd frames.
28:22	DAD	<b>DAD</b> : Device address The application program selects the desired device via this address.
21:20	MCNT	<b>MCNT</b> : Device Address (Multicount) This field indicates the number of transmissions the host must perform per frame on this periodic endpoint. This parameter is not used for non-periodic transmissions. 00: Reserved. Setting this value will produce undefined results. 01: 1 transmission. 10: At this endpoint, 2 transmissions need to be generated per frame. 11: At this endpoint, 3 transmissions need to be generated per frame. Note: The value of this field should be set to at least '01'.
19:18	EPTYP	<b>EPTYP</b> : Endpoint type indicates the type of transmission selected 00: Control Transmission; 01: Synchronized transmission;

		10: Block Transfer; 11: Interruption of transmission.
17	LSDEV	LSDEV: Low speed device (Low speed device) The application program sets this bit to indicate that the device with which it is communicating is a low-speed device.
16	Reserved	Reserved
15	EPDIR	EPDIR: Endpoint direction indicates whether the transmission is in the IN or OUT direction. 0: OUT 1: IN
14:11	EPNUM	EPNUM: Endpoint number (Endpoint number) indicates the endpoint number with which to communicate.
10:0	MPSIZ	MPSIZ: Maximum packet size indicates the maximum packet length for the selected endpoint.

### OTG\_FS host channel x interrupt register (OTG\_FS\_HCINTx) (where x represents the

channel number, x=0....7,)

Offset address: 0x508 + (channel number x 0x20)

Reset value: 0x0000 0000

This register indicates the status of the channel with respect to USB and AHB related time. Refer to Figure 280 for details. When the host channel interrupt bit of the controller interrupt register (HCINT bit of the OTG\_FS\_GINTSTS register) is set, the application program needs to read the host all channel interrupt register (OTG\_FS\_HAINT) first to get the channel number on which the host channel interrupt occurs, and then read the interrupt registers of the corresponding channels to get the detailed information of the interrupt. The application program clears the corresponding bits of the OTG\_FS\_HAINT register and the OTG\_FS\_GINTSTS register by clearing the corresponding bits of this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																					DTERR	FRMOR	BBERR	TXERR	Reserved	ACK	NAK	STALL	Reserved	CHH	XFRC
																					rc_w1	rc_w1	rc_w1	rc_w1		rc_w1	rc_w1	rc_w1		rc_w1	rc_w1

Bit	notation	clarification
31:11	Reserved	Reserved
10	DTERR	DTERR: Data PID error (Data toggle error)
9	FRMOR	FRMOR: Frame overrun
8	BBERR	BBERR: crosstalk error (Babble error)
7	TXERR	TXERR: Transaction error indicates that the following error occurred: CRC checksum failure timeout Bit Fill Error EOP Failed
6	Reserved	Reserved
5	ACK	ACK: ACK response received/transmitted interrupt
4	NAK	NAK: NAK response received interrupt
3	STALL	STALL: STALL response received interrupt
2	Reserved	Reserved
1	CHH	CHH: Channel halted Indicates an abnormal end of transmission due to a USB transmission error, or an application program abort request.
0	XFRC	XFRC: Transfer completed (Transfer completed) The transfer was completed normally without error.

**OTG\_FS host channel x interrupt mask register (OTG\_FS\_HCINTMSKx) (where x is the channel number, x=0.... 7)**

Offset address: 0x50C + (channel number x 0x20)

Reset value: 0x0000 0000

This register is used to mask the various types of channel interrupts described in the previous section.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALML	Reserved	CHHM	XFRM
																					r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w

Bit	notation	clarification
31:11	Reserved	Reserved
10	DTERRM	<b>DTERRM</b> : Data PID error interrupt mask (Data toggle error mask) 0: Mask interrupt; 1: Do not mask interrupts.
9	FRMORM	<b>FRMORM</b> : Frame overrun mask 0: Mask interrupt; 1: Do not mask interrupts.
8	BBERRM	<b>BBERRM</b> : crosstalk error interrupt mask (Babble error mask) 0: Mask interrupt; 1: Do not mask interrupts.
7	TXERRM	<b>TXERRM</b> : Transaction error mask 0: Mask interrupt; 1: Do not mask interrupts.
6	NYET	<b>NYET</b> : received response interrupt mask (response received interrupt mask) 0: Mask interrupt; 1: Do not mask interrupts.
5	ACKM	<b>ACKM</b> : ACK response received/transmitted interrupt mask 0: Mask interrupt; 1: Do not mask interrupts.
4	NAKM	<b>NAKM</b> : NAK response received interrupt mask 0: Mask interrupt; 1: Do not mask interrupts.
3	STALML	<b>STALML</b> : STALL response received interrupt mask (STALL response received interrupt mask) 0: Mask interrupt; 1: Do not mask interrupts.
2	Reserved	Reserved
1	CHHM	<b>CHHM</b> : Channel halted mask 0: Mask interrupt; 1: Do not mask interrupts.
0	XFRM	<b>XFRM</b> : Transfer completed mask 0: Mask interrupt; 1: Do not mask interrupts.

**OTG\_FS host channel x transmission length register (OTG\_FS\_HCTSIZx) (where x is the channel number, x=0.... 7)**

Offset address: 0x510 + (channel number x 0x20)

Reset value: 0x0000 0000

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

Bit	notation	clarification
31	Reserved	Reserved
30:29	DPID	<b>DPID</b> : Data PID (Data PID) The application program uses this bit to inform the controller of the type of PID used for the first transmission. The controller will automatically control the PID type for subsequent transmissions.

		00: DATA0; 01: DATA2; 10: DATA1; 11: MDATA (non-control)/SETUP (control).
28:19	PKTCNT	<b>PKTCNT:</b> Packet count The application program uses this bit to inform the controller of the number of packets it expects to receive (IN) or the number of packets it expects to send (OUT). The controller will automatically decrement this parameter after each successful transmission or reception of an OUT/IN packet, and once the field is 0, the application will receive an interrupt indicating that the transmission ended normally.
18:0	XFRSIZ	<b>XFRSIZ:</b> Transfer size For OUT transfers, these bits indicate the number of data bytes sent by the host during the transfer. For IN transmissions, these bits indicate the length of the buffer reserved by the application. For IN transfers (periodic and non-periodic), the application needs to predefine this parameter in integer multiples of the maximum packet.

## 29.16.4 Registers in Device Mode

### OTG\_FS Device Configuration Register (OTG\_FS\_DCFG)

Offset address: 0x800

Reset value: 0x0220 0000

After a reset, or special control command, or enumeration, this register is used to configure the operating characteristics of the controller in device mode. Do not modify this register after initialization.

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

Bit	notation	clarification
31:13	Reserved	Reserved
12:11	PFIVL	<b>PFIVL:</b> Periodic frame interval (Periodic frame interval) Indicates the time spent generating a periodic end-of-frame interrupt as a percentage of the entire frame. The application program can use this interrupt to determine if all synchronous transmissions within a frame have been transmitted. 00: 80% frame time; 01: 85% frame time; 10: 90% frame time; 11: 95% frame time.
10:4	DAD	<b>DAD:</b> Device address The application program fills this bit according to the parameters after receiving the SetAddress control command.
3	Reserved	Reserved
2	NZLSOHSK	<b>NZLSOHSK:</b> Non-zero-length status OUT handshake signal (Non-zero-length status OUT handshake) During the status phase of the control transfer, the application can choose to send a handshake signal with this bit if a non-zero length packet is received. 1: Sends the STALL handshake to a non-zero length state OUT transmission and does not transmit the received OUT packet to the application. 0: Depending on the state of the NAK bit and the STALL bit of the device port control register, the handshake signal is selected to be sent and the received OUT packet is transmitted to the application program (zero-length and non-zero-length).
1:0	DSPD	<b>DSPD:</b> Device speed Indicates the speed at which the application needs the controller to perform an enumeration operation, or the maximum speed that the application can support. However, the actual bus speed can only be determined after the entire sequence has been completed, based on the speed of the connected USB host. 00: Reserved; 01: Reserved; 10: Reserved; 11: Full Speed (USB 1.1 transceiver clocked at 48MHz).



## OTG\_FS Device Control Register (OTG\_FS\_DCTL)

Offset address: 0x804

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																				POPRGDNE	CGONAK	SGONAK	CGINAK	SGINAK	TCTL		GONSTS	GINSTS	SDIS	PWUSIG	
																				RW	W	W	W	W	RW	RW	RW	R	R	RW	RW

Bit	notation	instructions
31:12	Reserved	Reserved
11	POPRGDNE	<b>POPRGDNE:</b> Power-on programming done The application program uses this bit to indicate that configuration of the registers has been completed after waking up from the power-up state.
10	CGONAK	<b>CGONAK:</b> Clear global OUT NAK A write operation to this bit clears the global OUTNAK state.
9	SGONAK	<b>SGONAK:</b> Set global OUT NAK A write operation to this bit sets the global OUTNAK state. The application program uses this bit to tell the controller to send the NAK signal to all OUT endpoints. The program must verify that the global OUT NAK valid bit of the controller interrupt register (GONAKEFF bit of the OTG-FS_GINTSTS register) is cleared before setting this bit.
8	CGINAK	<b>CGINAK:</b> Clear global INNAK ((Clear global IN NAK)) Writing this bit will clear the global INNAK state.
7	SGINAK	<b>SGINAK:</b> Set global INNAK (SetglobalINNAK) A write operation to this bit will set the NAK state of the global nonperiodic IN. The program uses this bit to tell the controller to send the NAK handshake signal for all nonperiodic IN endpoints. The program must verify that the global IN NAK valid bit of the controller interrupt register (GINAKEFF bit of the OTG-FS_GINTSTS register) is cleared before setting this bit.
6:4	TCTL	<b>TCTL:</b> Test control 000: Test mode disabled; 001: Test_J mode; 010: Test_K mode; 011: Test_SE0_NAK mode; 100: Test_Packet mode; 101: Test_Force_Enable; Other: Reserved.
3	GONSTS	<b>GONSTS:</b> Global OUT NAK status 0: Sends a handshake signal based on the state of the FIFO and the state of the NAK bit and the STALL bit. 1: No data is written to the receive FIFO, regardless of whether the storage area is empty. sends the NAK handshake signal to all transmissions except SETUP, discarding all synchronized OUT packets.
2	GINSTS	<b>GINSTS:</b> Global IN NAK status 0: Handshake signal is sent according to the status of the data in the sending FIFO. 1: The NAK handshake signal is sent to all non-periodic IN endpoints regardless of the state of the data in the sending FIFO.
1	SDIS	<b>SDIS:</b> Soft disconnect The application program uses this bit to tell the OTG controller to perform a software disconnect operation. When this bit is set, the host will see that the device is disconnected and the device side will not receive any signal from the USB bus. The controller will remain in the disconnected state until the program clears this bit. 0: Normal operation. When this bit is cleared after a device software disconnect, the controller will send a device connection event to the host and the host will re-execute the enumeration operation. 1: The controller performs a device software disconnect operation.
0	RWUSIG	<b>RWUSIG:</b> Remote wakeup signaling When the application program sets this bit, the controller will send a remote wake-up signal to wake up the USB host. The application program must set this bit to make the controller exit the suspend state. According to USB2.0 specification, the program must clear this bit again between 1-15ms after setting it.

In order for the USB host to recognize a device disconnect operation, the Software Disconnect (SDIS) bit needs to be held for a certain period of time, and the following table lists the minimum durations (depending on the state of the device). To accommodate clock jitter, it is

recommended that the application program retains an additional period of delay beyond the defined minimum duration.

operating speed	device status	minimum duration
at full speed	pending	1ms+2.5us
at full speed	leisure time	2.5us
at full speed	Not idle or hung (transmission operation is being performed)	2.5us

### OTG\_FS Device Status Register (OTG\_FS\_DSTS)

Offset address: 0x808

Reset value: 0x0000 0010

This register indicates the status of the controller in relation to USB. This register is used to be read in the event of a Device All Interrupt (OTG\_FS\_DAINTE) register event.

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

Bit	notation	clarification
31:22	Reserved	Reserved
21:8	FNSOF	FNSOF: Frame number of the received SOF
7:4	Reserved	Reserved
3	EERR	<b>EERR:</b> Strange error (Erratic error) The controller sets this bit when some strange error occurs. If a strange error occurs, the OTG_FS controller will go into a hang state and set the early hang bit of the controller interrupt register (ESUSP bit of the OTG_FS_GINTSTS register) to generate an interrupt with it. If the early hang interrupt is caused by this bit, the application program can only resolve it by performing a software disconnect.
2:1	ENUMSPD	<b>ENUMSPD:</b> Enumerated speed (Enumerated speed) indicates the execution speed selected by the OTG_FS controller through the sequence. 01: Reserved; 10: Reserved; 11: Full speed (PHY clock running at 48MHz); Other: Reserved.
0	SUSPSTS	<b>SUSPSTS:</b> Suspend status In device mode, this bit will be set if a hang condition is detected on the USB bus. The controller will enter a hang condition if it detects no activity on the USB data line for 3ms. The controller exits the hang mode at the following conditions: When there is activity on the USB cable When the application sets the Remote Wakeup Signal bit of the Device Control Register (RWUSIG bit of the OTG_FS_DCTL register)

### OTG\_FS device IN endpoint generalized interrupt mask register (OTG\_FS\_DIEPMSK)

Offset address: 0x810

Reset value: 0x0000 0000

This register is used in conjunction with the device IN endpoint interrupt register (OTG\_FS\_DIEPINTx) to generate IN endpoint interrupts. The corresponding IN endpoint interrupt corresponding to the OTG\_FS\_DIEPINTx register can be masked by configuring this register. All interrupts are masked by default.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
Reserved																		NAKM	Reserved										INPNEM	INPNMM	ITTXFEMSK	TOM	Reserved	EPDM	XFRM
																		r w											r w	r w	r w	r w	r w	r w	r w

Bit	notation	clarification
31:14	Reserved	Reserved
13	NAKM	<b>NAKM:</b> NAK Interrupt Mask 0: Mask interrupt 1: Unmask interrupt

12:7	Reserved	Reserved
6	INEPNEM	<b>INEPNEM:</b> IN endpoint NAK effective mask 0: Interrupt Mask; 1: Interrupts are not masked.
5	INEPNMM	<b>INEPNMM:</b> Endpoint received IN command mismatch interrupt mask (IN token received with EP mismatch mask) 0: Interrupt Mask; 1: Interrupts are not masked.
4	ITTXFEMSK	<b>ITTXFEMSK:</b> IN command received when TxFIFO empty mask (IN token received when TxFIFO empty mask) 0: Interrupt Mask; 1: Interrupts are not masked.
3	TOM	<b>TOM:</b> Timeout condition mask (Non-isochronous endpoints) 0: Interrupt Mask; 1: Interrupts are not masked.
2	Reserved	Reserved
1	EPDM	<b>EPDM:</b> Endpoint disabled interrupt mask 0: Interrupt Mask; 1: Interrupts are not masked.
0	XFCRM	<b>XFCRM:</b> Transfer completed interrupt mask 0: Interrupt Mask; 1: Interrupts are not masked.

#### OTG\_FS device OUT endpoint generalized interrupt mask register (OTG\_FS\_DOEPMSK)

Offset address: 0x814

Reset value: 0x0000 0000

This register is used in conjunction with the device OUT endpoint interrupt register (OTG\_FS\_DOEPINTx) to generate OUT endpoint interrupts. Each bit of the OTG\_FS\_DOEPINTx register can be masked by writing the corresponding bit of this register. In the default state, all interrupts are masked.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																		NAK	BERRM	Reserved			OUTPKTERRM	Reserved		STSPHSRXM	OTEPDM	STUPM	Reserved	EPDM	XFCRM

Bit	notation	clarification
31:14	Reserved	Reserved
13	NAKMSK	<b>NAKMSK:</b> NAK Interrupt Mask 0: Interrupt Mask; 1: Interrupts are not masked.
12	BERRM	<b>BERRM:</b> Burble Error Interrupt Mask 0: Interrupt Mask; 1: Interrupts are not masked.
11:9	Reserved	Reserved
8	OUTPKTERRM	<b>OUTPKTERRM:</b> Output packet error mask 0: Interrupt Mask; 1: Interrupts are not masked.
7:6	Reserved	Reserved
5	STSPHSRXM	<b>STSPHSRXM:</b> Control write mask status phase received 0: Interrupt Mask; 1: Interrupts are not masked.
4	OTEPDM	<b>OTEPDM:</b> OUT command received when endpoint disabled interrupt mask (OUT token received when endpoint disabled mask) 0: Interrupt Mask; 1: Interrupts are not masked.
3	STUPM	<b>STUPM:</b> SETUP phase done mask valid only for control endpoints 0: Interrupt Mask; 1: Interrupts are not masked.
2	Reserved	Reserved
1	EPDM	<b>EPDM:</b> Endpoint disabled interrupt mask 0: Interrupt Mask; 1: Interrupts are not masked.
0	XFCRM	<b>XFCRM:</b> Transfer completed interrupt mask 0: Interrupt Mask; 1: Interrupts are not masked.

### OTG\_FS Device All Endpoint Interrupt Register (OTG\_FS\_DAIN)

Offset address: 0x818

Reset value: 0x0000 0000

When an event occurs at each endpoint, all of the device's endpoint interrupt registers generate an interrupt to interrupt the application program through the device OUT endpoint interrupt bit or the device IN endpoint interrupt bit of the controller's interrupt register (the OEPINT or IEPINT bit of the OTG\_FS\_GINTSTS register), respectively. Each endpoint has a corresponding bit, 16 bits for the OUT endpoint and 16 bits for the IN endpoint. For bidirectional endpoints, both IN and OUT bits are used. The corresponding bits of this register are automatically set and cleared when the application sets and clears the corresponding bits of the corresponding device endpoint x interrupt registers (OTG\_FS\_DIEPINTx and OTG\_FS\_DOEPINTx registers).

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

Bit	notation	clarification
31:16	OEPINT	<b>OEPINT:</b> OUT endpoint interrupt bits (OUT endpoint interrupt bits) Each bit corresponds to an OUT endpoint. Bit 16 corresponds to OUT endpoint 0 and bit 18 corresponds to OUT endpoint 3.
15:0	IEPINT	<b>IEPINT:</b> IN endpoint interrupt bits (IN endpoint interrupt bits) Each bit corresponds to an IN endpoint. Bit 0 corresponds to IN endpoint 0 and bit 3 corresponds to IN endpoint 3.

### OTG\_FS all endpoint interrupt mask register (OTG\_FS\_DAINMSK)

Offset address: 0x81C

Reset value: 0x0000 0000

The Device All Endpoint Interrupt Mask Register is used in conjunction with the Device Endpoint Interrupt Register to generate an interrupt to interrupt the application program when a device endpoint event occurs. However, the corresponding bit in the Device All Endpoint Interrupt Register (OTG\_FS\_DAIN) remains set when masked.

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

Bit	notation	clarification
31:16	OEPM	<b>OEPM:</b> OUT Endpoint Interrupt Mask Register (OUT EP interrupt mask bits) Each bit corresponds to one OUT endpoint. Bit 16 corresponds to OUT endpoint 0 and bit 31 corresponds to OUT endpoint 15. 0: Mask interrupt; 1: Do not mask interrupts.
15:0	IEPM	<b>IEPM:</b> IN endpoint interrupt mask bits (INEPinterruptmaskbits) Each bit corresponds to an IN endpoint. Bit 0 corresponds to IN endpoint 0 and bit 15 corresponds to IN endpoint 15. 0: Mask interrupt; 1: Do not mask interrupts.

### OTG\_FS Device VBUS Discharge Time Register (OTG\_FS\_DVBUSDIS)

Offset address: 0x828

Reset value: 0x0000 17D7

This register defines the VBUS discharge time after the VBUS pulse during SRP.

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

Bit	notation	clarification
31:16	Reserved	Reserved
15:0	VBUSDT	<b>VBUSDT:</b> Device VBUS discharge time (Device VBUS discharge time) Defines the VBUS discharge time after the VBUS pulse during SRP. This value is equal to the VBUS discharge time/1024 calculated with the PHY clock. Depending on the VBUS load, this value can be adjusted appropriately.

## OTG\_FS Device VBUS Pulse Time Register (OTG\_FS\_DBVUSPULSE)

Offset address: 0x82C

Reset value: 0x0000 05B8

This register defines the timing of the VBUS pulse during SRP.

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

Bit	notation	clarification
31:12	Reserved	Reserved
11:0	DVBUSP	<b>DVBUSP:</b> Device VBUS pulsing time (Device VBUS pulsing time) Defines the VBUS pulse time during SRP. This value is equal to the VBUS pulse time/1024 calculated with the PHY clock.

## OTG\_FS device IN endpoint FIFO air break mask register (OTG\_FS\_DIEPEMPMSK)

Offset address: 0x834

Reset value: 0x0000 0000

This register is used in conjunction with the IN endpoint FIFO air interrupt register (TXFE\_OTG\_FS\_DIEPINTx) to generate an interrupt.

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

Bit	notation	clarification
31:16	Reserved	Reserved
15:0	INEPTXFEM	<b>INEPTXFEM:</b> IN Endpoint Transmit FIFO empty interrupt mask bits (IN EP Tx FIFO empty interrupt mask bits) This bit is used to mask the corresponding bits of the OTG_FS_DIEPINTx register. A bit of the TXFE interrupt corresponds to an IN endpoint: bit 0 corresponds to IN endpoint 0 and bit 3 corresponds to IN endpoint 3. 0: Mask interrupt; 1: Do not mask interrupts.

## OTG\_FS device control IN endpoint 0 control register (OTG\_FS\_DIEPCTL0)

Offset address: 0x900

Reset value: 0x0000 0000

This section describes the device control IN endpoint 0 control registers. Non-0 control endpoints use the registers corresponding to endpoints 1-15.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EPENA	EPDIS	Reserved	SNAK	CNAK	TXFNUM				STALL	Reserved	EPTYP	NAKSTS	Reserved	USBAEP	Reserved																MPSIZ
rs	r		w	w	rw	rw	rw	rw	rs		r	r	r	r																	rw

Bit	notation	clarification
31	EPENA	<b>EPENA:</b> Endpoint enable The application program sets this bit to initiate a data transfer on endpoint 0. The controller clears this bit before generating the following endpoint interrupts: Endpoint disabled; transmission complete.
30	EPDIS	<b>EPDIS:</b> Endpoint disable An application program can stop data transmission on an endpoint immediately by setting this bit, even if the transmission has not completed. The application program needs to wait until the endpoint disable interrupt to make sure this endpoint is disabled. The controller clears this bit when the endpoint disable interrupt is set. The application program can only set this bit if the endpoint is already enabled.
29:28	Reserved	Reserved
27	SNAK	<b>SNAK:</b> Set NAK (Set NAK) Writing this bit sets the NAK state of the endpoint. By setting this bit, the program can tell the controller to send the NAK handshake signal. The controller also sets this bit when it receives a SETUP

		packet.
26	CNAK	CNAK: ClearNAK Write this bit to clear the NAK state of the endpoints
25:22	TXFNUM	TXFNUM: Transmit FIFO number (TxFIFO number) This bit sets the FIFO number assigned to IN endpoint 0.
21	STALL	STALL: The STALL handshake. The application program can only set this bit, and the controller clears it when it receives a SETUP command. The STALL bit has high priority even if the NAK bit, or the global INNAK bit, or the global OUTNAK bit is also set.
20	Reserved	Reserved
19:18	EPTYP	EPTYP: Endpoint type (for control endpoints, set to '00' by hardware).
17	NAKSTS	NAKSTS: NAK status indicates the following: 0: Depending on the FIFO status, the controller sends a non-NAK handshake signal; 1: The controller sends the NAK handshake signal. As soon as this bit is set, whether the application program sets this bit or the controller sets this bit, the controller stops the data transfer regardless of the validity of the data in the sending FIFO. Regardless of this bit setting, the controller will always send an ACK handshake in response to a SETUP packet.
16	Reserved	Reserved
15	USBAEP	USBAEP: USB active endpoint This bit is always '1', indicating that control endpoint 0 is active in any configuration situation.
14:2	Reserved	Reserved
1:0	MPSIZ	MPSIZ: Maximum packet size The application program configures the maximum packet length for this endpoint with this bit 00: 64 bytes; 01: 32 bytes; 10: 16 bytes; 11: 8 bytes.

### OTG device endpoint x control register (OTG\_FS\_DIEPCTLx) (where x is the endpoint number, x=1..3)

Offset address: 0x900 + (endpoint number x 0x20)

Reset value: 0x0000 0000

The application program controls the operating characteristics of the non-zero endpoints through these registers.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EPENA	EPDIS	SODDFRM	SDOPID/SEVNFRM	SNAK	CNAK	TXFNUM			STALL	Reserved	EPTYP	NAKSTS	EONUM/DPID	USBAEP	Reserved			MPSIZ													
rs	rs	w	w	w	w	rw	rw	rw	rw	rw		rw	rw	r	r	rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit	notation	clarification
31	EPENA	EPENA: End point enable The application program sets this bit to initiate a data transfer on the endpoint. The controller clears this bit before generating the following endpoint interrupts: SETUP phase completed Endpoint Disable Transmission Completion
30	EPDIS	EPDIS: End point disable An application program can stop data transmission on an endpoint immediately by setting this bit, even if the transmission has not completed. The application program needs to wait until the endpoint disable interrupt to make sure this endpoint is disabled. The controller clears this bit when the endpoint interrupt is set. The application program can only set this bit if the endpoint is already enabled.
29	SODDFRM	SODDFRM: Set odd frame (Set odd frame) is only valid for synchronized IN and

		OUT endpoints. Write this bit to select odd frames in EONUM.
28	SDOPID	<b>SDOPID:</b> Set DATA0PID (Set DATA0 PID) is valid only for interrupt/block transfer IN endpoints. Writing this bit will set the data PID bit to DATA0. <b>SEVNFRM:</b> Setting Even Frames Valid only for synchronized IN endpoints. Write this bit to select even frames in EONUM.
27	SNACK	<b>SNACK:</b> Set NAK (Set NAK) Writing this bit sets the NAK state of the endpoint By setting this bit, the program can tell the controller to send the NAK handshake signal. The controller also sets this bit when it receives a SETUP packet or an end-of-transmission interrupt from the OUT endpoint.
26	CNAK	<b>CNAK:</b> Clear NAK (Clear NAK) Writing this bit clears the NAK state of an endpoint
25:22	TXFNUM	<b>TXFNUM:</b> Transmit FIFO number (TxFIFO number) This bit sets the FIFO number assigned to this endpoint; a separate FIFO number must be assigned to each valid IN endpoint. This bit is only valid for IN endpoints.
21	STALL	<b>STALL:</b> The STALL handshake. Valid only for non-control, non-synchronized IN endpoints (operation type rw) The application program sets this bit and this endpoint responds to all host commands with STALL. The STALL bit has the highest priority even if the NAK bit, or the global IN NAK bit, or the global OUT NAK bit is also set. Only the application program can clear this bit; the controller cannot. Only valid for control endpoints (operation type is rs) The application program can only set this bit, and the controller clears it when it receives a SETUP command. The STALL bit has high priority even if the NAK bit, or the global IN NAK bit, or the global OUT NAK bit is also set. However the controller will always respond to SETUP packets with an ACK handshake.
20	Reserved	Reserved
19:18	EPTYP	<b>EPTYP:</b> Endpoint type This bit indicates the transmission type of the endpoint. 00: Control; 01: Synchronization; 10: block pass; 11: interrupt;
17	NAKSTS	<b>NAKSTS:</b> NAK status (NAK status) indicates the following status: 0: Depending on the FIFO status, the controller sends a non-NAK handshake; 1: The controller sends a NAK handshake signal. As long as this bit is set, it doesn't matter if it's an application setting or a controller setting: For non-synchronized IN endpoints, the controller stops data transmission regardless of the validity of the data in the sending FIFO. For synchronized IN endpoints, the controller will send a zero-length packet even if the data in the transmit FIFO is valid. Regardless of this bit setting, the controller will always send an ACK handshake in response to a SETUP packet.
16	EONUM	<b>EONUM:</b> Even/odd frame is valid only for synchronized IN endpoints: Indicates the frame number for synchronized data sending and receiving at this endpoint. The program must apply the state of the SEVNFRM and SODDFRM bits in this register to set the even/odd frame number of the desired frame to be sent/received. 0: even frames; 1: odd frames. <b>DPID:</b> Endpoint data PID (Endpoint data PID) is valid only for interrupt/block transfer IN endpoints: This bit holds the PID of the packet to be transmitted through this endpoint. The application program must configure this bit to select the PID of the first packet to be sent or received after enabling the endpoint. The application program configures DATA0 or DATA1 through the SDOPID register bit. 0: DATA0; 1: DATA1.
15	USBAEP	<b>USBAEP:</b> USB active endpoint Indicates whether this endpoint is an active endpoint in the current configuration. The controller clears this bit for all endpoints (except endpoint 0) when it detects a USB reset, and the application program must set this bit as necessary after receiving the SetConfiguration and SetInterface commands.
14:11	Reserved	Reserved
10:0	MPSIZ	<b>MPSIZ:</b> Maximum packet size The application program configures the maximum packet length for this endpoint with this bit. The value of this bit is in bytes.



## OTG\_FS device control OUT endpoint 0 control register (OTG\_FS\_DOEPCTL0)

Offset address: 0xB00

Reset value: 0x0000 8000

This section describes the control registers for device control OUT endpoint 0. Non-0 control endpoints use the corresponding endpoint 1-15 control registers.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EPENA	EPDIS	Reserved		SNAK	CNAK	Reserved					STALL	SNPM	EPTYP		NAKSTS	Reserved	USBAEP	Reserved												MPSIZ	
w	r			w	w					rs	rw	r	r	r		r														r	

Bit	notation	clarification
31	EPENA	<b>EPENA:</b> Endpoint enable The application program sets this bit to initiate a data transfer on endpoint 0. The controller clears this bit before generating the following endpoint interrupts: SETUP phase complete; endpoints disabled; Transmission complete.
30	EPDIS	<b>EPDIS:</b> Endpoint disable The application program cannot disable control of OUT endpoint 0.
29:28	Reserved	Reserved
27	SNAK	<b>SNAK:</b> Set NAK (Set NAK) Writing this bit sets the NAK state of the endpoint. By setting this bit, the application program can tell the controller to send the NAK handshake signal. The controller also sets this bit when it receives a SETUP packet or an end-of-transmission interrupt.
26	CNAK	<b>CNAK:</b> Clear NAK (Clear NAK) Setting this bit clears the NAK state of the endpoint.
25:22	Reserved	Reserved
21	STALL	<b>STALL:</b> The STALL handshake. The application program can only set this bit; it is cleared by the controller when a SETUP command is received. The STALL bit has high priority even if the NAK bit, or the global OUTNAK bit, is also set. Regardless of how this bit is set, however, the controller always responds to SETUP packets with an ACK handshake.
20	SNPM	<b>SNPM:</b> Snoop mode Setting this bit puts the endpoint into listen mode, in which the controller will not detect the correctness of the OUT packet before transmitting the data into the application buffer.
19:18	EPTYP	<b>EPTYP:</b> Endpoint type (Endpoint type) is set to '00' by hardware.
17	NAKSTS	<b>NAKSTS:</b> NAK status (NAK status) indicates the following status: 0: Depending on the FIFO status, the controller sends a non-NAK handshake signal. 1: The controller sends a NAK handshake signal. As long as this bit is set, it doesn't matter if it's an application setting or a controller setting: The controller will stop the data transfer regardless of the validity of the data in the receive FIFO. Regardless of this bit setting, the controller will always send an ACK handshake signal in response to a SETUP packet.
16	Reserved	Reserved
15	USBAEP	<b>USBAEP:</b> USB active endpoint is always '1', indicating that control endpoint 0 is active in any configuration.
14:2	Reserved	Reserved
1:0	MPSIZ	<b>MPSIZ:</b> Maximum packet size The maximum packet length for control OUT endpoint 0 must match the maximum packet length for control IN endpoint 0. 00: 64 bytes; 01: 32 bytes; 10: 16 bytes; 11: 8 bytes.



**OTG\_FS device OUT endpoint x control register (OTG\_FS\_DOEPCTLx) (where x is the endpoint number, x=1..3)**

Offset address: 0xB00 + (endpoint number x 0x20)

Reset value: 0x0000 0000

The application program uses this register to control the operating characteristics of endpoints other than endpoint 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																				
EPENA		EPDIS		SODDFRM		SDOPID/SEVNFRM		SNAK		CNAK		Reserved				STALL		SNPM		EPTYP		NAKSTS		EONUM/DPID		USBAEP		Reserved				MPSIZ																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																			
rs		rs		w		w		w		w						rw		rw		rw		rw		r		r		rw						rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw		rw	

Bit	notation	clarification
31	EPENA	<b>EPENA:</b> Endpoint enable is valid for both IN and OUT endpoints. The application program sets this bit to initiate a data transfer on the endpoint. The controller clears this bit before generating the following endpoint interrupts: SETUP phase complete; endpoints disabled; Transmission complete.
30	EPDIS	<b>EPDIS:</b> Endpoint disable An application program can stop data transmission on an endpoint immediately by setting this bit, even if the transmission has not completed. The application program needs to wait for an endpoint disable interrupt to confirm that this endpoint has been disabled. The controller clears this bit when the endpoint interrupt is set. The application program can only set this bit if the endpoint is already enabled.
29	SODDFRM	<b>SODDFRM:</b> Set odd frame (Set odd frame) is only valid for synchronized OUT endpoints Setting this bit selects the odd frame in the parity frame field (EONUM).
28	SDOPID	<b>SDOPID:</b> Set DATA0PID (Set DATA0 PID) is valid only for interrupt/block transfer OUT endpoints. Writing this bit will set the data PID bit to DATA0. <b>SEVNFRM:</b> Setting Even Frames Valid only for synchronized OUT endpoints. Setting this bit selects even frames in the parity frame field (EONUM).
27	SNAK	<b>SNAK:</b> Set NAK (Set NAK) Writing this bit sets the NAK state of the endpoint. By setting this bit, the application program can control the controller to send the NAK handshake signal. The controller also sets this bit when it receives a SETUP packet or an end-of-transmission interrupt from the OUT endpoint.
26	CNAK	<b>CNAK:</b> Clear NAK (Clear NAK) Writing this bit clears the NAK state of the endpoint.
25:22	Reserved	Reserved
21	STALL	<b>STALL:</b> The STALL handshake. Valid only for non-control, non-synchronized OUT endpoints (operation type rw) The application program sets this bit and this endpoint responds to all commands from the host with STALL. The STALL bit has the highest priority even if the NAK bit, or the global INNAK bit, or the global OUTNAK bit is also set. Only the application program can clear this bit; the controller cannot. Only valid for control endpoints (operation type is rs) The application program can only set this bit, and the controller clears it when it receives a SETUP command. The STALL bit has high priority even if the NAK bit, or the global INNAK bit, or the global OUTNAK bit is also set. However the controller will always respond to SETUP packets with an ACK handshake.
20	SNPM	<b>SNPM:</b> Snoop mode Setting this bit will put the endpoint into listen mode. In listen mode, the controller does not check the correctness of the data before writing the OUT packet to the application buffer.
19:18	EPTYP	<b>EPTYP:</b> Endpoint type This bit indicates the transmission type of the endpoint: 00: Control 01: Synchronization 10: Block Transfer 11: Interrupt
17	NAKSTS	<b>NAKSTS:</b> NAK status (NAK status) indicates the following status: 0: Depending on the FIFO status, the controller sends a non-NAK handshake signal. 1: The controller sends a NAK handshake signal.

		As long as this bit is set, it doesn't matter if it's an application setting or a controller setting: The controller will stop data transmission regardless of whether the receiving FIFO can receive data. The controller will always send an ACK handshake in response to a SETUP packet regardless of this bit setting.
16	EONUM	<b>EONUM:</b> Even/odd frame is valid only for synchronized OUT endpoints: Indicates the frame number for synchronized data sending and receiving at this endpoint. The program must set the even/odd frame number according to the status of the SEVNFRM and SODDFRM bits in this register. 0: even frames; 1: odd frames. <b>DPID:</b> Endpoint data PID (Endpoint data PID) is valid only for interrupt/block transfer OUT endpoints: This bit holds the PID of the packet to be transmitted through this endpoint. The program must configure this bit after enabling the endpoint to select the PID of the first packet to be sent or received. The application program configures DATA0 or DATA1 through the SD0PID register bit. 0: DATA0; 1: DATA1.
15	USBAEP	<b>USBAEP:</b> USB active endpoint Indicates whether this endpoint is an active endpoint in the current configuration. The controller clears this bit for all endpoints (except endpoint 0) when it detects a USB reset, and the program must set this bit as necessary after receiving the SetConfiguration and SetInterface commands.
14:11	Reserved	Reserved
10:0	MPSIZ	<b>MPSIZ:</b> Maximum packet size The application program configures the maximum packet length value in bytes for this endpoint with this bit.

**OTG\_FS device endpoint x interrupt register (OTG\_FS\_DIEPINTx) (where x is the endpoint number, x=0..3)**

Offset address: 0x908 + (endpoint number x 0x20)

Reset value: 0x0000 0080

This register indicates the status of USB and AHB related events at the corresponding endpoints. Refer to Figure 280 for details. When the IN endpoint interrupt bit of the controller interrupt register (IEPINT bit of the OTG\_FS\_GINTSTS register) is '1', the program must first read the device all endpoint interrupt register (OTG\_FS\_DAINTE) to obtain the endpoint number where the event occurred, and then read the interrupt registers of the corresponding endpoint to obtain detailed information. The application program must clear this bit to clear the corresponding bits of the OTG\_FS\_DAINTE and OTG\_FS\_GINTSTS registers.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved																		NAK	Reserved	PKTDRPSTS					TXFE	INEPNE	INEPNM	ITTXFE	TOC	Reserved	EPDISD	XFRC
																		rc_w1		rc_w1				r	rc_w1		rc_w1		rc_w1		rc_w1	rc_w1

Bit	notation	clarification
31:14	Reserved	Reserved
13	NAK	<b>NAK:</b> NAK Input The core generates this interrupt when the device transmits or receives a NAK. For isochronous IN endpoints, an interrupt is generated when a zero-length packet is transmitted because there is no data in the Tx FIFO.
12	Reserved	Reserved
11	PKTDRPSTS	<b>PKTDRPSTS:</b> Packet Loss Status This bit indicates to the application program that the ISOCOUT packet has been dropped. This bit has no mask bit associated with it and does not generate an interrupt.
10:8	Reserved	Reserved
7	TXFE	<b>TXFE:</b> Send FIFO empty (Transmit FIFO empty) This interrupt is generated when the transmit FIFO corresponding to this endpoint is empty or half empty. The Tx FIFO empty level bit of the controller's AHB configuration register (TXFELVL bit of the OTG_FS_GAHBCFG register) will determine whether the interrupt is generated when the transmit FIFO is in the empty or half empty state.
6	INEPNE	<b>INEPNE:</b> IN endpoint NAK effective

		Writing the CNAK bit of the OTG_FS_DIEPCTLx register will clear the NAK state of the IN endpoint, as well as clearing this bit. This interrupt means that the controller has detected that NAK has been set (either by the application program or by the controller) and this interrupt indicates that the NAK bit set by the application program has taken effect. This interrupt does not guarantee that the NAK signal has been sent on the USB line. the STALL bit has a higher priority than the NAK bit.
5	Reserved	Reserved
4	ITTXFE	ITTXFE: IN token received when TxFIFO is empty is valid only for non-periodic IN endpoints. Indicates that an IN command was received when the transmit FIFO (periodic/non-periodic) associated with this endpoint was empty. This interrupt is generated only at the endpoint where the IN command was received.
3	TOC	TOC: Timeout (Timeout condition) is only valid for controlling IN endpoints. Indicates that the controller has monitored a timeout condition since the last IN command was received by this endpoint.
2	Reserved	Reserved
1	EPDISD	EPDISD: Endpoint disabled interrupt This interrupt indicates that the endpoint has been disabled at the request of the application program.
0	XFRC	XFRC: Transfer completed interrupt This interrupt indicates that a configured transfer on this endpoint, either on the AHB side or the USB side, has been transmitted.

### OTG\_FS device endpoint x interrupt register (OTG\_FS\_DOEPINTx) (where x is the

endpoint number, x=0..3)

Offset address: 0xB08 + (endpoint number x 0x20)

Reset value: 0x0000 0080

This register indicates the status of USB and AHB related events at the corresponding endpoints. Refer to Figure 280 for details. When the OUT endpoint interrupt bit of the controller interrupt register (OEPINT bit of the OTG\_FS\_GINTSTS register) is '1', the application program must first read the device all endpoint interrupt register (OTG\_FS\_DAIN) to obtain the endpoint number where the event occurred, and then read the interrupt registers of the corresponding endpoint to obtain detailed information. The application program must clear the corresponding bits of the OTG\_FS\_DAIN and OTG\_FS\_GINTSTS registers by clearing this bit.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																		NAK	BERR	Reserved			OUTPKTER	Reserved	STSPHSRX	OTEPDIS	STUP	Reserved	EPDISD	XFRC	
																		rc_w1	rc_w1				rc_w1			rc_w1	rc_w1	rc_w1		rc_w1	rc_w1

Bit	notation	instructions
31:13	Reserved	Reserved
13	NAK	NAK: NAK Input The core generates this interrupt when the device transmits or receives a NAK. For isochronous IN endpoints, an interrupt is generated when a zero-length packet is transmitted because there is no data in the Tx FIFO.
12	BERR	BERR: Burble Error Interrupt The core generates this interrupt when a murmur is received for the termination point.
11:9	Reserved	Reserved
8	OUTPKTERR	OUTPKTERR: OUT packet error The core asserts this interrupt when it detects an output packet overflow or CRC error. This interrupt is only valid when the value is enabled.
7:6	Reserved	Reserved
5	STSPHSRX	STSPHSRX: Control write status phase received This interrupt is generated only after the core has transferred all the data sent by the host during the data phase of the control write transfer to the system memory buffer. This interrupt indicates to the application program that the host has switched from the data phase to the status phase of the control write transfer. The application program can use this interrupt to acknowledge or pause the status phase after decoding the data phase.

4	OTEPDIS	OTEPDIS: OUT command received when endpoint disabled (OUT token received when endpoint disabled) Valid only for controlling OUT endpoints. Indicates that an OUT command was received when the endpoint was disabled; this interrupt is generated only on the endpoint that received the OUT command.
3	STUP	STUP: SETUP phase done (SETUP phase done) is valid only for the control OUT endpoint. Indicates that the SETUP phase associated with this control endpoint has been completed and that there will be no more consecutive SETUP packets for the current transmission. After generating this interrupt, the application program can begin processing incoming SETUP packets.
2	Reserved	Reserved
1	EPDISD	EPDISD: Endpoint disabled interrupt This interrupt indicates that the endpoint has been disabled at the request of the application program.
0	XFRC	XFRC: Transfer completed interrupt This interrupt indicates that the configured transmission on this endpoint, either on the AHB side or the USB side, has been transmitted.

### OTG\_FS device IN endpoint 0 transmission length register (OTG\_FS\_DIEPTSIZ0)

Offset address: 0x910

Reset value: 0x0000 0000

The application program must configure this register before enabling endpoint 0. Once Endpoint 0 is enabled via the Endpoint Enable bit of the Device Control Endpoint 0 Control Register (EPENA bit of the OTG\_FS\_DIEPCTL0 register), only the controller can modify this register.

The application program can only read this register until the endpoint enable bit is cleared. Non-zero endpoints use the registers at endpoint 1 through endpoint 15.

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

Bit	notation	clarification
31:21	Reserved	Reserved
20:19	PKTCNT	<b>PKTCNT:</b> Packet count Indicates the number of USB packets required to transmit "transmission length" data on endpoint 0. The value of this register is automatically decremented by 1 for every packet (maximum length packets and short packets) that the controller reads from the transmit FIFO.
18:7	Reserved	Reserved
6:0	XFRSIZ	<b>XFRSIZ:</b> Transfer size Indicates the number of bytes to be transferred on endpoint 0. When the number of bytes transferred is reduced to 0, the controller generates an interrupt and notifies the application program. The value of this register can be set to the maximum transmission length of the endpoint after each packet. The controller automatically decrements this field when writing data from storage to the sending FIFO.



**OTG\_FS device IN endpoint transfer FIFO status register (OTG\_FS\_DTXFSTSx) (where x is the endpoint number, x=0..3)**

Offset address: 0x918 + (endpoint number x 0x20)

This register is a read-only register that holds information about the remaining space in the transmit FIFO of the device's IN endpoint.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
Reserved																INEPTFSAV																		
																r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
Bit	notation															clarification																		
31:16	Reserved															Reserved																		
15:0	INEPTFSAV															<b>INEPTFSAV:</b> IN endpoint Tx FIFO space avail indicates the remaining space information of the IN endpoint's transmit FIFO, the register value is in 32-bit word. 0x0: Send FIFO full; 0x1: 1 word remaining; 0x02: 2 words remaining; 0xn: n words remaining (where 0<n<512); 0x200: 512 words remaining; Other: Reserved.																		

**OTG\_FS device endpoint x transmission length register (OTG\_FS\_DOEPTSIZx) (where x is the endpoint number, x=1..3)**

Offset address: 0xB10 + (endpoint number x 0x20)

Reset value: 0x0000 0000

The application program must configure this register before enabling the endpoint. Once the endpoint is enabled via the Endpoint Enable bit of the Device Endpoint x Control Register (EPENA bit of the OTG\_FS\_DOEPCTLx register), only the controller can modify this register. Until the controller clears the endpoint enable bit, the application program can only read this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Rese	RXDPID/S TUPCNT	PKTCNT											XFRSIZ																		
r/rw r/rw r/w																															
Bit		notation											clarification																		
31		Reserved											Reserved																		
30:29		RXDPID											RXDPID: Received data PID (Received data PID) This bit is valid only for synchronized OUT endpoints. Indicates the data PID of the last packet received: 00: DATA0; 01: DATA2; 10: DATA1; 11: MDATA. STUPCNT: SETUPpacketcount This bit is valid only for the control OUT endpoint. Indicates the number of consecutive SETUP packets received by the endpoint: 01: 1 packet; 10: 2 packets; 11: 3 packets.																		
28:19		PKTCNT											PKTCNT: Packet count indicates the number of USB packets transmitted on this endpoint. The controller automatically decrements this field each time a packet is written from the storage area into the receive FIFO.																		
18:0		XFRSIZ											XFRSIZ: Transfer size This bit indicates the number of bytes to be transferred from this endpoint. The controller generates an interrupt to notify the application when this field is 0. You can configure this field to be the maximum transmission length of the endpoint and generate an interrupt at the end of each packet. The controller automatically decrements this value each time it writes data to the storage area from the receive FIFO.																		

## 29.16.5 OTG\_FS Power and Clock Gating Register (OTG\_FS\_PCGCCTL)

Offset address: 0xE00

Reset value: 0x0000 0000

This register is valid in both device mode and host mode.

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

Bit	notation	instructions
31:5	Reserved	Reserved
4	PHYSUSP	<b>PHYSUSP:</b> PHY hangs up Indicates that the PHY is hung. This bit is set to '1' when the PHY is put into a hung state by setting the STPPCLK bit (bit 0) through the application program.
3:2	Reserved	Reserved
1	GATEHCLK	<b>GATEHCLK:</b> HCLK Gate Control (Gate HCLK) The application program can control HCLK by setting this bit to wake up the logic module when the USB hangs or the session is invalid. The application clears this bit when the USB wakes up or a new session is initiated.
0	STPPCLK	<b>STPPCLK:</b> Stop PHY clock The application can stop the PHY's clock drive by setting this bit when the USB hangs, or the session is invalid, or the device is disconnected. The application can clear this bit when the USB wakes up or a new session is initiated.

## 29.16.6 OTG\_FS Register Image

The following table gives the USBOTG register image and Reset values.

Table160 Registers of the OTG\_FS module and their Reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																										
000h	OTG_FS_GOTGCTL	Reserved												BSVLD	ASVLD	DBCT	CIDSTS	Reserved				DHNPN	HSHNPN	HNPRQ	HNGSCS	Reserved				SRQ	SRQSCS																												
	Reset value													0	0	0	1					0	0	0	0					0	0																												
004h	OTG_FS_GOTGINT	Reserved												DBCNE	ADTOCHG	HNGDET	Reserved				HNSSCHG	SRSSCHG	Reserved				SEDET	Reserved																															
	Reset value													0	0	0									0	0					0																												
008h	OTG_FS_GAHBCFG	Reserved																								PTXFELVL	TXFELVL	Reserved					GINT																										
	Reset value																									0	0						0																										
00Ch	OTG_FS_GUSBCFG	CTXPKT	FDMOD	FHMOD	Reserved										NPTXRWEN			TRDT			HNPCAP			SRPCAP	Reserved				TOTAL																														
	Reset value	0	0	0											0			0			0							0	0	0																													
010h	OTG_FS_GRSTCTL	AHBIDL	Reserved																									TXFNUM			TXFFLSH			RXFFLSH	Reserved	FCRST	HSRST	CSRST																					
	Reset value	1																										0			0	0	0	0	0	0	0	0																					

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
014h	OTG_FS_GINTSTS	WKUINT	SRQINT	DISCINT	CIDSCHG	Reserved	PTXFE	HCINT	HPRTINT	Reserved	Reserved	IPXFR/INCOMPISOUT	ISOIXFR	OEPINT	IEPINT	Reserved	Reserved	EOPF	ISOODRP	ENUMDNE	USBRST	USBSUSP	ESUSP	Reserved	Reserved	BOUTNAKEFF	GINAKEFF	NPTXFE	RXFLVL	SOF	OTGINT	MMIS	CMOD			
	Reset value	0	0	0	0		1	0	0			0	0	0	0			0	0	0	0	0	0			0	0	1	0	0	0	0	0			
018h	OTG_FS_GINTMSK	WUIM	SRQIM	DISCINT	CIDSCHGM	Reserved	PTXFEM	HCIM	PRTIM	Reserved	Reserved	IPXFRM/IISOXFRM	IISOIXFRM	OEPINT	IEPINT	EPIMSM	Reserved	EOPFM	ISOODRPM	ENUMDNEM	USBRST	USBSUSPM	ESUSPM	Reserved	Reserved	GONAKEFFM	GINAKEFFM	NPTXFEM	RXFLVLM	SOFM	OTGINT	MMISM	Reserved			
	Reset value	0	0	0	0		0	0	0			0	0	0	0	0		0	0	0	0	0	0			0	0	0	0	0	0	0	0			
01Ch	OTG_FS_GRXSTSR (host mode)	Reserved											PKTSTS			DPID		BCNT										CHNUM								
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
	OTG_FS_GRXSTSR (equipment mode)	Reserved								FRMNUM			PKTSTS			DPID		BCNT										EPNUM								
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
020h	OTG_FS_GRXSTSPR (host mode)	Reserved											PKTSTS			DPID		BCNT										CHNUM								
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	OTG_FS_GRXSTSPR (equipment mode)	Reserved								PRMNUM			PKTSTS			DPID		BCNT										EPNUM								
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
024h	OTG_FS_GRXFSIZ	Reserved																	RXFD																	
	Reset value																		0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
028h	otg_fs_gnptxfsize	NPTXFD																	NPTXFSA																	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0			
02Ch	OTG_FS_GNPTXSTS	Reserve	NPTXQTOP							NPTQXSAV							NPTXFSAV																			
	0		0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0				
038h	OTG_FS_GCCFG	Reserved													SOFOUTEN	VBUSBSEN	VBUSASEN	Reserved	PWRDWN	Reserved																
	Reset value														0	0	0		0																	
03Ch	OTG_FS_CID	PRODUCT_ID																																		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0			
100h	otg_fs_hptxfsize	PTXFSIZ																	PTXSA																	
	Reset value	0	0	0	0	0	1	1	1	0	1	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	1	0	0			
104h	OTG_FS_DIEPTXF1	INEPTXFD																	INEPTXSA																	
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0			
108h	OTG_FS_DIEPTXF2	INEPTXFD																	INEPTXSA																	
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0			
10Ch	OTG_FS_DIEPTXF3	INEPTXFD																	INEPTXSA																	
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0			
110h	OTG_FS_DIEPTXF4	INEPTXFD																	INEPTXSA																	
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0			



Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
400h	OTG_FS_HCFG	Reserved																										FSLSS			FSLSPCS				
	Reset value																											0	0	0					
404h	OTG_FS_HFIR	Reserved															FRIVL																		
	Reset value																1	1	1	0	1	0	1	0	0	1	1	0	0	0	0	0	0		
408h	OTG_FS_HFNUM	FTREM															FRNUM																		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1			
410h	OTG_FS_HPTXSTS	PTXQTOP							PTXQSAV							PTXFSAVL																			
	Reset value	0	0	0	0	0	0	0	0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y			
414h	OTG_FS_HAINT	Reserved															HAINT																		
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
418h	OTG_FS_HAINTMSK	Reserved															HAINTM																		
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
440h	OTG_FS_HPRT	Reserved													PSPD		PTCTL			PPWR		PLSTS		Reserved	PRST	PSUSP	PRES	POCCHNG	POCA	PENCHNG	PENA	PCDET	PCSTS		
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
500h	OTG_FS_HCCHAR0	CHENA	CHDIS	ODDFRM	DAD						MCNT		EPTYP		LSDEV	Reserved	EPDIR	EPNUM			MPSIZ														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
520h	OTG_FS_HCCHAR1	CHENA	CHDIS	ODDFRM	DAD						MCNT		EPTYP		LSDEV	Reserved	EPDIR	EPNUM			MPSIZ														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
540h	OTG_FS_HCCHAR2	CHENA	CHDIS	ODDFRM	DAD						MCNT		EPTYP		LSDEV	Reserved	EPDIR	EPNUM			MPSIZ														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
560h	OTG_FS_HCCHAR3	CHENA	CHDIS	ODDFRM	DAD						MCNT		EPTYP		LSDEV	Reserved	EPDIR	EPNUM			MPSIZ														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
580h	OTG_FS_HCCHAR4	CHENA	CHDIS	ODDFRM	DAD						MCNT		EPTYP		LSDEV	Reserved	EPDIR	EPNUM			MPSIZ														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
5A0h	OTG_FS_HCCHAR5	CHENA	CHDIS	ODDFRM	DAD						MCNT		EPTYP		LSDEV	Reserved	EPDIR	EPNUM			MPSIZ														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
5C0h	OTG_FS_HCCHAR6	CHENA	CHDIS	ODDFRM	DAD						MCNT		EPTYP		LSDEV	Reserved	EPDIR	EPNUM			MPSIZ														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
5E0h	OTG_FS_HCCHAR7	CHENA	CHDIS	ODDFRM	DAD						MCNT		EPTYP		LSDEV	Reserved	EPDIR	EPNUM			MPSIZ														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
508h	OTG_FS_HCINT0	Reserved																					DTERR	FRMOR	BBERR	TXERR	Reserved	ACK	NAK	STALL	Reserved	CHH	XFRC		
	Reset value																						0	0	0	0	0	0	0	0	0	0	0		

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
528h	OTG_FS_HCINT1	Reserved																					DTERR	FRMOR	BBERR	TXERR	Reserved	ACK	NAK	STALL	Reserved	CHH	XFRC
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
548h	OTG_FS_HCINT2	Reserved																					DTERR	FRMOR	BBERR	TXERR	Reserved	ACK	NAK	STALL	Reserved	CHH	XFRC
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
568h	OTG_FS_HCINT3	Reserved																					DTERR	FRMOR	BBERR	TXERR	Reserved	ACK	NAK	STALL	Reserved	CHH	XFRC
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
588h	OTG_FS_HCINT4	Reserved																					DTERR	FRMOR	BBERR	TXERR	Reserved	ACK	NAK	STALL	Reserved	CHH	XFRC
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
5A8h	OTG_FS_HCINT5	Reserved																					DTERR	FRMOR	BBERR	TXERR	Reserved	ACK	NAK	STALL	Reserved	CHH	XFRC
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
5C8h	OTG_FS_HCINT6	Reserved																					DTERR	FRMOR	BBERR	TXERR	Reserved	ACK	NAK	STALL	Reserved	CHH	XFRC
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
5E8h	OTG_FS_HCINT7	Reserved																					DTERR	FRMOR	BBERR	TXERR	Reserved	ACK	NAK	STALL	Reserved	CHH	XFRC
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
50Ch	otg_fs_hcintmsk0	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	Reserved	ACKM	NAKM	STALLM	Reserved	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
52Ch	otg_fs_hcintmsk1	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	Reserved	ACKM	NAKM	STALLM	Reserved	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
54Ch	otg_fs_hcintmsk2	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	Reserved	ACKM	NAKM	STALLM	Reserved	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
56Ch	OTG_FS_HCINTMSK3	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	Reserved	ACKM	NAKM	STALLM	Reserved	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
58Ch	otg_fs_hcintmsk4	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	Reserved	ACKM	NAKM	STALLM	Reserved	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
5ACh	otg_fs_hcintmsk5	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	Reserved	ACKM	NAKM	STALLM	Reserved	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
5CCh	otg_fs_hcintmsk6	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	Reserved	ACKM	NAKM	STALLM	Reserved	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
5ECh	otg_fs_hcintmsk7	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	Reserved	ACKM	NAKM	STALLM	Reserved	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
510h	OTG_FS_HCTSIZ0	Res	DPID	PKTCNT										XFRSIZ																			

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
530h	OTG_FS_HCTSIZ1	Reserved	DPID		PKTCNT										XFRSIZ																				
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
550h	OTG_FS_HCTSIZ2	Reserved	DPID		PKTCNT										XFRSIZ																				
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
570h	OTG_FS_HCTSIZ3	Reserved	DPID		PKTCNT										XFRSIZ																				
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
590h	OTG_FS_HCTSIZ4	Reserved	DPID		PKTCNT										XFRSIZ																				
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
5B0h	OTG_FS_HCTSIZ5	Reserved	DPID		PKTCNT										XFRSIZ																				
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
5D0h	OTG_FS_HCTSIZ6	Reserved	DPID		PKTCNT										XFRSIZ																				
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
5F0h	OTG_FS_HCTSIZ7	Reserved	DPID		PKTCNT										XFRSIZ																				
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
800h	OTG_FS_DCFG	Reserved																			PFIVL		DAD					Reserved	NZLSOHSK	DSPD					
	Reset value																				0	0	0	0	0	0	0			0	0	0	0	0	0
804h	OTG_FS_DCTL	Reserved																			POPRGDNE	CGONAK	SGONAK	CGINAK	SGINAK	TCTL		GONSTS	GINSTS	SDIS	RWUSIG				
	Reset value																									0	0					0	0	0	0
808h	OTG_FS_DSTS	Reserved										FNSOF										Reserved					EERR	ENUMSPD	SUSPSTS						
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				0					
810h	OTG_FS_DIEPMSK	Reserved																			NAKM	Reserved					INENEM	INENMM	ITTXFEMSK	TOM	Reserved	EPDM	XFCRM		
	Reset value																					0	0	0	0	0								0	0
814h	OTG_FS_DOEPMASK	Reserved																			NAKMSK	BERRM	Reserved		OUTPKTERRM	Reserved		STSPHSRXM	OTEPDM	STUPM	Reserved	EPDM	SFCRM		
	Reset value																						0	0		0	0							0	0
818h	OTG_FS_DAIN	OEPINT										IEPINT																							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
81Ch	OTG_FS_DAINMSK	OEPM										IEPM																							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
828h	OTG_FS_DVBUSDIS	Reserved										VBUSDT																							
	Reset value											0	0	0	1	0	1	1	1	1	1	1	0	1	0	1	1	1	0	0	0	0	0	0	0
82Ch	OTG_FS_DVBUSPULSE	Reserved										DVBUSP																							
	Reset value											0	1	0	1	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
834h	otg_fs_diepempmsk	Reserved										INEPTXFEM																							
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
900h	OTG_FS_DIEPCTL0	EPENA	EPDIS	Reserved		SNAK	CNAK	TXFNUM				STALL	Reserved	EPTYP	NAKSTS		Reserved	USBAEP	Reserved										MPSIZ						
	Reset value	0	0			0	0	0	0	0	0	0	0		0	0	0	Reserved	1											0	0				
918h	OTG_FS_DTXFSTS0	Reserved																INEPTFSAV																	
	Reset value																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
920h	OTG_FS_DIEPCTL1	EPENA	EPDIS	SODDFRM	SDOPID/SEVNFRM	SNAK	CNAK	TXFNUM				STALL	Reserved	EPTYP	NAKSTS		EONUM/DPID	USBAEP	Reserved		MPSIZ														
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0		
938h	OTG_FS_DTXFSTS1	Reserved																INEPTFSAV																	
	Reset value																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
940h	OTG_FS_DIEPCTL2	EPENA	EPDIS	SODDFRM	SDOPID/SEVNFRM	SNAK	CNAK	TXFNUM				STALL	Reserved	EPTYP	NAKSTS		EONUM/DPID	USBAEP	Reserved		MPSIZ														
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0		
958h	OTG_FS_DTXFSTS2	Reserved																INEPTFSAV																	
	Reset value																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
960h	OTG_FS_DIEPCTL3	EPENA	EPDIS	SODDFRM	SDOPID/SEVNFRM	SNAK	CNAK	TXFNUM				STALL	Reserved	EPTYP	NAKSTS		EONUM/DPID	USBAEP	Reserved		MPSIZ														
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0		
978h	OTG_FS_DTXFSTS3	Reserved																INEPTFSAV																	
	Reset value																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
B00h	otg_fs_oeepctl0	EPENA	EPDIS	Reserved		SNAK	CNAK	Reserved				STALL	SNPM	EPTYP	NAKSTS		Reserved	USBAEP	Reserved										MPSIZ						
	Reset value	0	0			0	0					0	0	0	0	0	Reserved	1											0	0					
B20h	otg_fs_oeepctl1	EPENA	EPDIS	SODDFRM	SDOPID/SEVNFRM	SNAK	CNAK	Reserved				STALL	SNPM	EPTYP	NAKSTS		EONUM/DPID	USBAEP	Reserved		MPSIZ														
	Reset value	0	0	0	0	0	0					0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0		
B40h	otg_fs_oeepctl2	EPENA	EPDIS	SODDFRM	SDOPID/SEVNFRM	SNAK	CNAK	Reserved				STALL	SNPM	EPTYP	NAKSTS		EONUM/DPID	USBAEP	Reserved		MPSIZ														
	Reset value	0	0	0	0	0	0					0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0		

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																																																																																																																																																																																																																																																	
B60h	otg_fs_doeptcl3	EPENA	EPDIS	SODDFRM	SDOPID/SEVNFRM	SNAK	CNAK	Reserved				STALL	SNPM	EPTYP	NAKSTS		EONUM/DPID	USBAEP	Reserved					0	0	0																																																																																																																																																																																																																																																								
	Reset value	0	0	0	0	0	0					0	0	0			0	0																		0	0	0	0																																																																																																																																																																																																																																											
908h	OTG_FS_DIEPINT0	Reserved																					Reserved	Reserved	TXFE	INEPNE	INEPNM	ITTXFE	TOC	Reserved	EPDIS	XFRC																																																																																																																																																																																																																																																		
	Reset value																																0	Reserved	0	PKTDRPS																																																																																																																																																																																																																																														
928h	OTG_FS_DIEPINT1																																Reserved																				Reserved	Reserved	TXFE	INEPNE	INEPNM	ITTXFE	TOC	Reserved	EPDIS	XFRC																																																																																																																																																																																																																				
	Reset value																																																														0	Reserved	0	PKTDRPSTS																																																																																																																																																																																																																
948h	OTG_FS_DIEPINT2																																																														Reserved																				Reserved	Reserved	TXFE	INEPNE	INEPNM	ITTXFE	TOC	Reserved	EPDIS	XFRC																																																																																																																																																																																						
	Reset value																																																																																												0	Reserved	0	PKTDRPSTS																																																																																																																																																																																		
968h	OTG_FS_DIEPINT3																																																																																												Reserved																				Reserved	Reserved	TXFE	INEPNE	INEPNM	ITTXFE	TOC	Reserved	EPDIS	XFRC																																																																																																																																																								
	Reset value																																																																																																																										0	Reserved	0	PKTDRPSTS																																																																																																																																																				
B08h	OTG_FS_DOEPINT0																																																																																																																										Reserved																				Reserved	Reserved	OUTPKTERR	Reserved	STSPHSRX	OTEPDIS	STUP	Reserved	EPDIS	XFRC																																																																																																																										
	Reset value																																																																																																																																																								0	BERR	Reserved	0																																																																																																																						
B28h	OTG_FS_DOEPINT1																																																																																																																																																								Reserved																				Reserved	Reserved	OUTPKTERR	Reserved	STSPHSRX	OTEPDIS	STUP	Reserved	EPDIS	XFRC																																																																																												
	Reset value																																																																																																																																																																																						0	BERR	Reserved	0																																																																																								
B48h	OTG_FS_DOEPINT2																																																																																																																																																																																						Reserved																				Reserved	Reserved	OUTPKTERR	Reserved	STSPHSRX	OTEPDIS	STUP	Reserved	EPDIS	XFRC																																																														
	Reset value																																																																																																																																																																																																																				0	BERR	Reserved	0																																																										
B68h	OTG_FS_DOEPINT3																																																																																																																																																																																																																				Reserved																				Reserved	Reserved	OUTPKTERR	Reserved	STSPHSRX	OTEPDIS	STUP	Reserved	EPDIS	XFRC																																
	Reset value																																																																																																																																																																																																																																																		0	BERR	Reserved	0																												
910h	OTG_FS_DIEPTSIZ0																																																																																																																																																																																																																																																		Reserved											PKT CNT	0	0																		
	Reset value																																																																																																																																																																																																																																																																																	
930h	OTG_FS_DIEPTSIZ1																																																																																																																																																																																																																																																		Reserved	PKTCN												XFRSIZ																		
	Reset value																																																																																																																																																																																																																																																			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
950h	OTG_FS_DIEPTSZ2	Reserved		PKTCNT										XFRSZ																			
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
970h	OTG_FS_DIEPTSZ3	Reserved		PKTCNT										XFRSZ																			
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B10h	OTG_FS_DOEPTSZ0	Reserved	STUPCNT	Reserved										PKTCNT	Reserved										XFRSZ								
	Reset value			0	0											0											0	0	0	0	0	0	0
B30h	OTG_FS_DOEPTSZ1	Reserved	RXDPID/ STUPCNT	PKTCNT										XFRSZ																			
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B50h	OTG_FS_DOEPTSZ2	Reserved	RXDPID/ STUPCNT	PKTCNT										XFRSZ																			
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B70h	OTG_FS_DOEPTSZ3	Reserved	RXDPID/ STUPCNT	PKTCNT										XFRSZ																			
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E00h	otg_fs_pgcctl	Reserved																										PHYSUSP	Reserved	GATECLK	STPPCLK		
	Reset value																																

Refer to Table 1 for the base addresses of these registers.

---

## 29.17 OTG\_FS Programming Rules

### 29.17.1 Controller Initialization

The application must initialize the controller in sequence. If the USB cable is connected during power-up, the current operating mode bit of the controller interrupt register (CMOD bit of OTG\_FS\_GINTSTS) will indicate the current operating mode. The OTG\_FS controller operates in host mode when a Class A plug is inserted and in device mode when a Class B plug is inserted. This section describes the initialization of the OTG\_FS controller after power-up. Whether working in host mode or in device mode, the application must initialize the controller in order. All controller global registers must be initialized according to the controller configuration:

1. Configure the following bits of the AHB global configuration register (OTG\_FS\_GAHBCFG):
  - Global interrupt mask bit GINT 1 =
  - Receive FIFO non-null bit (RXFLVL bit of the OTG\_FS\_GINTSTS register)
  - Periodic sending of FIFO null levels
2. Configure the following bits of the OTG\_FS\_GUSBCFG register:
  - HNP enable bit
  - SRP Enable Bit
  - FS timeout calibration bit
  - USB reflection time bit
3. The application program cannot mask the following bits of the GINTMSK register:
  - OTG Interrupt Mask Bit
  - Mode mismatch interrupt mask bit
4. The application program determines whether the current OTG\_FS controller is in host mode or device mode by reading the CMOD bit of the OTG\_FS\_GINTSTS register.

### 29.17.2 Initialization in Host Mode

In host mode, the application needs to follow the following steps to configure the controller:

1. Configure the HPRTINT bit of the GINTMSK register to '1' to enable this interrupt.
2. Configure the OTG\_FS\_HCFG register to select to operate in host mode at full speed.
3. Configure the PPWR bit of the OTG\_FS\_HPRT register to '1' to enable VBUS power on the USB cable.
4. Wait for the PCDET interrupt from the OTG\_FS\_HPRT0 register, which indicates that a USB device is connected to the port.
5. Configure the PRST bit of the OTG\_FS\_HPRT register to '1' to initiate a reset operation on the device.
6. Wait a minimum of 10ms to ensure that the reset operation is complete.
7. Configure the PRST bit of the OTG\_FS\_HPRT register to '0'.
8. Wait for the PENCHNG bit of the OTG\_FS\_HPRT register to interrupt.
9. Reads the PSPD bit of the OTG\_FS\_HPRT register, which indicates whether the device uses full or low speed for enumeration.
10. Configure the HFIR registers according to the PHY clock 1 that has been selected.
11. Configure the OTG\_FS\_RXFSIZE register to select the length of the receive FIFO.
12. Configure the OTG\_FS\_NPTXFSIZE register to select the length and start address of the non-periodic transmit FIFO.
13. Configure the OTG\_FS\_HPTXFSIZ register to select the length and start address of the periodic transmit FIFO. In order to communicate with the device, the application must enable and initialize at least one channel.

### 29.17.3 Initialization in Device Mode

The application must follow the steps below to initialize the controller to operate in device mode at power-up or when switching from host mode to device mode:

1. Configure the following bits of the OTG\_FS\_DCFG register:
  - Equipment speed
  - Response status to non-zero length OUT packets
2. Configure the OTG\_FS\_GINTMSK register to enable the following interrupts:
  - USB Reset
  - Enumeration completion
  - USB hangs early

- USB hang
  - SOF
3. In Class B device mode, configure the VBUSBSEN bit of the OTG\_FS\_GCCFG register to enable VBUS to pull up the DP line to 5V.
  4. Wait for the USBRST bit in the OTG\_FS\_GINTSTS register to indicate that a reset signal lasting approximately 10ms has been detected on the USB line.
- Wait for the ENUMDNE bit in the OTG\_FS\_GINTSTS register, which indicates the end of the USB reset operation. After receiving this interrupt, the application program must read the OTG\_FS\_DSTS register to get the enumeration speed information and initialize it according to the section "Endpoint Initialization after Enumeration Completion".
- At this point, the device is ready to receive SOF packets and control transmission is achieved via endpoint 0.

## 29.17.4 Programming Rules in Host Mode

### Channel Initialization

Before the host can communicate with a connected device, the application needs to initialize one or more channels. Channels can be initialized and enabled by following these steps:

1. Configure the GINTMSK register to enable the following interrupts:
2. channel disruption
  - Non-periodic send FIFO null for OUT transfers (applies to slave mode, i.e., configured to run more than 1 packet on the pipeline at the transfer level)
  - Non-periodic send FIFO half-empty for OUT transfers (applies to slave mode, i.e., configured to run more than 1 packet on the pipeline at the transfer level)
3. Configure the OTG\_FS\_HAINTMSK register to enable interrupts for the selected channel.
4. Configure the OTG\_FS\_HCINTMSK register of the selected channel to enable transmission-related interrupts in the host channel interrupt register.
5. Configure the OTG\_FS\_HCTSIZx register of the selected channel to set the total transmission length in bytes, and the number of packets expected to be received, including short packets. The PID bit of the register needs to be set according to the initial data PID number (which will be used for the data PID number of the first transmitted OUT packet and the data PID number of the expected first received IN packet).
6. Configure the OTG\_FS\_HCCHARx register of the selected channel to set the characteristics of the device endpoint, such as transmission type, speed, direction, etc. (The channel enable bit needs to be set to '1' to enable the channel only when the application is ready to transmit or receive packets.)

### Abortive Channel

The application program can abort either channel by setting the CHDIS and CHENA bits of the OTG\_FS\_HCCHARx register to '1'. This action will cause the OTG\_FS host controller to clear the submitted transfer request (if present) and generate a channel abort interrupt. The application program needs to wait for the CHH bit of the OTG\_FS\_HCINTx register to be '1' (indicating that this channel has been aborted) before reassigning this channel for other communications. The OTG\_FS host controller cannot interrupt a transfer that has already begun on the USB bus. Before aborting a channel, the application program needs to confirm that there is at least one remaining space in the non-periodic request queue (when the abort is for a non-periodic channel), or the periodic request queue (when the abort is for a periodic channel). When the request queue is full (before performing a channel abort operation), a submitted transfer request can be cleared by writing the CHDIS bit of the OTG\_FS\_HCCHARx register to '1' and waiting for the CHENA bit to become '0'.

The application needs to abort the channel in the following cases:

1. An XFRC bit of '1' in the OTG\_FS\_HCINTx register is detected during a non-periodic IN transfer or a high-bandwidth interrupt IN transfer (in slave mode only).
2. A STALL, TXERR, BBERR, or DTERR event is generated in the OTG\_FS\_HCINTx register for any IN or OUT channel (in slave mode only). For high-bandwidth interrupt IN transfers (in slave mode only), once the application program detects a DTERR event, it must abort the channel and wait for an event that the channel has been successfully aborted. The application program needs to be able to receive other events (DTERR, NAK, DATA, TXERR) for the channel before receiving the message that the channel has been successfully aborted.



3. When the DISCINT of the OTG\_FS\_GINTSTS register is '1' (indicating that the device is disconnected), the application needs to abort all enabled channels.
4. When an application needs to abort a transmission before it ends properly.

### Operating Mode

The application needs to initialize a channel before communicating with the connected device. This section describes the operations performed for the different USB transfer types.

### Write to Transmit FIFO

The OTG\_FS Host Mode Controller automatically writes a request to the periodic/non-periodic request queue (OUT request) when the program writes a packet in DWORD, so it is important to make sure that there is at least one free position in the periodic/non-periodic request queue before writing to the send FIFO. The application program can only write to the send FIFO in DWORD, if the packet to be written is not DWORD aligned, the remaining bytes need to be made up. The OTG\_FS Host Mode Controller determines the length of the packet that will actually be sent in accordance with the configured maximum packet length and the actual transmission length.

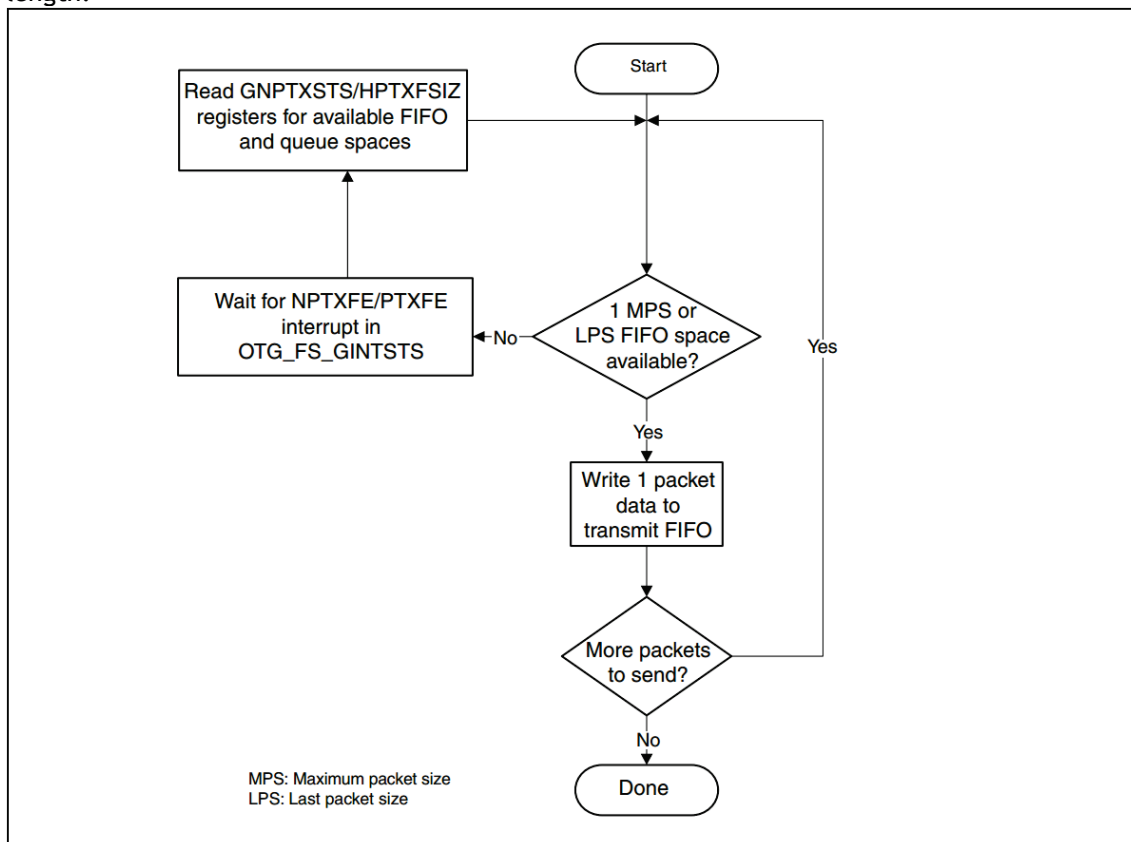


Figure282 Send FIFO Write Task

## Read Receive FIFO

The application needs to read the receive FIFO only when it receives an IN packet (0x0010).

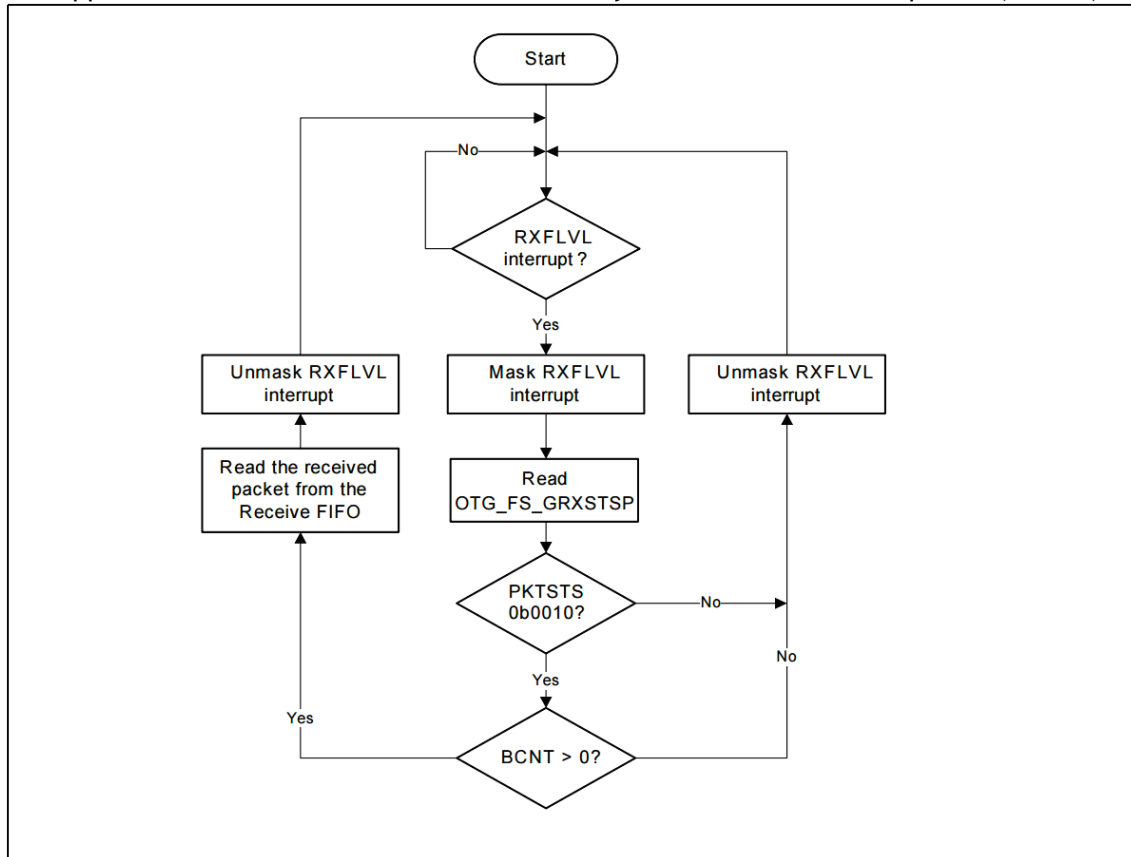


Figure283 Receive FIFO readout task

## OUT/SETUP for block transfer and control transfer

The following figure gives the flow of operation of a typical OUT/SETUP for a block transfer or control transfer. See channel 1 (ch\_1) for details. 2 OUT packets for block transfers need to be sent and 1 SETUP packet transfer for control transfers needs to be processed.

Hypothesis:

- The application needs to send 2 packets of maximum packet length (transmission length of 1024 bytes)
- The non-periodic transmit FIFO has saved 2 packets (128K bytes for full speed transmission)
- Off-cycle request queue depth is 4

## OUT/SETUP Processing Flow for Common Block Transfers and Control Transfers

The processing flow depicted in the figure below (using channel 1) is as follows:

- a) Initialize channel 1.
- b) Write the first packet of channel 1.
- c) With the last DWORD data written, the controller writes a request to the non-periodic request queue.
- d) When the non-periodic request queue is not empty, the controller sends an OUT token within the current frame.
- e) Write the second packet (last) to channel 1.
- f) The controller generates an XFRC interrupt at the normal end of the previous transmission.
- g) An XFRC event is received and the channel is rearranged to serve other transmissions.
- h) Controls non-ACK responses.

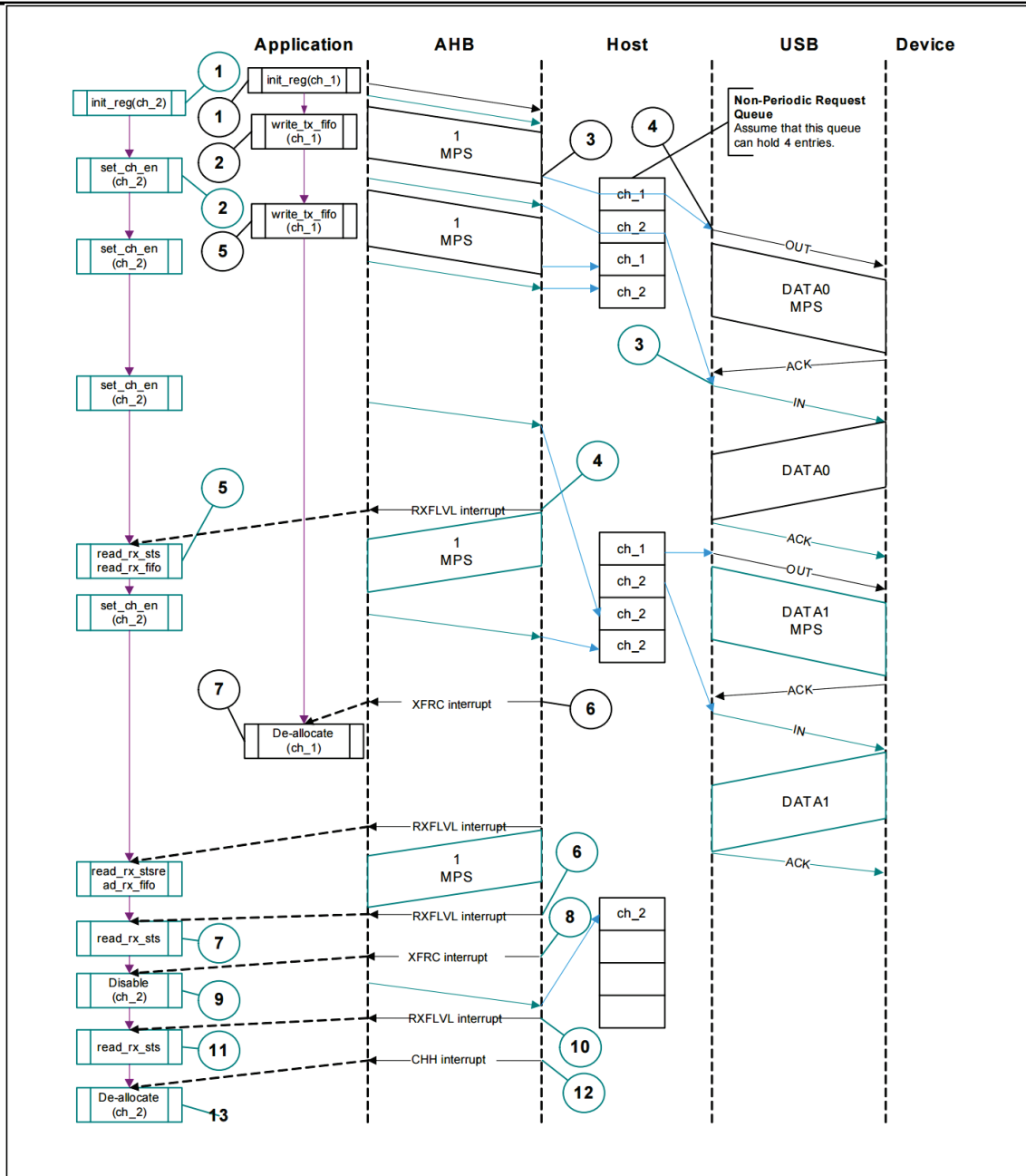


Figure 284 Normal block/control OUT/SETUP and block/control IN transfer process

The following interrupt handler code routines associated with the channel are listed for the OUT/SETUP process for block transfers and control transfers in slave mode.

### Interrupt Handling Flow for Block and Control OUT/SETUP Transmission and Block and

#### Control IN Transmission

```

1. Block and control OUT/SETUP transfers
2. Enable (NAK/TXERR/STALL/XFRC) interrupts
3. if(XFRC)
4. {
5.   Clear Error Count
6.   Mask ACK interrupt
7.   Unassigning channels
8. }
9. else if(STALL)
10. {
11.   Transmission complete = 1
12.   Enable CHH interrupt
13.   abortive channel
14. }

```

```

15. else if(NAK or TXERR)
16. {
17.     Reset buffer pointer
18.     Enable CHH interrupt
19.     abortive channel
20.     if(TXERR)
21.     {
22.         Increase error count
23.         Enable ACK interrupt
24.     }
25. else
26. {
27.     Clear Error Count
28. }
29. }
30. else if(CHH)
31. {
32.     Mask CHH interrupt
33.     if(transfer complete or (error count == 3))
34.     {
35.         Unassigning channels
36.     }
37. else
38. {
39.     Reinitialize channels
40. }
41. }
42. else if(ACK)
43. {
44.     Clear Error Count Mask ACK Interrupt
45. }
46.

```

The application program must have free space in the sending FIFO and the request queue before it can write packets to the sending FIFO. The availability of free space in the send FIFO can be detected by using the NPTXFE bit in the OTG\_FS\_GINTSTS register.

```

1. IN transfer of blocks and controls
2. Enable (TXERR/XFRC/BBERR/STALL/DTERR) Interrupt
3. if(XFRC)
4. {
5.     Reset Error Count
6.     Enable CHH interrupt
7.     abortive channel
8.     Shielded ACK
9. }
10. else if(TXERR or BBERR or STALL)
11. {
12.     Enable CHH interrupt
13.     abortive channel
14.     if(TXERR)
15.     {
16.         Increase error count
17.         Enable ACK interrupt
18.     }
19. }
20. else if(CHH)
21. {
22.     Shield CHH
23.     if(transfer complete or (error count == 3))
24.     {
25.         Redistribution of channels
26.     }
27. else
28. {
29.         Reinitialize channels
30.     }
31. }
32. else if(ACK)
33. {
34.     Reset Error Counter Mask ACK
35. }

```

```

36. else if(DTERR)
37. {
38.     Reset Error Counter
39. }
40.

```

The application must wait until an XFRC event occurs before it can write a request if there is space left in the request queue.

### IN Transfer of Blocks/Controls

The operational flow of a typical block (BULK) and control IN transfer is shown below; see channel 2. make the following assumptions:

- The application needs to receive 2 packets of maximum packet length (transmission length of 1024 bytes).
- The receive FIFO can hold at least 1 packet of maximum packet length and 2 DWORD type statuses specific to each packet. (72 bytes for full speed communication).
- The depth of the non-periodic request queue is 4.

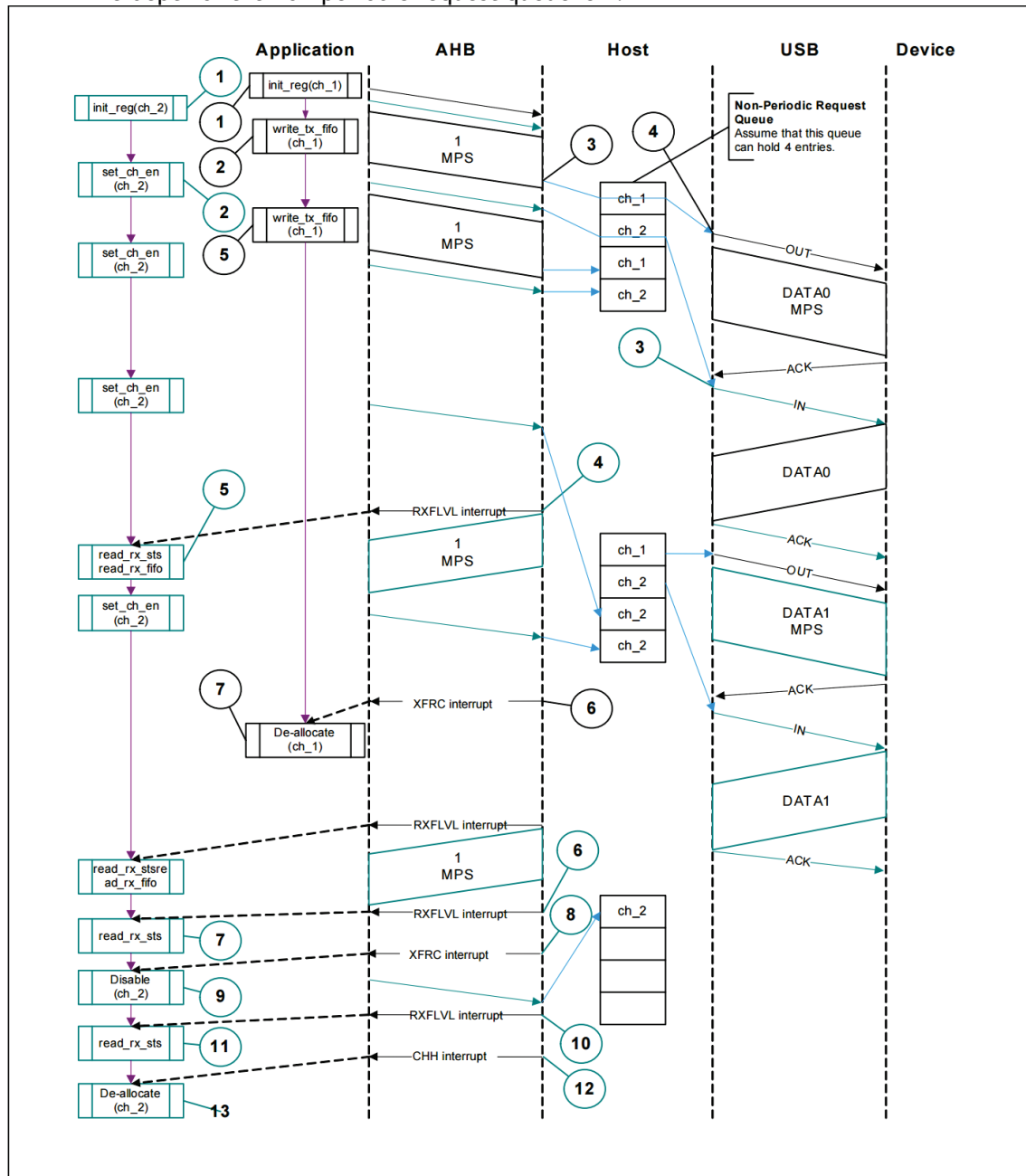


Figure 285 IN transfer process of Bulk/Control

---

The operation process is as follows:

- a) Initialize Channel 2
- b) Set the CHENA bit of the HCCHAR2 register to write an IN request to the non-periodic request queue.
- c) The controller attempts to send an IN token after completing the current OUT transmission.
- d) After the received data is written to the receive FIFO, the controller generates the RXFLVL interrupt.
- e) In the RXFLVL interrupt handler, it is necessary to mask the RXFLVL interrupt, read the received status to determine the number of bytes received, and read the corresponding data from the receive FIFO. Turn on the RXFLVL interrupt after completing the above operations.
- f) The controller generates the RXFLVL interrupt after the transmission completion status is deposited into the receive FIFO.
- g) The application reads the status of the received packet and ignores it if it is not an IN packet (i.e., the PKTSTS bit of the GRXSTSR register  $\neq$  0x0010).
- h) The controller generates an XFRC interrupt after the status of the received packet has been read.
- i) In the XFRC interrupt handler, the OTG\_FS\_HCCHAR2 register needs to be configured to abort the channel and stop writing more requests. The controller will write a channel abort request to the acyclic request queue after the OTG\_FS\_HCCHAR2 register has been set.
- j) The controller will generate the RXFLVL interrupt after the abort status is written to the receive FIFO.
- k) Reads the packet state, but does not process it.
- l) The controller will generate a CHH interrupt after the abort message is read from the receive FIFO.
- m) In the CHH interrupt handler, the application program can reassign the channel to serve other transmissions.
- n) Controls non-ACK responses.

### Control Transmission in Slave Mode

Setup, data, and status are the 3 phases of the control transfer and must be treated as 3 separate transfers. The Setup, data OUT, and status OUT transfers are handled similarly to the block OUT transfer described earlier. The Data IN and Status IN transfers are handled similarly to the Block IN transfer described earlier. In all 3 phases, the application needs to set the EPTYP bit of the OTG\_FS\_HCCHAR1 register to indicate that the transfer type is a control transfer. In the Setup phase, the application needs to set the DPID bit of the OTG\_FS\_HCTSIZ1 register, indicating a SETUP packet.

### Interrupt OUT Transmission

The flow of operations for a typical interrupt OUT transfer is shown below. Make the following assumptions:

- The application sends a maximum packet length packet in each frame starting with the odd numbered frames (send length of 1024 bytes).
- 1 packet (1024 bytes) can be stored in the periodic FIFO.
- The periodic request queue depth is 4. the operation flow is as follows:
  - a) Initialize channel 1 while the application needs to configure the ODDFRM bit of the OTG\_FS\_HCCHAR1 register to indicate to start transmitting from odd frames.
  - b) Writes the first packet to be transmitted to channel 1. for high-bandwidth interrupt transmissions. the application needs to write subsequent packets to be transmitted to the channel before switching to another channel. the number of packets is specified by the MCNT bit) the number of packets that need to be transmitted at the next frame.
  - c) At the end of writing the last DWORD of each packet, the controller writes the request to the periodic request queue.
  - d) On the next odd-numbered frame, the controller sends an OUT token.
  - e) The controller generates an XFRC interrupt after the last packet has been transmitted normally.
  - f) In the XFRC interrupt handler, the application can reinitialize the channel so that the channel is available for other transmissions.

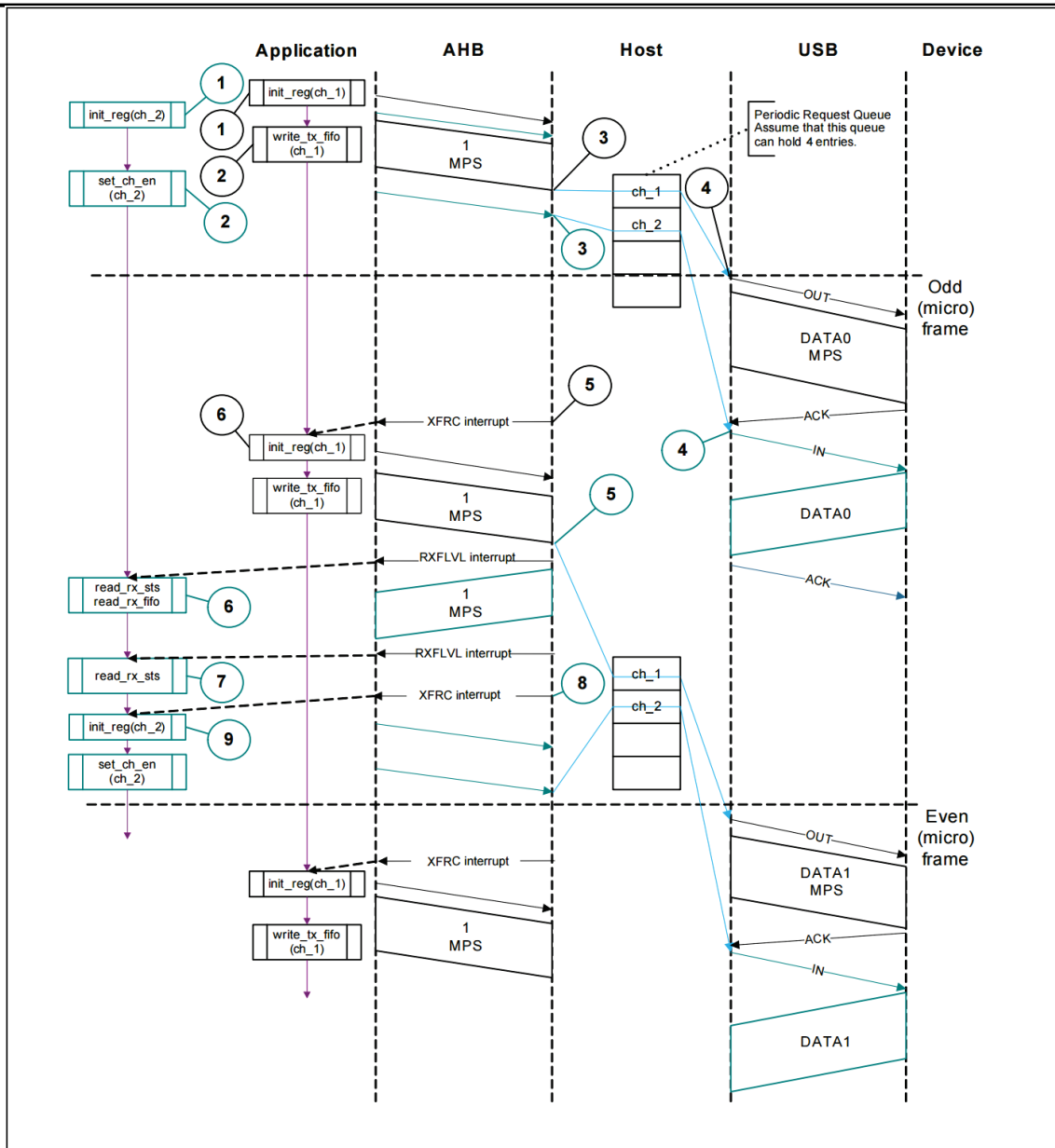


Figure 286 Normal Interrupt OUT/IN transmission process

#### Interrupt service program for interrupt OUT/IN transfers

```

1. Interrupt OUT
2. Enable (NAK/TXERR/STALL/XFRC/FRMOR) Interrupt
3. if(XFRC)
4. {
5.   Reset Error Count
6.   Mask ACK interrupt
7.   Redistribution of channels
8. }
9. else if(STALL or FRMOR)
10. {
11.   Mask ACK interrupt
12.   Enable CHH interrupt
13.   abortive channel
14.   if(STALL)
15.   {
16.     Transmission complete = 1
17.   }
18. }
19. else if(NAK or TXERR)
20. {
21.   Reset the buffer pointer
22.   Reset Error Count

```

```

23.  Mask ACK interrupt
24.  Enable CHH interrupt
25.  abortive channel
26. }
27. else if(CHH)
28. {
29.  Mask CHH interrupt
30.  if(transfer complete or (error count == 3))
31.  {
32.      Redistribution of channels
33.  }
34.  else
35.  {
36.      Reinitialize the channel (for the next b_interval-1 frame)
37.  }
38. }
39. else if(ACK)
40. {
41.  Reset Error Count Mask ACK Interrupt
42. }
43.

```

The application program needs to write packets and requests to the transmit FIFO and request queue according to the number of packets specified by the MCNT bit before switching to another channel, at this time the transmit FIFO must have space left, you can get the information whether the transmit FIFO has space left or not by using the PTXFE bit of the OTG\_FS\_GINTSTS register.

```

1.  Interrupt IN
2.  Enable (NAK/TXERR/XFRC/BBERR/STALL/FRMOR/DTERR)
3.  if(XFRC)
4.  {
5.      Reset Error Count
6.      Shielded ACK
7.      if(OTG_FS_HCTSIZx.PKTCNT==0)
8.      {
9.          Redistribution of channels
10.     }
11.     else
12.     {
13.         Transmission complete = 1
14.         Enable CHH abort channel
15.     }
16. }
17. else if(STALL or FRMOR or NAK or DTERR or BBERR)
18. {
19.     Mask ACK Enable
20.     CHH
21.     abortive channel
22.     if(STALL or BBERR)
23.     {
24.         Reset Error Counter
25.         Transmission complete = 1
26.     }
27.     else if(!FRMOR)
28.     {
29.         Reset Error Count
30.     }
31. }
32. else if(TXERR)
33. {
34.     Increase error count
35.     Enable ACK
36.     Enable CHH
37.     abortive channel
38. }
39. else if(CHH)
40. {
41.     Shield CHH
42.     if(transfer complete or (error count == 3))
43.     {
44.         Redistribution of channels

```



```
45. }
46. else
47. {
48.     Reinitialize the channel (for the next b_interval-1 frame)
49. }
50. }
51. else if(ACK)
52. {
53.     Reset Error Count Mask ACK
54. }
55.
```

The application program needs to write requests to the request queue with space remaining until the number of requests written reaches the number specified in the MCNT bit before switching to another channel.

### Interrupts IN Transmission

Hypothesis:

- The program needs to receive one packet of maximum packet length per frame starting from odd numbered frames (transmission length is 1024 bytes)
- The receive FIFO can hold packets indicating 1 maximum packet length and 2 status words of type DWORD (1031 bytes in total)
- Periodic request queue depth of 4

### Normal Interrupt IN Operation

The operation process is as follows:

- a) To initialize channel 2, the program needs to write the ODDFRM bit of the OTG\_FS\_HCCHAR2 register to specify the odd frame.
- b) Setting the CHENA bit of the OTG\_FS\_HCCHAR2 register causes the controller to write an IN request to the periodic request queue. For high-bandwidth interrupt transmissions, the program needs to write the MCNT bit of the OTG\_FS\_HCCHAR2 register (which specifies the number of packets to be received during the next frame) before switching to another channel.
- c) Each time the program sets the OTG\_FS\_HCCHAR2 register CHENA bit, the controller writes an IN request to the periodic request queue.
- d) On the next odd-numbered frame, the controller sends an IN token.
- e) The RXFLVL interrupt is generated when the controller receives a packet from IN and writes it to RXFIFO.
- f) In the RXFLVL interrupt handler, the application program reads the status of the received packet, learns the number of bytes received, and reads the receive FIFO accordingly. Before reading the receive FIFO, it is necessary to mask the RXFLVL interrupt and enable the RXFLVL interrupt after reading the complete packet.
- g) The controller generates the RXFLVL interrupt after the transmission completion status is deposited into the receive FIFO. The program needs to read this packet status and discard the packet when the status indicates a non-IN packet (PKTSTS bit of the GRXSTSR register  $\neq$  0x0010).
- h) After reading the received packet status, the controller generates an XFRC interrupt.
- i) In the XFRC interrupt handler, the application program reads the PKTCNT bit of the OTG\_FS\_HCTSIZ2 register and, if the bit is not 0, aborts the channel and then reinitializes the channel in preparation for the next transmission. If the PKTCNT bit is 0, reinitialize the channel and prepare for the next transmission. At this point the application needs to reset the ODDFRM bit of the OTG\_FS\_HCCHAR2 register.

### Synchronized OUT Transmission

The flowchart for a typical synchronized OUT transfer in slave mode is shown below. Assumptions:

- The program needs to send one packet of maximum packet length per frame starting with the odd number of frames (sending length of 1024 bytes).
- The periodic transmit FIFO is capable of storing 1 packet of maximum packet length (1024 bytes).
- The periodic request queue depth is 4. the operation flow is as follows:

- To initialize and enable channel 1, the program needs to set the ODDFRM bit of the OTG\_FS\_HCCHAR1 register.
- Write the first packet to be sent to channel 1. For high-bandwidth synchronous transmissions, the application needs to write all subsequent packets to be sent to the channel based on the MCNT bit (maximum number of packets to be sent in the next frame time) before switching to another channel.
- After the last DWORD data is written for each packet, the controller writes a request to the periodic request queue.
- The controller will send the OUT token on the next odd frame.
- When the last packet has been correctly transmitted, the controller generates an XFRC interrupt.
- In the XFRC interrupt handler, the channel needs to be reinitialized in preparation for the next transmission.
- Controls non-ACK responses.

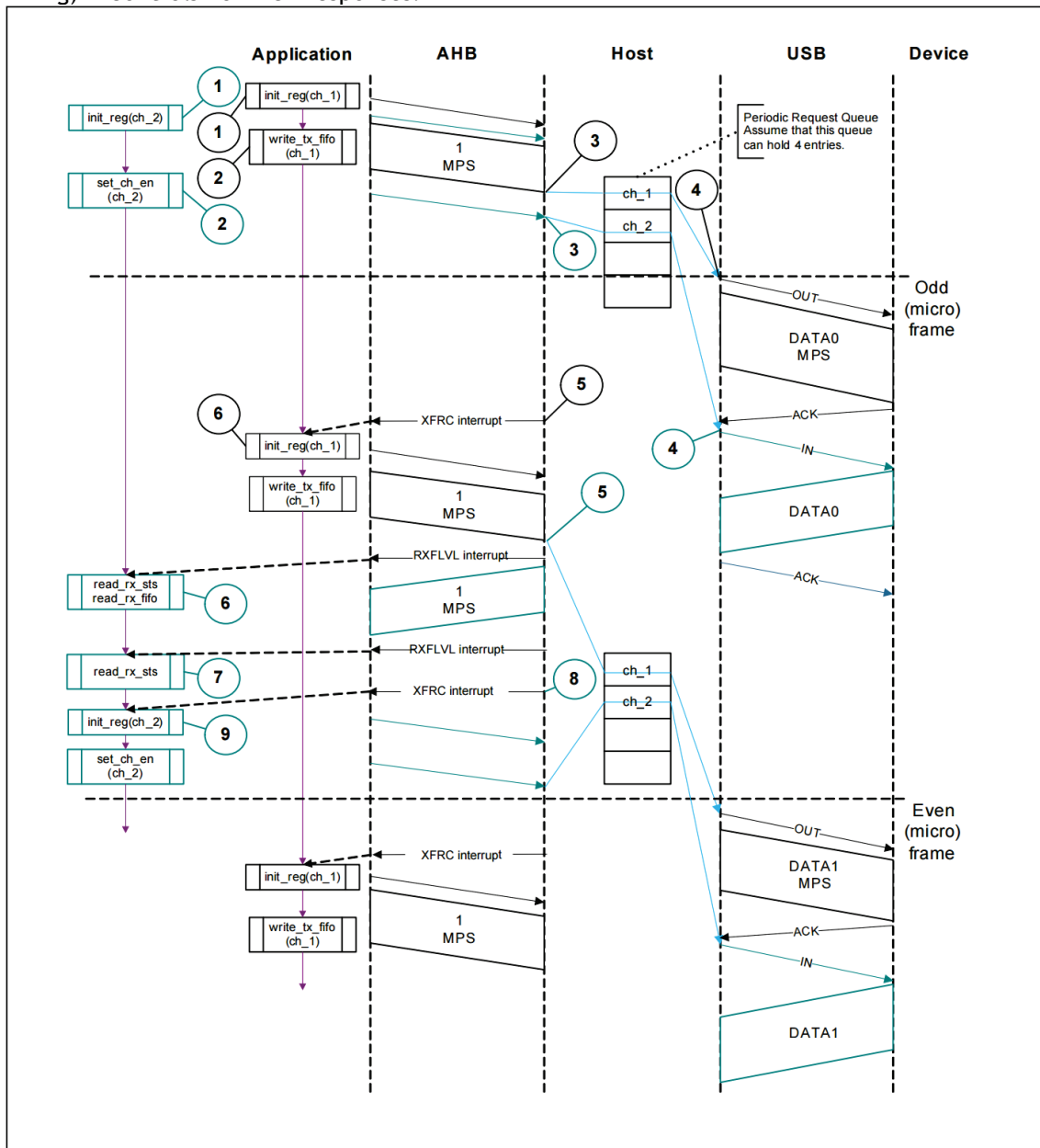


Figure 287 Normal Isochronous OUT/IN transmission process

## Interrupt Handler for Synchronized OUT/IN Transmission

Code example: synchronize OUT:

```
1. Enable (FRMOR/XFRC)
2. if(XFRC)
3. {
4.     Redistribution of channels
5. }
6. else if(FRMOR)
7. {
8.     Enable CHH
9.     Stop channel}
10. else if(CHH)
11. {
12.     Shield CHH
13.     Redistribution of channels
14. }
15. Code example: Synchronization IN
16. Enable (TXERR/XFRC/FRMOR/BBERR)
17. if(XFRC or FRMOR)
18. {
19.     if(XFRCandOTG_FS_HCTSIZx.PKTCNT==0)
20.     {
21.         Reset Error Count
22.         Redistribution of channels
23.     }
24. else
25.     {
26.         Enable CHH
27.         abortive channel
28.     }
29. }
30. else if(TXERR or BBERR)
31. {
32.     Increase error count
33.     Enable CHH
34.     abortive channel
35. }
36. else if(CHH)
37. {
38.     Shield CHH
39.     if(transfer complete or (error count == 3))
40.     {
41.         Redistribution of channels
42.     }
43. else
44.     {
45.         Reinitialize channels
46.     }
47. }
48.
```

## Synchronized IN Transmission

Hypothesis:

- The program needs to receive a maximum data length packet (transmission length of 1024 bytes) for each frame starting from the odd numbered frames.
- The receive FIFO can store at least one packet of maximum data length and two status words of type DWORD (1031 bytes total) specific to each packet.
- The periodic request queue depth is 4. the operation flow is as follows:
  - a) To initialize channel 2, the application needs to set the ODDFRM bit of the OTG\_FS\_HCCHAR2 register.
  - b) Setting the CHENA bit of the OTG\_FS\_HCCHAR2 register causes the controller to write IN requests to the periodic request queue. For high bandwidth synchronous transmissions, the application needs to write the MCNT bit of the OTG\_FS\_HCCHAR2 register (maximum number of packets to be received in the next frame time) before switching to another channel.

- 
- c) Each time the CHENA bit of the OTG\_FS\_HCCHAR2 register is set, the controller writes an IN request to the periodic request queue.
  - d) The controller will send the IN token on the next odd frame.
  - e) The RXFLVL interrupt is generated when the controller receives the IN packet and writes it to the receive FIFO.
  - f) In the RXFLVL interrupt handler, the status of the received packet is read and the number of bytes to be received is determined, and then the receive FIFO is read. The application program needs to mask the RXFLVL interrupt before reading the receive FIFO and enable it after reading.
  - g) The RXFLVL interrupt is generated after the controller writes the status information of the completed transmission to the receive FIFO. At this point the application program needs to read the status and discard the packet after determining that it is not an IN packet (PKTST bit  $\neq$  0x0010 in the OTG\_FS\_GRXSTSR register).
  - h) The controller will generate an XFRC interrupt after reading the received packet status.
  - i) In the XFRC interrupt handler, the application needs to read the PKTCNT bit of the OTG\_FS\_HCTSIZ2 register. If the PKTCNT bit is not 0, the application needs to abort the channel and reinitialize the channel in preparation for the next transmission. If PKTCNT is 0, reinitialize the channel in preparation for the next transmission. At this point the application needs to reset the ODDFRM bit of the OTG\_FS\_HCCHAR2 register.

### Setting Queue Depth

Care needs to be taken to choose the depth of the periodic and non-periodic request queues to match the number of periodic/non-periodic endpoints.

The depth of the acyclic request queue only affects the efficiency of acyclic transfers. The deeper the request queue (in conjunction with sufficient FIFO space), the more acyclic transfers the controller can pipeline. If the request queue is small, the controller can only write a new request if there is room left in the request queue.

The depth of the periodic request queue will have a large impact on the scheduling of periodic transmissions. Be sure to determine the depth of the periodic request queue by the number of packets to be periodically transmitted in each microframe time. In slave mode, the application also needs to take into account invalid requests that must be written to the request queue. Therefore, if there are 2 non-high-bandwidth periodic endpoints, the depth of the periodic request queue is at least 4. If at least one high-bandwidth periodic endpoint is supported, the request queue depth must be 8. If the depth of the periodic request queue is less than the number of packets to be periodically transmitted in each microframe time, a frame overflow occurs.

### Mismanagement

The OTG\_FS controller manages two types of confusion: packet confusion and port confusion. Packet confusion occurs when a device sends data that exceeds the maximum packet length supported by the host channel. Port confusion occurs when the host keeps receiving data from the device at EOF2 (End-of-Frame Type II signal, which is very close to the head-of-frame signal). When the OTG\_FS controller detects packet garbled, it will stop writing data to the receive buffer and wait for an end-of-packet signal (EOP). When an EOP is detected, the controller will clear the data that has been written to the buffer and generate a garbled interrupt to notify the application program.

When the OTG\_FS controller detects port confusion, it clears the receive FIFO and aborts the port, and generates a port invalidation interrupt (HPRTINT bit of the OTG\_FS\_CINTSTS register and PENCHNG bit of the OTG\_FS\_HPRT register). The application program handling this interrupt needs to read the POCA bit of the OTG\_FS\_HPRT register to rule out that the invalidation is due to port overcurrent (which is another source of port invalidation interrupts) before performing a software reset operation. The controller cannot send any more tokens after a port disruption event has been detected.

---

## 29.17.5 Programming Rules in Device Mode

### Endpoint Initialization at USB Reset

1. Set all OUT endpoints to the NAK state.
  - Set the SNAK bit of the OTG\_FS\_DOEPTCTLx (all OUT endpoints) register to '1'
2. Enable the following interrupt bits:
  - Set INEP0 bit of OTG\_FS\_DAINTEMSK register to '1' (control IN endpoint 0)
  - Set the OUEP0 bit of the OTG\_FS\_DAINTEMSK register to '1' (control OUT endpoint 0)
  - Set the STUP bit of the DOEPTMSK register to '1'
  - Set the XFRC bit of the DOEPTMSK register to '1'
  - Set the XFRC bit of the DIEPTMSK register to '1'
  - Set the TOC bit of the DIEPTMSK register to '1'
3. Allocate RAM space for each FIFO
  - Set the OTG\_FS\_GRXFSIZ register for receiving OUT data and SETUP data for control transmission. The minimum value of the allocated RAM should be the length of 1 maximum packet for control endpoint 0 + 2 DWORD spaces (for status information of OUT packets for control transfer) + 10 DWORD spaces (for SETUP packets).
  - Configure the OTG\_FS\_GNPTXFSIZ register (according to the selected FIFO number) for sending IN data for control transmission. The minimum value of the allocated RAM should be 1 maximum packet length for control endpoint 0.
4. Set the following bits of the register associated with the endpoint to control the reception of SETUP packets at OUT endpoint 0.
  - Set STUPCNT = 3 of the OTG\_FS\_DOEPTSIZ0 register (for receiving 3 consecutive SETUP packets).

At this point, initialization is complete and you can start receiving SETUP packets.

### Endpoint initialization after Enumeration is Complete

1. On the enumeration completion interrupt (ENUMDNE bit of the OTG\_FS\_GINTSTS register), read the OTG\_FS\_DSTS register to get information about the enumeration speed.
2. Configure the MPSIZ bit of the OTG\_FS\_DIEPTCTL0 register to set the maximum packet length. This step configures control endpoint 0. For control endpoints, the maximum packet length depends on the enumeration speed.

At this point, the device is ready to receive SOF packets and is able to perform control transmissions on endpoint 0.

### Endpoint Configuration for Set Address

The action to be performed by an application program after it receives a SetAddress command in a SETUP packet:

1. Configure the OTG\_FS\_DCFG register with the device address information included in the SetAddress command.
2. Configure the controller to send out status IN packets.

### Endpoint Configuration at Set Configuration/Set Interface Command

An action to be performed by an application after it receives a Set Configuration or Set Interface command in a SETUP package:

1. When the Set Configuration command is received, the endpoint registers need to be configured according to the attributes defined in the new configuration.
2. When you receive the Set Interface command, configure the endpoint registers involved in the command.
3. Some of the endpoints may have been valid only in the previous setup and may have been invalidated in the new configuration or setup; these invalidated endpoints need to be reconfigured to be invalid.
4. Set the OTG\_FS\_DAINTEMSK register to enable interrupts for each valid endpoint and block interrupts for all invalid endpoints.
5. Allocate RAM space for each FIFO.
6. After all endpoints have been configured, the application needs to enable the controller to send a status IN packet. At this point, the controller in device mode is ready to send or receive packets.

## Endpoint Activation

Steps to activate an endpoint or configure an existing endpoint as a new type:

1. Configure the following bits of the OTG\_FS\_DIEPCTLx register (for IN or bidirectional endpoints) or the OTG\_FS\_DOEPCTLx register (for OUT or bidirectional endpoints):
  - Maximum Packet Length
  - USB effective endpoint bit = '1'
  - Data rollover number at the start of the endpoint (for interrupt and block transfer type endpoints)
  - Types of endpoints
  - Send FIFO number
2. Once an endpoint is activated, the controller will parse the tokens sent to that endpoint and send valid handshake packets for valid tokens.

## Invalid Endpoints

An operation that invalidates a pre-existing endpoint:

1. Clear the USB valid endpoint bit in the OTG\_FS\_DIEPCTLx register (for IN or bidirectional endpoints) or the OTG\_FS\_DOEPCTLx register (for OUT or bidirectional endpoints).
2. Once an endpoint fails, the controller will ignore tokens issued to that endpoint, which will result in a USB timeout.

*Notes: The application program requires the following configuration for the controller in device mode to control communications: the GINTMSK register of the NPTXFEM and RXFLVLM bits must be cleared '0'.*

## 29.17.6 Workflow

### SETUP and OUT Data Transfer

This section describes the internal data flow of OUT data transfers and SETUP transfers and the operational flow of the application program.

- read operation

To read a packet (OUT and SETUP packets) from the receive FIFO in slave mode, proceed as follows:

1. In the RXFLVL interrupt handler, the receive status pop-up register (OTG\_FS\_GRXSTSP) needs to be read.
2. Write RXFLVL=0 (OTG\_FS\_GINTSTS register) to mask the RXFLVL interrupt during packet reads from the receive FIFO.
3. If the byte count of the received packet is not 0, the corresponding data will be popped out of the data FIFO and stored into memory, if the byte count of the received packet is 0, no data will be popped out of the data FIFO.
4. The packet status read from the receive FIFO is one of the following patterns:
  - a) Global OUTNAK mode:
 

PKTSTS = global OUTNAK, BCNT = 0x000, EPNUM = don't care (0x0), DPID = don't care (0x0).

Such a data representation has a global OUTNAK.
  - b) SETUP package mode:
 

PKTSTS = SETUP, BCNT = 0x008, EPNUM = control endpoint number, DPID = D0. Such a data table shows that the SETUP packet for the specified endpoint is valid and can be read from the receive FIFO.
  - c) The SETUP phase has completed mode:
 

PKTSTS = SETUP phase completed, BCNT = 0, EPNUM = control endpoint number, DPID = don't care (0x0).

Such data indicates that a SETUP phase for the specified endpoint has been completed and a data phase has begun. After this message is popped out of the receive FIFO, the controller will generate a SETUP interruption for the specified control OUT endpoint.

Disconnect.
  - d) Data OUT packet mode:
 

PKTSTS = data OUT, BCNT = received OUT packet length ( $0 \leq BCNT \leq 1024$ ), EPNUM = received endpoint number of the received packet, DPID = actual data PID.
  - e) The data phase has completed the model:

PKTSTS = data OUT phase completed, BCNT = 0x0, EPNUM = OUT endpoint data transfer endpoint number, DPID = don't care (0x0).

Such data indicates that an OUT data phase has been completed for the specified OUT endpoint. After this message is popped out of the receive FIFO, the controller will generate a transmission completion interrupt for the specified OUT endpoint.

5. The RXFLVL interrupt (OTG\_FS\_GINTSTS) needs to be turned on again after reading out the data in the receive FIFO.
6. The above five steps need to be repeated each time an RXFLVL(OTG\_FS\_GINTSTS) interrupt is detected. Reading an empty receive FIFO can result in undefined consequences.

Below is a flowchart of the above process

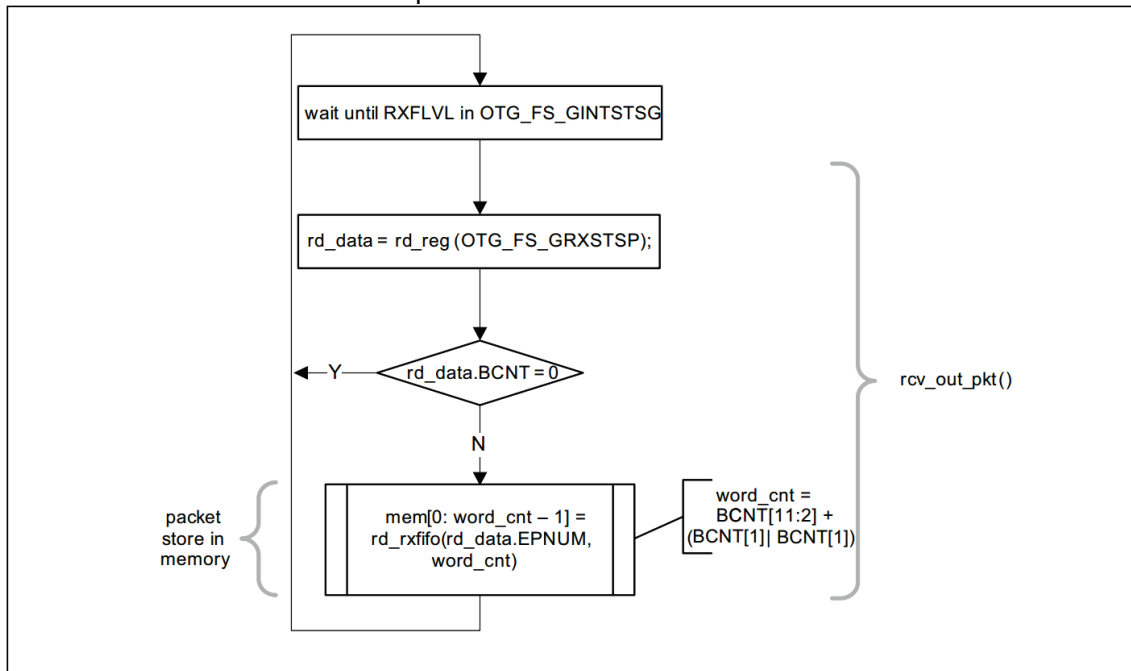


Figure288 Reading out a data message from a receive FIFO in slave mode

#### ● SETUP transmission

This section describes how the controller controls a SETUP packet and how the application handles SETUP transfers.

#### ● Requirements for the application

1. In order to receive a SETUP packet, the STUPCNT bit (in the OTG\_FS\_DOEPTSIZE register) that controls the OUT endpoint must be configured to a non-zero value. When the application program configures the STUPCNT bit to be non-zero, the controller receives the SETUP packet and writes it to the receive FIFO, which operates the same way as the NAKSTS of the OTG\_FS\_DOEPTCTL register and the

The setting of the EPENA bit is independent. Whenever the control endpoint receives a SETUP packet, the value of the STUPCNT bit is self-decremented by 1. If the an appropriate STUPCNT value is not set before receiving a SETUP packet, the controller will still receive the SETUP packet and self-subtract the STUPCNT bit, but the application program would not know how many correct SETUP packets were actually received during the SETUP phase of the control transfer. correct SETUP packets.

- Set STUPCNT bit of OTG\_FS\_DOEPTSIZE register = 3
- 2. The program needs to allocate some extra space in the receive FIFO to receive up to 3 SETUP packets.
  - The space to be allocated is 10 DWORDs. three DWORD spaces are used for the first SETUP packet, one DWORD space is used for the SETUP phase completion information, and six DWORD spaces are used to store two additional SETUP packets for the control endpoint.
  - A SETUP packet requires 3 DWORD spaces, 8 bytes of which are used to store the SETUP data, 4 bytes

Used to store SETUP status (SETUP packet mode). The controller reserves this space for received data.



- 
- FIFO is used only for SETUP packets, not for data packets.
  - 3. The application needs to read the SETUP packet's 2 DWORD length data from the receive FIFO.
  - 4. The application needs to read a DWORD SETUP phase completion message from the receive FIFO.
  - Internal data processes
  - 5. When a SETUP packet is received, the controller stores the received data into the receive FIFO, and at this time, it does not detect whether there is any space left in the receive FIFO, nor does it detect the state of the NAK or STALL bit at that endpoint.
    - Upon receipt of a SETUP packet, the controller internally sets the status bits controlling the IN/OUT endpoints to INNAK and OUTNAK.
  - 6. Upon receipt of each SETUP packet from the USB line, 3 DWORD's of data are written to the receive FIFO and the STUPCNT bit is self-subtracted by one.
    - The first DWORD data contains control information used internally by the controller.
    - The second DWORD data contains the first 4 bytes of the SETUP command.
    - The third DWORD data contains the last 4 bytes of the SETUP command.
  - 7. When the SETUP phase is complete and the data IN/OUT phase begins, the controller writes a message (SETUP phase completion message of type DWORD) to the receive FIFO indicating completion of the SETUP phase.
  - 8. The application reads the SETUP packet over the AHB bus.
  - 9. When the application takes out the SETUP phase completion message of type DWORD in the receive FIFO, the controller generates a STUP interrupt (OTG\_FS\_DOEPINTx) indicating that the SETUP packet has been received and that the application program can begin processing the received SETUP packet.
    - The controller will clear the endpoint enable bit that controls the OUT endpoint.
  - application processing flow
  - 1. Configuring the OTG\_FS\_DOEPTSIZE Registers
    - STUPCNT 3 =
  - 2. Wait for the RXFLVL interrupt (OTG\_FS\_GINTSTS) and read the packet in the receive FIFO.
  - 3. Wait for the STUP interrupt (OTG\_FS\_DOEPINTx) indicating a successfully completed SETUP phase.
    - In this interrupt handler, the application needs to read the OTG\_FS\_DOEPTSIZE register to know how many received how many SETUP packets were received and to process the last received SETUP packet.



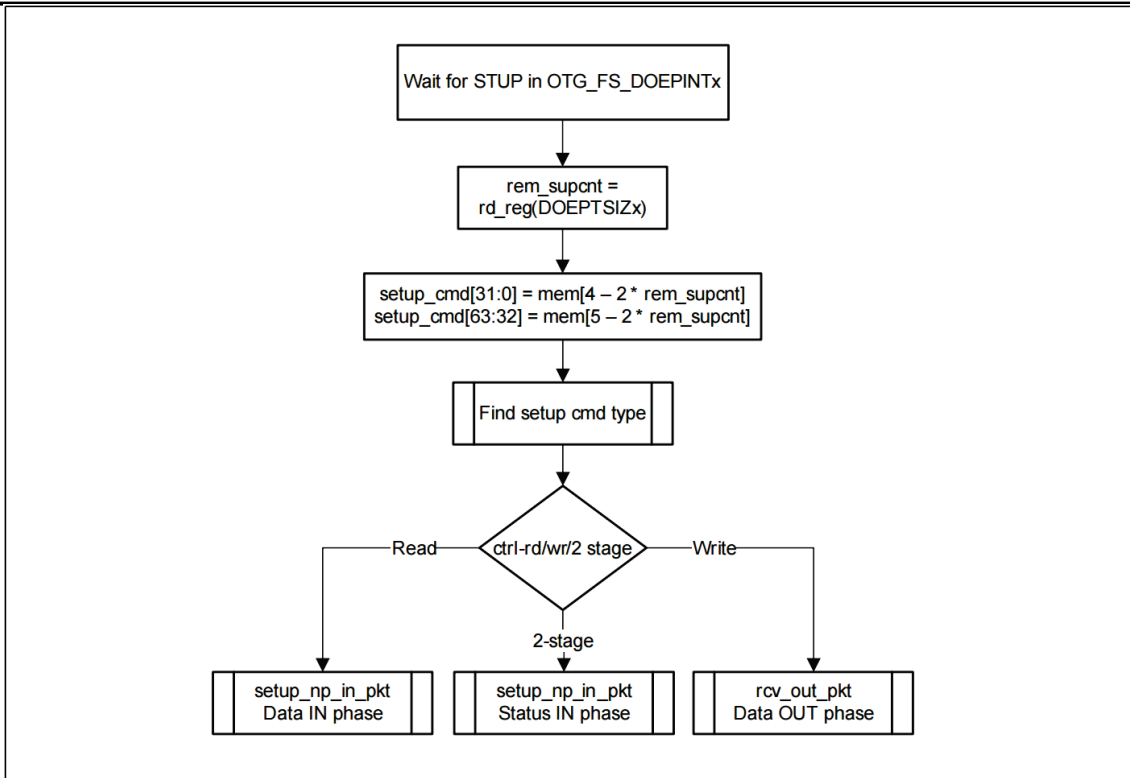


Figure289 Processing a SETUP data message

- Processing more than 3 consecutive SETUP packets

According to the USB 2.0 specification, normally a host cannot send more than three consecutive SETUP packets to the same port if there is an erroneous SETUP packet. However, the USB 2.0 specification does not limit the number of consecutive SETUP packets that a host can send to the same port. When this occurs, the OTG\_FS controller generates a B2BSTUP interrupt (OTG\_FS\_DOEPINTx).

- Setting the global OUTNAK

Internal data flow:

1. When a global OUTNAK is set by the application program (SGONAK bit in the OTG\_FS\_DCTL register), the controller will stop writing any data to the receive FIFO except for SETUP packets. Regardless of whether there is space remaining in the receive FIFO, the asynchronous OUT endpoint will receive the NAK handshake signal and the controller will discard the synchronized OUT packet.
2. The controller will write the global OUTNAK message to the receive FIFO, and the application needs to ensure that there is enough space in the receive FIFO to write this message.
3. The controller generates a GONAKEFF interrupt (OTG\_FS\_GINTSTS) when a global OUTNAK message of type DWORD is read from the receive FIFO.
4. Once this interrupt is detected, it can be determined that the controller is in the global OUTNAK state. The application program can clear this interrupt by clearing the SGONAK bit in the OTG\_FS\_DCTL register.

application processing flow

1. The application needs to set the global OUTNAK bit to stop the controller from receiving any data into the receive FIFO.
  - SGONAK of OTG\_FS\_CTL register 1 =
2. Wait for the GONAKEFF interrupt (OTG\_FS\_GINTSTS), which instructs the controller to stop receiving any data except SETUP packets.
3. After setting the SGONAK bit in the OTG\_FS\_DCTL register, the application can receive valid OUT packets before the controller generates the GONAKEFF interrupt.
4. This interrupt can be temporarily masked by writing the GINAKEFFM bit of the GINTMSK register.
  - GINAKEFFM of the GINTMSK register 0 =

5. The SGONAK bit of the OTG\_FS\_DCTL register can be cleared once it is necessary to exit global OUTNAK mode. This operation will also clear the GONAKEFF (OTG\_FS\_GINTSTS) interrupt.
  - CGONAK bit of OTG\_FS\_DCTL register = 1
6. If this interrupt has been masked, you can enable the interrupt by doing the following.
  - GINAKEFFM of the GINTMSK register 1 =

● Abort an OUT endpoint

The following procedure is required to abort an enabled OUT endpoint:

1. Before aborting an OUT endpoint, the application needs to enable the global OUTNAK state.
  - SGONAK of OTG\_FS\_DCTL register 1 =
2. Wait for GONAKEFF interrupt (OTG\_FS\_GINTSTS)
3. Abort the OUT endpoint by programming the following bits
  - EPDIS=1 for OTG\_FS\_DOEPCTLx registers
  - OTG\_FS\_DOEPCTLx register for SNAK 1 =
4. Wait for the EPDISD interrupt (OTG\_FS\_DOEPINTx), which indicates that the OUT endpoint has been successfully aborted. When the EPDISD interrupt is generated, the controller will simultaneously clear the following bits:
  - EPDIS=0 for OTG\_FS\_DOEPCTLx registers
  - EPENA for OTG\_FS\_DOEPCTLx register 0 =
5. The global OUTNAK state needs to be cleared so that other OUT endpoints that are not suspended can receive data normally.
  - OTG\_FS\_DCTL register for SGONAK 0 =

● Normal asynchronous OUT data transfer

This section describes a standard unsynchronized OUT data transfer process (control, block or interrupt transfers).

Requirements for the application:

1. Before enabling an OUT transmission, a buffer area needs to be allocated in the storage area for holding all data received in the OUT transmission.
2. For OUT transmissions, the transmission length bit in the endpoint's transmission length register needs to be set to a multiple of the endpoint's maximum packet length, aligned with the DWORD type.
  - Transmission Length [EPNUM] =  $n \times (\text{MPSIZ}[\text{EPNUM}] + 4 - (\text{MPSIZ}[\text{EPNUM}] \bmod 4))$
  - Number of packets [EPNUM]  $n =$
  - $n > 0$
3. In each OUT endpoint interrupt, it is necessary to read the transmission length register of the endpoint in order to calculate the length of the data in the buffer. The received data length can be smaller than the set transfer length.
  - Data length in the buffer = initially configured transmission length - controller updated transmission length.

— Number of USB packets received = Number of packets initially configured - Number of packets updated by the controller

Internal data flow:

1. The application program needs to configure the Transmit Length bit and the Packet Number bit of the relevant endpoint registers, clear the NAK bit, and enable the endpoint to receive data.
2. Once the NAK bit is cleared, the controller will start receiving data and write the data into the receive FIFO when there is space left in the receive FIFO. For each packet received, the packet and its status information will be written to the receive FIFO. For each packet (maximum packet length packet or short packet) written to the receive FIFO, the packet number bit of the endpoint register will be self-subtracted by one.
  - An OUT packet with an incorrect CRC is automatically cleared from the receive FIFO.
  - After sending an ACK signal, the host does not receive the ACK signal and resends an unsynchronized ACK signal.

OUT packets, which will be discarded by the controller. The application program does not have to deal with packets with the same data PID number for the same endpoint. consecutive OUT packets, in which case the packet digits of the registers do not self-decrement.

- If there is no space left in the receive FIFO, either synchronized or unsynchronized packets will be discarded and will not be written to the receive FIFO. In addition, a NAK handshake signal will be sent for unsynchronized OUT packets.
- For all three of these cases, since none of the actual data is written to the receive FIFO, the packet number bit of the endpoint register are not self-decremented.
- 3. When the number of packets self-decreases to 0, or the endpoint receives a short packet, the controller sets the state of that endpoint to NAK. Once the endpoint is in the NAK state, packets for both synchronous and asynchronous transmissions are discarded without being written to the receive FIFO, while for asynchronous OUT transmissions, the controller returns a handshake packet of NAK.
- 4. After the data has been written to the receive FIFO by the controller, the application program can read the data from the receive FIFO and write it to other storage areas. Each endpoint can only operate on one packet of data at a time.
- 5. Whenever a packet's data is written to other storage areas over the AHB bus, the value of the endpoint's transfer length register is automatically subtracted from the length of the data that has been written.
- 6. The OUT transmission completion flag of the OUT endpoint is written to the receive FIFO when the
  - The value of the Transmit Length register is 0, as well as the value of the Packet Count register.
  - The last OUT packet written to the receive FIFO is a short packet ( $0 \leq \text{received packet length} < \text{maximum packet length}$ )
- 7. When the application program reads this flag (OUT transmission completion flag) out of the receive FIFO, the controller generates a transmission completion interrupt for that endpoint, and at the same time, the enable flag for that endpoint is cleared.

#### Application Processing Flow:

1. Configure the OTG\_FS\_DOEPTSLx register to set the transmission length and the appropriate number of packets.
2. Configure the OTG\_FS\_DOEPCTLx register to set the characteristics of the endpoint, and also set the EPENA and CNAK bits to '1'.
  - EPENA=1 for OTG\_FS\_DOEPCTLx registers
  - CNAK of OTG\_FS\_DOEPCTLx register 1 =
3. Wait for the RXFLVL(OTG\_FS\_GINTSTS) interrupt and read the packet from the receive FIFO.
  - Depending on the length of the transmission, this step will be repeated several times.
4. Wait for the XFRC(OTG\_FS\_DOEPINTx) interrupt of the asynchronous OUT transfer normal completion flag.
5. Read the OTG\_FS\_DOEPTSLx register to get information about the actual length of data received.
- Normal synchronized OUT data transfer

This chapter describes the standard synchronized OUT transfer processing.

#### Requirements for the application:

1. All requirements for asynchronous OUT transmission apply to synchronous OUT transmission.
2. For synchronous OUT transmissions, both the Transmission Length bit and the Number of Packets bit of the Control Register must be set to the maximum packet length and number of packets received within a frame. Synchronous OUT transmissions cannot span a frame.
3. The application needs to read all synchronized OUT packets (both data and status) from the receive FIFO before the end of a periodic frame (EOPF interrupt in the OTG\_FS\_GINTSTS register).
4. In order to be ready to receive data for the next frame, the synchronization OUT endpoint needs to be enabled after EOPF(OTG\_FS\_GINTSTS) and before SOF(OTG\_FS\_GINTSTS).

#### Internal data flow:

1. The internal data flow of the synchronized OUT endpoints is roughly the same as that of the unsynchronized OUT endpoints, except for some minor differences.
2. When the Synchronization OUT endpoint is enabled (by setting the endpoint enable bit) and the NAK state is cleared, the application must set the appropriate odd/even framing bits. The controller will only receive data in special frames at the Synchronization OUT endpoint if the
  - EONUM (OTG\_FS\_DOEPCTLx register) = SOFFN[0] (OTG\_FS\_DSTS register)

3. After reading a complete synchronization OUT packet (including data and status) from the receive FIFO, the controller updates the RXDPID bit of the OTG\_FS\_DOEPTSIZx register based on the last synchronization OUT packet read from the receive FIFO.

Application operation flow:

1. Configure the OTG\_FS\_DOEPTSIZx register to set the appropriate transmission length and packet count.
2. Configure the OTG\_FS\_DOEPCTLx register to set the characteristics of the endpoints and enable the endpoints, as well as clear the NAK state and select the appropriate odd/even frames.
  - EPENA=1
  - CNAK=1
  - EONUM=(0: even/1: odd)
3. Wait for the RXFLVL interrupt (OTG\_FS\_GINTSTS) and read the packet from the receive FIFO
  - Depending on the length of the transmission, this step will be repeated several times.
4. Wait for the XFRC interrupt generated by the synchronized OUT transfer end flag (OTG\_FS\_DOEPINTx). This interrupt does not mean that the data in the memory area is correct.
5. In synchronous transfers, the XFRC interrupt is not always generated, but the IISOOXFRM interrupt of the OTG\_FS\_GINTSTS register can be detected.
6. Read the OTG\_FS\_DOEPTSIZx register in order to know how much valid data has been received within a frame. Received data is determined to be valid only when one of the following conditions is met.

Number of USB packets received within this frame = number of initialized packets - final number of packets updated by the controller.

— RXDPID=D0(OTG\_FS\_DOEPTSIZx) while the number of USB packets received within this frame is 1.

— RXDPID=D1(OTG\_FS\_DOEPTSIZx) while the number of USB packets received within this frame is 2.

— RXDPID = D2(OTG\_FS\_DOEPTSIZx) while the number of USB packets received within this frame is 3. In this frame

Number of USB packets received within this frame = number of initialized packets - final number of packets updated by the controller. The application program can discard

The application program can discard invalid packets.

- Incomplete synchronization of OUT data transfer

This section describes the processing flow when a synchronized OUT packet is received incompletely.

Internal data flow:

1. For synchronized OUT endpoints, the XFRC interrupt (OTG\_FS\_DOEPINTx) is not generated every time. In the following cases, the controller will stop receiving synchronized OUT packets and the application will not detect the XFRC interrupt (OTG\_FS\_DOEPINTx).
  - The controller will stop receiving when the receive FIFO does not have enough space to hold a complete synchronized OUT packet.
  - A CRC error occurs when a synchronized OUT packet is received.
  - When a synchronization OUT token is received in error.
  - When the application is too slow to read data from the receiving FIFO.
2. When the controller detects an end-of-frame signal before completing all synchronous OUT transfers, an incomplete synchronous OUT interrupt is generated (IISOOXFRM in the OTG\_FS\_GINTSTS register), which indicates that at least one of the synchronous OUT endpoints did not generate an XFRC interrupt (OTG\_FS\_DOEPINTx). In this case, the endpoint is still valid, but no further data will be transferred on the USB line.

Application Processing Flow:

1. The IISOOXFRM interrupt (OTG\_FS\_GINTSTS) indicates that an incomplete transmission has occurred at at least one of the synchronized OUT endpoints within the current frame.
2. The application needs to determine if an incomplete transfer has occurred because the receive FIFO was not read empty. It should be ensured that all synchronized OUT data (both data and status) in the receive FIFO must be read before further operations.
  - The XFRC interrupt (OTG\_FS\_DOEPINTx) is generated when the application program reads the receive FIFO empty, at which point it is necessary to re

newly enable the endpoint in order to receive synchronized OUT data on the next frame.

3. When the LISOOXFRM interrupt (OTG\_FS\_GINTSTS) is detected, it is necessary to read the control registers (OTG\_FS\_DOEPCTLx) of all synchronized OUT endpoints in order to confirm which endpoint within the current frame an incomplete transmission has occurred. An incomplete transmission is indicated when both of the following conditions are met.
  - EONUM bit (OTG\_FS\_DOEPCTLx) = SOFFN[0](OTG\_FS\_DSTS)
  - EPENA=1(OTG\_FS\_DOEPCTLx)
4. The above must be done before the SOF interrupt (OTG\_FS\_GINTSTS) is detected to ensure that the current frame number has not changed.
5. For a synchronized OUT endpoint where an incomplete transfer has occurred, it is necessary to discard the data stored in memory and invalidate the endpoint by setting the EPDIS bit in the OTG\_FS\_DOEPCTLx register.
6. Wait for the EPDIS interrupt (OTG\_FS\_DOEPINTx) and re-enable the endpoint to receive new data at the beginning of the next frame.
  - Since the controller takes some time to close the endpoint, after receiving an erroneous synchronization data, the application program may not be able to receive data at the next frame.

● Abort an unsynchronized OUT endpoint

This section describes how an application can abort an asynchronous endpoint.

1. Sets the controller into a global OUTNAK response state.
2. Invalid corresponding endpoints.
  - By setting STALL=1 (OTG-FS\_DOEPCTL) instead of setting the SNAK bit (OTG\_FS\_DOEPCTL) to invalidate the endpoint.
3. The STALL bit (OTG\_FS\_DOEPCTLx) must be cleared when the application is ready to terminate in response to the endpoint with STALL.
4. When the application program receives a SetFeature.Endpoint Halt command or a ClearFeature.Endpoint Halt command, it needs to set or clear the STALL bit of the endpoint, and this setting or clearing must be done before the control endpoint initiates the state phase transmission.

### Typical Example

This section gives some examples of how to handle basic transmission types and situations.

● Block OUT transfer in device mode

The following figure gives the flow of receiving a single block OUT packet from the USB to the AHB and the associated events.

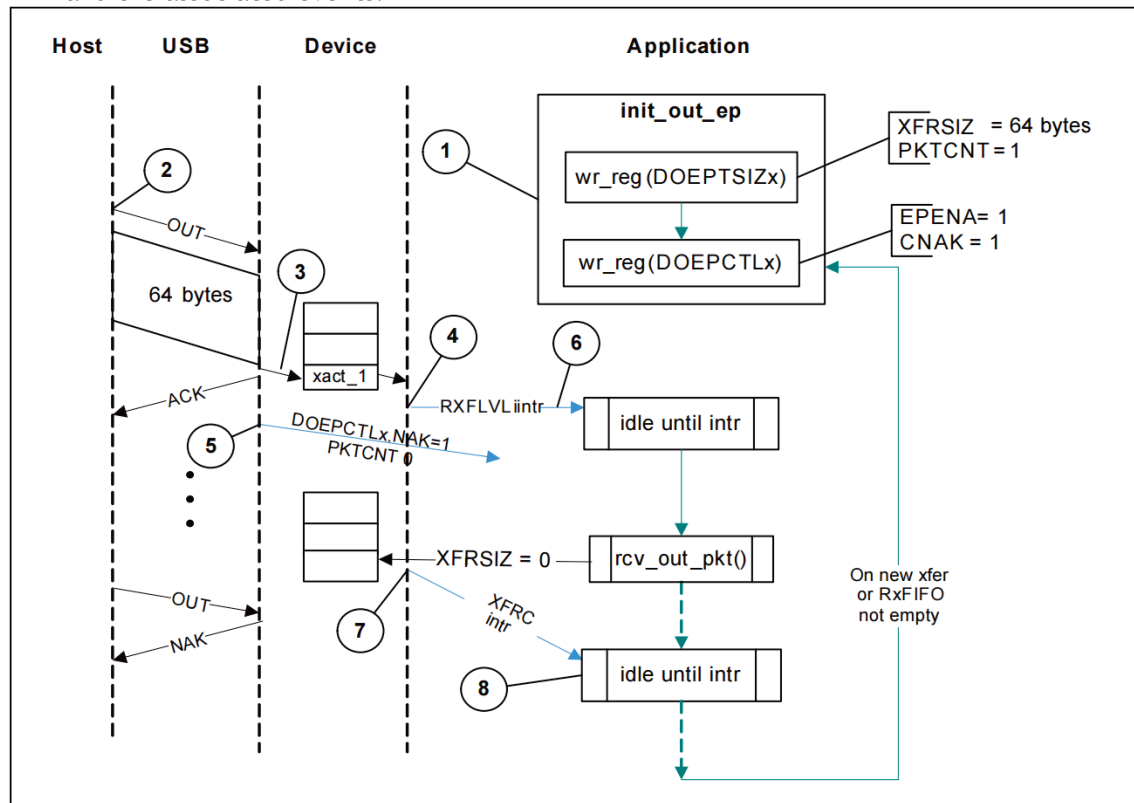


Figure290 Block (Bulk) OUT transfer process in slave mode

After receiving the SetConfiguration/SetInterface command, the application initializes all OUT endpoints by setting (in OTG\_FS\_DOEPCTLx) CNAK=1 and EPENA=1, and sets reasonable XFRSIZ and PKTCNT values for the OTG\_FS\_DOEPSIZx registers.

1. Host attempts to send data to an endpoint (OUT token)
2. When an OUT token is received on the USB line, the controller stores the packet in the receive FIFO, which should have space remaining at this time.
3. The RXFLVL interrupt (OTG\_FS\_GINTSTS) is generated when the controller writes the complete packet to the receive FIFO.
4. After receiving PKTCNT USB packets, the controller will automatically set the NAK bit of the endpoint state to prevent continual reception of more packets.
5. The application needs to respond to the interrupt and read data from the receive FIFO.
6. When all data has been read (data length equal to XFRSIZ), the controller will generate an (OTG\_FS\_DOEPINTx) XFRC interrupt.
7. The application needs to respond to the interrupt and use the XFRC interrupt bit's to determine if the entire transfer has completed.

### IN Data Transfer

This section describes how to write packets to the endpoint's FIFO in device mode when the corresponding transmit FIFO is enabled.

1. You can choose to use the polling method or the interrupt method
  - With the polling method, it is necessary to monitor the corresponding transmit FIFO status of the endpoint by reading the OTG\_FS\_DTXFSTSx register to see if there is any space left in the transmit FIFO.
  - Using the interrupt method, you need to wait for the TXFE interrupt (OTG\_FS\_DIEPINTx) and read the OTG\_FS\_DTXFSTSx register to find out if there is enough space left in the transmit FIFO.
  - If a single non-zero length packet is to be written, the remaining space in the transmit FIFO must be sufficient to write the entire complete packet.
  - If a zero-length packet is to be written, there is no need to check the remaining space in the sending FIFO.
2. Regardless of which of the above methods is used, when there is enough space left in the transmit FIFO to write a packet, the application program needs to write the endpoint control register before writing the packet to the transmit FIFO. Usually, in addition to setting the enable bit of the endpoint, the application program is required to set the OTG\_FS\_DIEPCTLx register by a read-modify-write operation to prevent the contents of the register from being modified.

If the remaining space in the transmit FIFO is large enough, multiple packets can be written to the same endpoint in succession. For periodic IN endpoints, only one packet can be written in a frame, and the data to be sent in the next cycle can be written only after the transmission completion flag of the previous packet has been received.

#### ● Setting the IN endpoint NAK state

Internal data processes:

1. When setting the state of a specific IN endpoint to NAK, the controller will disregard whether or not there is data to be transmitted in the transmit FIFO of that endpoint and stop the data transmission of that endpoint.
2. For non-synchronized IN commands, it will be replied with a NAK handshake signal.
  - For the Synchronize IN command, it will be replied with a zero-length packet.
3. For the SNAK bit of the OTG\_FS\_DIEPCTLx register, the controller will generate an INEPNE (IN endpoint NAK active) interrupt (OTG\_FS\_DIEPINTx).
4. When this interrupt is received, it can be determined that the endpoint has entered the IN endpoint NAK state. The application program can clear this interrupt by setting the CNAK bit of the OTG\_FS\_DIEPCTLx register.

Application Programming Process:

1. The INNAK bit needs to be set to stop data transfer activity at the specified IN endpoint:
  - OTG\_FS\_DIEPCTLx register with SNAK=1
2. Wait for the INEPNE interrupt from the OTG\_FS\_DIEPCTLx register. This interrupt indicates that the controller has stopped data transfer activity at the specified endpoint.
3. The controller can send a valid IN packet at this endpoint after the NAK bit is set by the application program and before generating a NAK valid interrupt.
4. The application program can temporarily mask this interrupt by writing the INEPNEM bit of the DIEPMSK register.



- INEPNEM of DIEPMSK register = 0
- 5. Exit the endpoint NAK status mode by clearing the NAK status bit (NAKSTS) of the OTG\_FS\_DIEPCTLx register. The INEPNE interrupt (OTG\_FS\_DIEPINTx) also needs to be cleared.
  - CNAK=1 for OTG\_FS\_DIEPCTLx registers
- 6. If the application has previously masked this interrupt, it needs to unmask it:
  - INEPNEM of DIEPMSK register = 1
- IN endpoint invalid

An enabled IN endpoint can be invalidated by following the steps below.

Application Programming Process:

1. The application needs to stop writing packets from the AHB before invalidating an IN endpoint.
2. The application needs to set up the endpoint to enter NAK mode
  - OTG\_FS\_DIEPCTLx register with SNAK=1
3. Wait for the INEPNE interrupt from the OTG\_FS\_DIEPINTx register.
4. For endpoints that need to be invalidated, set the following bits of the OTG\_FS\_DIEPCTLx register:
  - EPDIS=1 for OTG\_FS\_DIEPCTLx registers
  - OTG\_FS\_DIEPCTLx register with SNAK=1
5. Wait for the EPDISD interrupt in the OTG\_FS\_DIEPINTx register, which indicates that the controller has invalidated the specified endpoint. Along with this interrupt, the controller will also clear the following bits:
  - EPENA=0 for OTG\_FS\_DIEPCTLx registers
  - EPDIS=0 for OTG\_FS\_DIEPCTLx registers
6. The application must read the OTG\_FS\_DIEPTSIZx register of the periodic IN endpoint to see how much data the endpoint has sent to the USB bus.
7. The application needs to clear the data in the transmit FIFO of this endpoint by setting the following bits in the OTG\_FS\_GRSTCTL register:
  - TXFNUM(OTG\_FS\_GRSTCTL)=Transmit FIFO number of the endpoint
  - TXFFLSH(OTG\_FS\_GRSTCTL)=1

The application needs to query the OTG\_FS\_GRSTCTL register until the TXFFLSH bit is cleared by the controller, which indicates that the refresh operation is complete. The application can then re-enable the endpoint to send new data.

- Ordinary non-periodic IN data transfer

The application needs:

1. Before initiating an IN transfer, it is necessary to ensure that all data to be sent via the IN transfer belongs to the same buffer.
2. For IN transmissions, the Transmit Length bit of the Endpoint Transmit Length Register indicates the entire length of data to be sent, including several maximum packet length packets and one short packet. The short packet will be transmitted at the end of the transmission.
  - In order to have to send a number of packets of maximum packet length first and send a short packet at the end of the transmission:

Transmission length [EPNUM] =  $x \times \text{MPSIZ} [\text{EPNUM}] + \text{sp}$

If (sp>0)

Number of packages [EPNUM]=x+1

else

Number of packages [EPNUM] = x

- In order to send a single zero-length packet:

Transmission length [EPNUM]=0

Number of packages [EPNUM]=1

- In order to send a number of maximum packet length packets first and a zero-length packet at the end of the transmission, the application needs to split the entire sending process into two parts.

packet, the application needs to split the entire sending process into two parts. The first part sends a number of packets of maximum packet length, and the second part sends a separate packet of zero length. packets of maximum packet length, and the second part sends a separate packet of zero length.

First part: transmission length [EPNUM] =  $x \times \text{MPSIZ} [\text{EPNUM}]$ ; number of packets [PENUM] = n

Part II: Transmission Length [EPNUM]=0; number of packets [EPNUM]=1

3. Once an endpoint starts sending data, the controller updates the value of the transfer length register, which needs to be read at the end of an IN transfer to see how much data in the sending FIFO has been sent out over the USB bus.
4. Data still remaining in the transmit FIFO = set transmission length - final transmission length updated by the controller.
  - Data transferred on the USB bus = (initial number of packets set - final number of packets updated by the controller) × MPSIZ[EPNUM]
  - Data still to be sent = Initial transmission length set - Length of data already sent via USB.

Internal data processes:

1. The application program needs to set the Transmit Length and Packet Number bits of the Endpoint Control Register and enable the endpoints that need to transmit data.
2. The application needs to write the data to be transferred to the transmit FIFO of the corresponding endpoint.
3. Whenever the application writes a packet to the transmit FIFO, the controller will automatically subtract the length of the written packet from the transmission length. It is necessary to keep writing data until the transmission length of that endpoint becomes 0. After writing data to the FIFO, the number of packets in the FIFO bit (each IN endpoint indicates the number of packets in 3 bits, internally managed by the controller, the maximum number of packets that can be managed by the controller for an IN endpoint at any one time is 8) will be automatically increased by 1. For zero-length packets, there will be a corresponding flag bit to indicate that no data needs to be in the FIFO. does not need to have any data.
4. After data is written to the transmit FIFO, the controller will read this data upon receipt of an IN command. For each asynchronous IN packet transmission that ends with an ACK, the packet count register at that endpoint will self-decrement by 1 until the packet count becomes 0. The packet count register will not self-decrement due to a timeout.
5. For zero-length packets (indicated by the internal zero-length flag bit), the controller will send zero-length packets according to the IN command and the packet count register is self-decremented by one.
6. If an IN command is received when the packet count register has changed to 0 and there is no more data in the FIFO to be sent, the controller generates an interrupt (ITTXFE) with the words "An IN command was received when the transmit FIFO was empty" without setting the NAK flag bit for that endpoint. The controller will respond to a transmission from an asynchronous endpoint with a NAK handshake signal.
7. The controller internally recovers the pointer to the FIFO and does not generate a timeout interrupt.
8. If the transmission length is 0 and the packet count is also 0, the transmission completion interrupt (XFRC) is generated and the valid flag of the endpoint is cleared.

Application Process:

1. Set the OTG\_FS\_DIEPTSIZx register according to the transmission length and the corresponding number of packets.
2. Set the OTG\_FS\_DIEPTCTLx register according to the characteristics of the endpoint and set the CNAK and EPENA (endpoint enable) bits.
3. When transmitting non-zero length packets, the application needs to query the OTG\_FS\_DTXFSTSx register (where x represents the FIFO number associated with the endpoint) to see if there is any space left in the sending FIFO. The application can also use the TXFE(OTG\_FS\_DIEPINTx) interrupt to decide whether to write data to the FIFO.
- Ordinary periodic IN transmission

This section describes a typical periodic IN transmission

The application needs:

1. Application requirements 1, 2, 3, and 4 described in the previous section for normal non-periodic IN transfers apply equally well to periodic IN transfers in this section, except that 2 is slightly different.
  - An application can transmit multiple packets of maximum packet length, or multiple packets of maximum packet length with a short packet ending. packet lengths. Transmitting a number of maximum packet length packets and ending the transmission with a short packet at the end of the transmission needs to be done:  
 Transmission length [EPNUM] = x × MPSIZ [EPNUM] + sp  
 (here x is an integer ≥ 0 and 0 ≤ sp ≤ MPSIZ[EPNUM])



- If(sp>0)  
 Number of packages [EPNUM]=x+1  
 else  
 Number of packages [EPNUM] = x  
 MCNT[EPNUM] = number of packages [EPNUM]
- An application cannot send a zero-length packet at the end of a transmission, but it can automatically send a separate zero-length packet automatically. In order to send a separate zero-length packet, the setting is required:  
 Transmission length [EPNUM]=0 Packet number [EPNUM]=1  
 MCNT[EPNUM] = number of packages [EPNUM]
2. The application can only schedule one frame of data transmission at a time.
    - $(MCNT-1) \times MPSIZ \leq XFERSIZ \leq MCNT \times MPSIZ$
    - $PKTCNT = MCNT(OTG\_FS\_DIEPTSIZx)$
    - If  $XFERSIZ < MCNT \times MPSIZ$ , then the last packet transmitted is a short packet.
    - Note: The MCNT bit is in the OTG\_FS\_DIEPTSIZx register, the MPSIZ bit is in the OTG\_FS\_DIEPCTLx register, the PKTCNT bit is in the OTG\_FS\_DIEPTSIZx register, the XFERSIZ bit is in the OTG\_FS\_DIEPTSIZx register.
  3. All data to be sent within a frame must be written to the transmit FIFO before the IN command is received. Even if only one DWORD is not written to the transmit FIFO for data that needs to be sent within a frame, the controller still considers the FIFO to be empty. When the send FIFO is empty:
    - For synchronized IN endpoints, a zero-length packet is sent.
    - For interrupt IN endpoints, a NAK handshake signal is sent.
  4. For high bandwidth IN endpoints with three packets in a frame, the application needs to set the length of the endpoint FIFO to 2 x max\_pkt\_size (maximum packet length) and write the third packet after the first one has already been sent to the USB bus.

#### Internal data processes:

1. The application program needs to set the transmission length and packet number bits of the corresponding endpoint register and enable that transmission endpoint.
2. The application needs to write the data to the appropriate transmit FIFO.
3. Whenever the application writes a packet to the transmit FIFO, the Transmit Length register value for that endpoint is automatically subtracted from the length of the packet written. The application needs to keep writing data until the transmission length of the endpoint becomes 0.
4. When the periodic endpoint receives an IN command, the controller will automatically send the data in the FIFO. If there is no complete packet of the current frame in the FIFO (there is a complete packet in the specified FIFO mode), the controller generates an interrupt "Transmit FIFO Empty on Receipt of IN Command".
  - For synchronized IN endpoints, the controller sends zero-length packets
  - For interrupt IN endpoints, the controller sends a handshake response of NAK
5. The packet count register value for the corresponding endpoint is automatically decremented by 1 if the
  - For synchronized endpoints, when a zero-length or non-zero-length packet is sent
  - For interrupt endpoints, when an ACK handshake signal is transmitted
  - When both the transmit length and packet number are 0, an end-of-transmission interrupt is generated and the corresponding endpoint's enable is cleared.
6. During the Periodic Frame Interval Time (controlled by the PFIVL bit in the OTG\_FS\_DCFG register), the controller generates an IISOIXFR interrupt in the OTG\_FS\_GINTSTS register when it finds that any of the synchronization IN endpoint FIFOs, which are supposed to be empty during the current frame, is not empty.

#### Application Process:

1. Configure the OTG\_FS\_DIEPCTLx register according to the endpoint characteristics and set the CNAK and EPENA bits.
2. Writes the data that needs to be sent at the next frame into the transmit FIFO.
3. If the ITTXFE interrupt (OTG\_FS\_DIEPINTx) is generated, it indicates that the application program did not have time to write the data to be sent into the transmit FIFO.
4. When the endpoint generating the interrupt has been enabled before the interrupt is generated, disregard this interrupt; otherwise, enable the endpoint and the controller will send data when the next IN command is received.

5. When the XFRC interrupt (OTG\_FS\_DIEPINTx) is generated and there is no ITTXFE interrupt (OTG\_FS\_DIEPINTx), it indicates that the synchronous IN transfer has ended normally. The OTG\_FS\_DIEPTSIZx register can be read and both the Transfer Length register value and the Packet Count register value are equal to 0, indicating that all data has been transferred over USB.
6. When the XFRC interrupt (OTG\_FS\_DIEPINTx) is generated, with or without the ITTXFE interrupt (OTG\_FS\_DIEPINTx), it indicates that the interrupt IN transfer has ended normally. The OTG\_FS\_DIEPTSIZx register can be read, and the transfer length and packet count register values are both equal to 0, indicating that all data has been transferred over USB.
7. When an unfinished synchronized IN transmission interrupt (IISOIXFR of OTG\_FS\_GINTSTS) is generated without generating the aforementioned interrupt, it indicates that the controller is under-receiving at least one periodic IN command in the current frame.
- Outstanding Synchronized IN Data Transfers

This section describes the application's processing flow when an unfinished synchronized IN transfer occurs. Internal Data Flow:

1. Synchronized IN transmission is considered incomplete when any of the following conditions occur:
  - The controller receives an erroneous Synchronization IN command on at least one Synchronization IN endpoint. At this point, the application program detects an "Incomplete Synchronous IN Transfer" interrupt (IISOIXFR in the OTG\_FS\_GINTSTS register) is detected by the application.
  - The application is too late to write the complete data into the send FIFO, that is, before writing the complete packet to the send FIFO before writing the complete packet to the transmit FIFO. In this case, the application program detects the "Send FIFO is empty at the time of receiving IN command" interrupt (OTG\_FS\_DIEPINTx). The application can ignore this interrupt as it will cause an "outstanding" interrupt at the end of the periodic frame. At the end of the periodic frame, resulting in an "incomplete synchronous IN transmission" interrupt (IISOIXFR of OTG\_FS\_GINTSTS). The controller will send a zero-length packet to the USB bus in response to the received IN command.
2. The application should stop continuing to write data to the sending FIFO as soon as possible.
3. The application should set the NAK bit and disable bit of the endpoint.
4. The controller disables the endpoint, clears the endpoint's cancel enable bit, and generates an "endpoint cancel enable" interrupt for the endpoint.

Application Process:

1. The application can ignore the received IN command when it receives an OTG\_FS\_DIEPINTx transmit FIFO air-break on any synchronized IN endpoint, as this will result in an incomplete synchronized IN transmission interrupt (OTG\_FS\_GINTSTS).
2. The "Outstanding Synchronous IN Transmission" interrupt (OTG\_FS\_GINTSTS) indicates that an outstanding synchronous IN transmission has occurred at at least one synchronous IN endpoint.
3. The application needs to read the endpoint control registers of all synchronized IN endpoints to see which endpoint had the outstanding IN transfer event.
4. The application program must stop writing data to the transmit FIFO of this endpoint.
5. Configure the following bits of the OTG\_FS\_DIEPCTLx register to de-enable the endpoint:
  - OTG\_FS\_DIEPCTLx register with SNAK=1
  - EPDIS=1 for OTG\_FS\_DIEPCTLx registers
6. The "endpoint de-enable" interrupt in the OTG\_FS\_DIEPINTx register indicates that the controller is de-enabling the endpoint.
  - In this case, the application needs to clear the corresponding transmit FIFO or overwrite the data that still exists in the FIFO after re-enabling the endpoint. In the FIFO after re-enabling the endpoint. To clear the data, the application must use the OTG\_FS\_GRSTCTL register.

#### ● Aborting unsynchronized IN endpoints

This section describes how an application can abort an asynchronous endpoint.

Application Process:

1. Release the enable state of the IN endpoint that needs to be aborted. Set the STALL bit.
2. If the endpoint is already enabled, set EPDIS = of the OTG\_FS\_DIEPCTLx register 1.
  - STALL=1 for OTG\_FS\_DIEPCTLx.
  - The STALL bit is used for higher priority than the NAK bit.

3. The endpoint "disable interrupt" (OTG\_FS\_DIEPINTx) informs the application that the controller has de-enabled the corresponding endpoint.
4. The application program needs to clear either the aperiodic or cyclic transmit FIFO, depending on the type of transmission of the endpoint. for a non-cyclic endpoint, the application program needs to re-enable another non-cyclic endpoint that does not need to be aborted in order to complete the data transfer.
5. When the application is ready to end responding to the endpoint with the STALL handshake signal, the STALL bit of the OTG\_FS\_DIEPCTLx register must be cleared.
6. When the application program sets or clears the STALL bit according to the SetFeature.Endpoint Halt command, or the ClearFeature.Endpoint Halt command, it is necessary to perform the setting and clearing of the STALL bit before the state transfer phase is established on the control endpoint.

Special case: aborting a control OUT endpoint

The controller can STALL the IN/OUT command during the data transfer phase of the control transfer when the host sends more IN/OUT commands than the number defined in the SETUP packet. In this case, the application program must enable the ITTXFE interrupt for OTG\_FS\_DIEPINTx and the OTEPDIS interrupt for OTG\_FS\_DOEPINTx during the data transfer phase of the control transfer. When the application program receives an interrupt, it means that the STALL bit of the corresponding endpoint control register must be set and the interrupt cleared.

## 29.17.7 Worst Case Response Time

When the OTG\_FS controller is operating in device mode, there is a worst-case response time for any command that follows a synchronized OUT transmission. This worst-case response time varies depending on the clock frequency of the AHB.

The controller registers are within the range of the AHB clock domain and the controller cannot receive new commands until these register values have been updated. The worst case scenario is that for any command that follows a synchronized OUT transfer, this command may arrive immediately since synchronized transfers do not require a handshake signal. This worst case response time is 7 PHY clocks when the AHB clock is equal to the PHY clock frequency. This response time becomes faster when the AHB clock is faster.

When this worst-case scenario occurs, the controller responds to block transfer and interrupt transfer commands with NAK, discarding synchronization and SETUP commands. For SETUP commands, the host handles this situation with a SETUP timeout and retransmits the SETUP packet. For synchronous transfers, an incomplete synchronous IN transfer interrupt (IISOIXFR) and an incomplete synchronous OUT transfer interrupt (IISOXFR) are generated to notify the application that there are synchronous IN/OUT commands that have been dropped.

### Selects the TRDT value of the OTG\_FS\_GUSBCFG register

The TRDT value of the OTG\_FS\_GUSBCFG register indicates, in units of PHY clock ticks, the time it takes the MAC to receive an IN command, acquire the FIFO status, and fetch the initial data from the PFC (Packet FIFO Controller). This time includes the delay in synchronizing the PHY and AHB clocks. The worst case is that the AHB clock frequency is the same as the PHY clock frequency, in which case this delay is 5 clock cycles.

Once the MAC receives the IN command, this information (received command) is synchronized to the AHB clock by the PFC (which is driven by the AHB clock.) The PFC reads the data from the SPRAM and writes it to the buffer of the dual clock source. the MAC then reads the data from the buffer (4 deep).

If the AHB clock frequency is higher than the PHY, a smaller TRDT value (OTG\_FS\_GUSBCFG) can be selected. There are the following signals in the figure below:

- tkn\_rcvd: command received message from MAC to PFC
- dynced\_tkn\_rcvd: double synchronized tkn\_rcvd signal from PCLK to HCLK
- spr\_read: read SPRAM
- spr\_addr: SPRAM addressing
- spr\_rdata: read data from SPRAM
- srcbuf\_push: feed to source buffer
- srcbuf\_rdata: reads data from the source buffer. the MAC detects the data. The application can use the following formula to calculate the TRDT value:

$4 \times \text{AHB clocks} + 1 \text{ PHY clock} = (2 \text{ clocks synchronized} + 1 \text{ clock for memory addressing} + 1 \text{ clock to fetch data from synchronized RAM}) + 1 \text{ PHY clock (the next PHY clock MAC can sample 2 clocks of FIFO output).}$

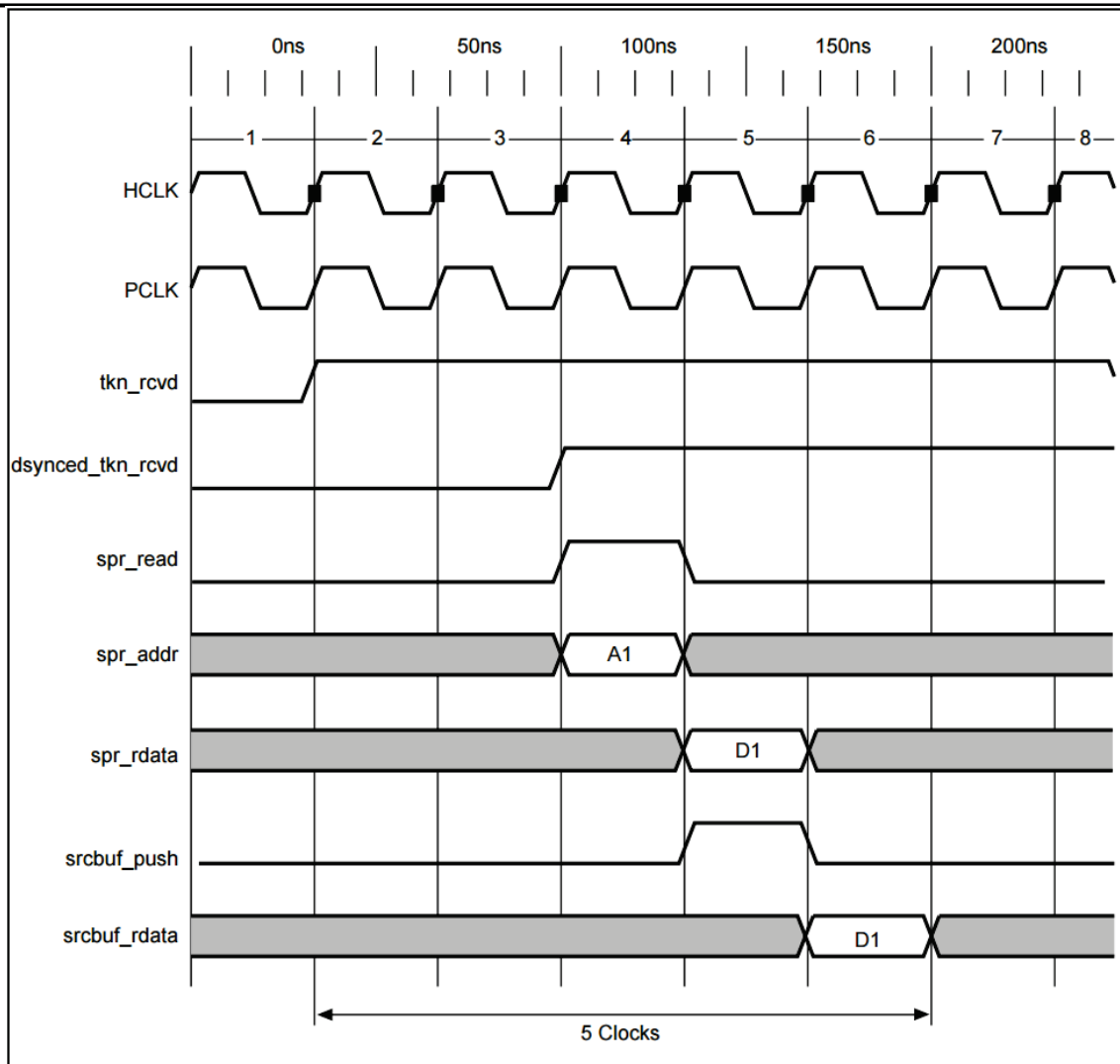


Figure291 TRDT Maximum Timing of the Case

## 29.17.8 OTG Programming Rules

The OTG\_FS controller can be used to design an OTG device that supports the HNP and SRP protocols. When the controller finds a Class A plug inserted, the controller performs a Class A device operation and when a Class B plug is inserted, a Class B operation is performed. In host mode, the OTG\_FS controller can turn off the VBUS to save power consumption. The SRP protocol is used when a Class B device requests a Class A device to turn on power to the VBUS. The device must generate pulses on both the control data line and the VBUS line, but the host can recognize the pulse signal of only one of them as the SRP signal. The HNP protocol is used for the Class B device to negotiate and switch to the host role, and after the negotiation, the Class B device can also hang up the bus and return to the device role.

### Session Request Protocol for Class A Devices

The application must set the SRP enable bit in the controller's USB configuration register, which will enable the OTG\_FS controller to recognize SRP requests in Class A device mode.

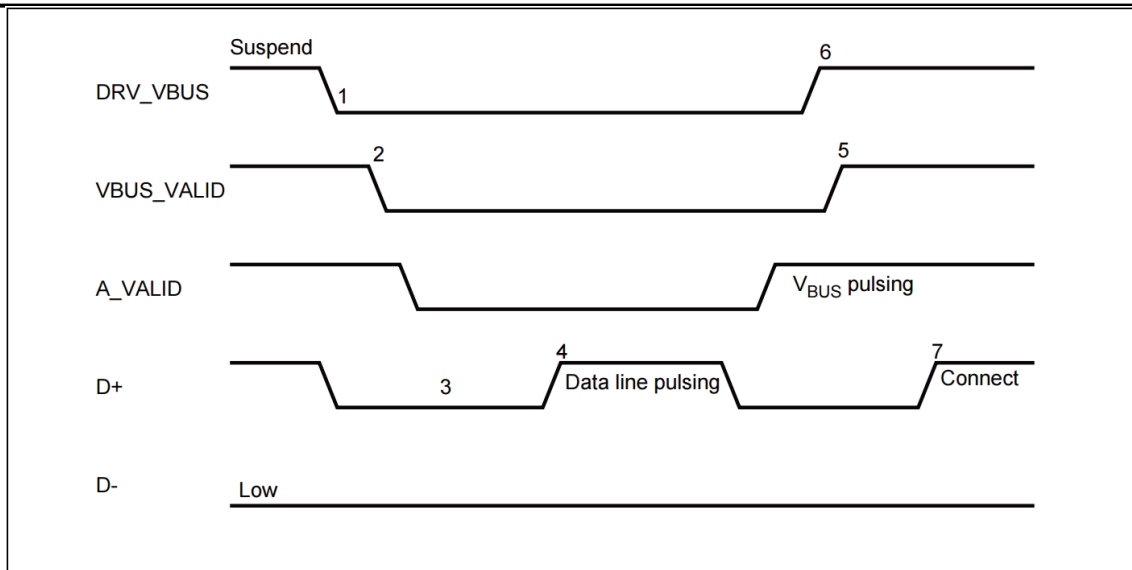


Figure292 SRP for the A device

DRV\_VBUS: VBUS drive signal sent to PHY

VBUS\_VALID: VBUS valid signal of PHY

A\_VALID: VBUS level signal sent to the PHY's Class A device

D+: Positive data line

D-: Reverse data line

1. To save power consumption, the application program needs to set the Port Hookup bit and Port Power bit of the Host Port Control and Status Register to hook up the bus and turn off power to the port when the bus is idle.
2. The PHY pulls down the VBUS\_VALID signal to indicate that port power is off.
3. When the VBUS bus power supply is off, the device must detect the SE0 signal for at least 2ms to initiate the SRP request.
4. In order to initiate the SRP request, the device needs to open the pull-up resistor on the data line for 5 to 10 ms. the OTG\_FS controller will detect the pulse signal on the data line.
5. The device needs to drive VBUS to provide a VBUS pulse that exceeds the Class A device session valid level (minimum 2.0 V.) The OTG\_FS controller will notify the application program with an interrupt that an SRP request has been detected and set the session request detection bit of the Controller's Global Interrupt Status Register (SRQINT bit of OTG\_FS\_GINTSTS).
6. The application needs to respond to the session request to detect the interrupt and turn on power to the port by writing the port power bit in the host port control and status register. the PHY pulls up the VBUS\_VALID signal to indicate that power to the port has been turned on.
7. When power is restored to the USB bus and the device is connected, the SRP process ends.

### Session Request Protocol for Class B Devices

The application program must set the SRP enable bit of the controller's USB configuration register. This bit will enable the SRP function when the OTG\_FS controller is used as a Class B device. The SRP function allows the OTG\_FS controller to initiate a session request to the host.

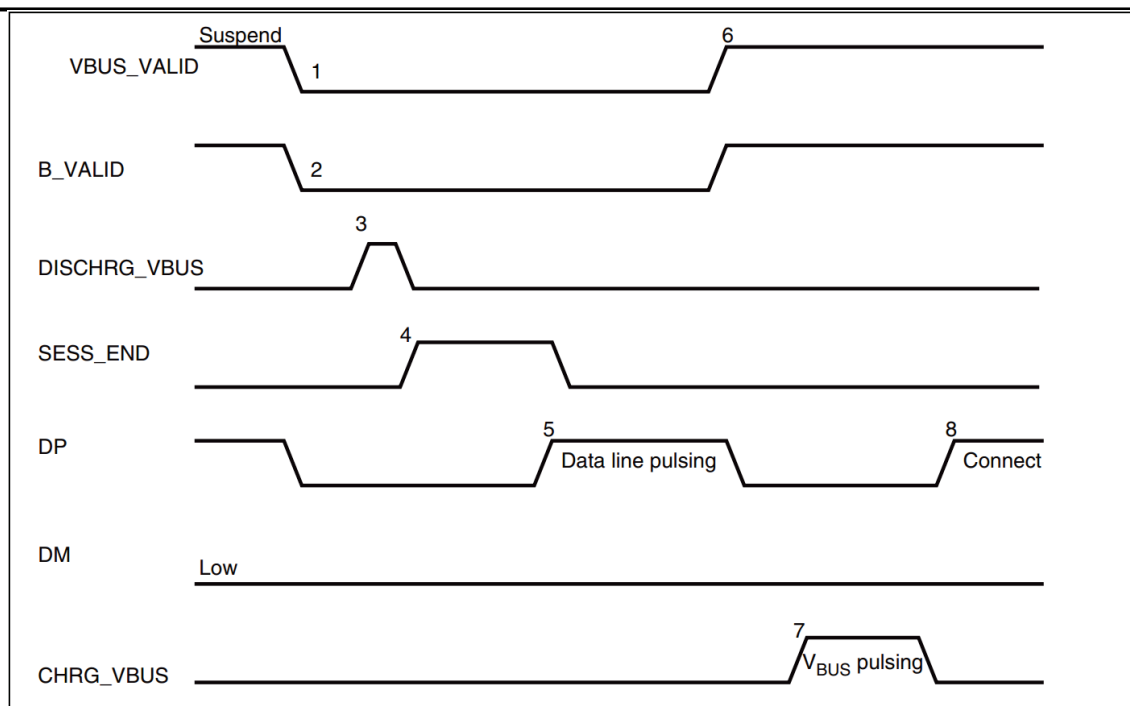


Figure293 Class B Device SRP

VBUS\_VALID: VBUS valid signal from PHY

B\_VALID: Valid signal for Class B devices that notify PHYs

DISCHRG\_VBUS: Discharge signal to notify the PHY

SESS\_END: end-of-session signal to notify the PHY

CHRG\_VBUS: Signal to notify the PHY of the drive VBUS

DP: positive data line

DM: Reversed data lines

1. To save power consumption, the host can hang and turn off port power when the bus is idle.

The OTG\_FS controller will set the early pending bit of the controller interrupt register after the bus has been idle for 3ms. Immediately following this, the OTG\_FS controller will set the USB pending bit of the controller interrupt register.

The OTG\_FS controller will notify the PHY to stop powering the VBUS.

2. The PHY will indicate the session stop signal sent to the device. This is a prerequisite for initiating an SRP request. the OTG\_FS controller needs to maintain the SE0 state for at least 2 ms before initiating an SRP request.

For USB 1.1 full-speed transceivers, applications need to wait for the BSVLD bit (OTG\_FS\_GOTGCTL) to be reset after the This wait time is determined by the transceiver vendor and varies from transceiver to transceiver.

3. The application program initiates the SRP request by writing the session request bit of the OTG control and status registers. the OTG\_FS controller will output pulses for the data lines first and then for VBUS.
4. The host can recognize an SRP request to turn on power to the VBUS based on pulses from the VBUS or data lines. the PHY will indicate that the VBUS has been repowered.
5. The OTG\_FS controller generates VBUS pulses.

The host will turn on the power to VBUS and start a new session, which indicates that the SRP request was successful. The OTG\_FS controller will set the "Session Request Success Status Change Bit" in the OTG Interrupt Status Register to notify the application program. The application program can notify the application program by reading the The application program can obtain this information by reading the Session Request Success bit in the OTG Control and Status Register.

6. When the USB is repowered and the OTG\_FS controller is connected, the SRP request is successfully completed.

## Host Negotiation Protocol for Class A Devices



The HNP protocol is used to switch host roles between Class A and Class B devices. The application needs to set the HNP enable bit of the controller's USB configuration register to enable the HNP function when the OTG\_FS controller is used as a Class A device.

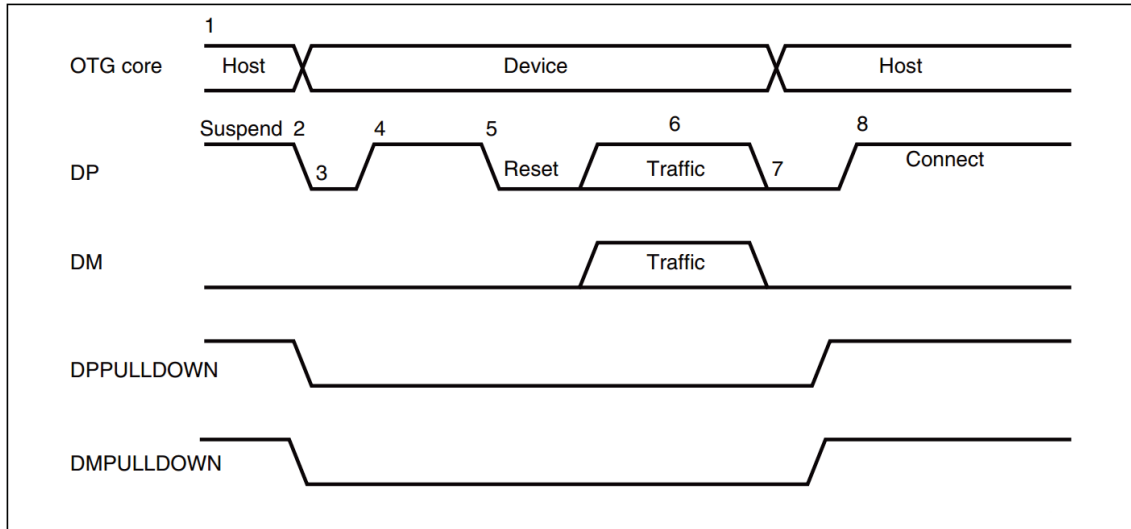


Figure294 Class A equipment HNP

DPPULLDOWN: Signal sent from the controller to the PHY to enable/disable the pull-down of the DP line inside the PHY  
DMPULLDOWN: Signal sent from the controller to the PHY to enable/disable the pull-down of the DM line inside the PHY

1. The OTG\_FS controller sends the SetFeatureb\_hnp\_enable command to the Class B device to enable the HNP functionality. The Class B device responds to the command with an ACK to indicate that the Class B device supports the HNP protocol. The application needs to set the HNP enable bit in the OTG control and status registers to inform the OTG\_FS controller that the connected Class B device supports the HNP protocol.
2. When the application program no longer needs to use the bus, it needs to set the port suspend bit in the host port control and status registers to suspend the bus.
3. When a Class B device detects the USB hang signal, it can disconnect and be instructed to start initiating HNP requests. a Class B device only needs to initiate an HNP request when it needs to perform a host role, otherwise it keeps the bus in the hang state.

The OTG\_FS controller sets the Host Negotiation Detected Interrupt bit of the OTG Interrupt Status Register to notify the application program that it has detected an HNP request. that an HNP request has been detected.

The OTG\_FS controller deasserts the pull-down of the DM and DP lines to enforce device roles. The PHY enables the pull-up of the DP line to indicate Class B device access.

The application program must read the current mode bits of the OTG control and status registers to know the current operating mode.

4. The Class B device detects access to the device, initiates a USB reset, and enumerates the OTG\_FS control.
5. Class B devices always perform the host role, initiate communication, and hang up the bus when they have completed their operations.

The OTG\_FS controller will set the early pending bit of the controller interrupt register after detecting that the bus has been idle for more than 3ms. This is immediately followed by setting the USB pending bit of the controller interrupt register.

6. In negotiation mode, the OTG\_FS controller detects a bus hang, disconnects the device, and switches back to the host role. the OTG\_FS controller enables the pull-down of the DM and DP lines inside the PHY, indicating that the host role is restarted.
7. The OTG\_FS controller sets the Access ID line status of the OTG Interrupt Status Register to change interrupts. The application program must read the Access ID Line Status bit of the OTG Control and Status Register to determine if the OTG\_FS controller is operating in Class A device mode. This marks the completion of the HNP protocol. The application program must read the Current Mode bit of the OTG Look Oh You Look Inform and Status Registers to determine if it is operating in host mode.
8. Class B devices are re-accessed to complete the HNP protocol.

## Host Negotiation Protocol for Class B Devices

The HNP protocol is used to switch host roles between Class A and Class B devices. The application program must set the HNP enable bit of the controller's USB configuration register to notify the OTG\_FS controller to enable the HNP function when acting as a Class B device.

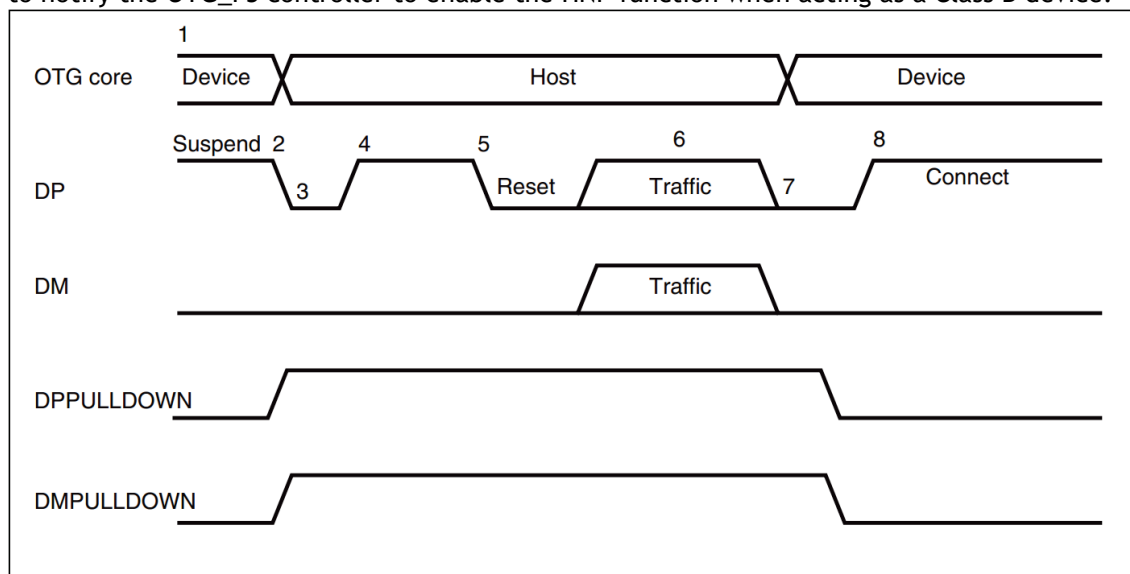


Figure295 Class B equipment HNP

DPPULLDOWN: Pull-down enable/disable signal from the controller to the PHY for the DP line inside the PHY. DMPULLDOWN: Pull-down enable/disable signal from the controller to the PHY for the DM line inside the PHY.

1. Class A devices send the SetFeatureb\_hnp\_enable command to enable the HNP protocol. the OTG\_FS controller needs to reply with an ACK response to inform that the HNP protocol is supported. Devices for which the application must set the OTG control and status registers

HNP enable bit to indicate support for the HNP protocol.

The application program needs to set the HNP request bit of the OTG control and status register to notify the OTG\_FS controller to initiate the HNP request.

2. Class A devices hang the bus when they stop using it. Class A devices set the port hang bit of the host port control and status registers to hang the bus.

The OTG\_FS controller will set the early pending bit of the controller interrupt register when it checks for more than 3ms of bus idle, immediately followed by setting the USB pending bit of the controller interrupt register.

The OTG\_FS controller will disconnect and the Class A device will detect the SE0 state on the bus, indicating the start of the HNP protocol. The OTG\_FS controller will enable the DP and DM pull-downs, indicating the start of the host role.

Class A devices will enable the pull-up resistor on the DP line after the SE0 state has been detected for more than 3 ms. the OTG\_FS controller will assume that that a device is inserted.

The OTG\_FS controller sets the "Host Negotiation Successful Status Change" interrupt in the OTG Interrupt Status Register to indicate the start of the HNP protocol. The application program must read the Host Negotiation Success bit in the OTG Control and Status Register to obtain information on whether the host negotiation was successful or not. The application must read the Host Negotiation Success bit in the OTG Control and Status Register to obtain information on whether host negotiation was successful. The application must read the Current Mode bit in the Controller Interrupt Register to determine if the controller is operating in the host mode.

3. The application sets the reset bit (PRST bit of OTG\_FS\_HPRT) and the OTG\_FS controller performs a USB reset operation and enumerates the Class A devices.
4. The OTG\_FS controller always performs the host role, initiates communication, and at the end of the demand, hangs the bus by writing the port hang bit of the host port control and status registers.
5. In negotiation mode, when a Class A device detects a bus hang, it will disconnect and switch back to the host role. the OTG\_FS controller will cancel the DP and DM drop-down instructions to re-enforce the device role.



- 
6. The application program must read the current mode bit of the controller interrupt register to determine if it is operating in host mode.
  7. The OTG\_FS controller restores the connection and ends the HNP protocol.

## 30 Device Electronic Signature

### 30.1 Memory Capacity Register

#### 30.1.1 Flash Capacity Register

Base address: 0x1FFF F7E0

Read-only, its contents are written at the factory

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

Bit	notation	clarification
15:0	F_SIZE	F_SIZE: Flash memory capacity Indicates the flash memory capacity in the product in K bytes. Example: 0x0080 = 128K bytes

### 30.2 Product Unique Identification Register (96 bits)

The unique identification of the product is ideally suited:

- Used as a serial number (e.g. USB character serial number or other terminal applications)
- Used as a password, this unique identifier is used in conjunction with software encryption and decryption algorithms when writing flash memory to improve the security of the code within the flash memory storage.
- Used to activate bootstrap processes with safety mechanisms

The reference number provided by the 96-digit Product Unique Identifier is unique to any W55MH32 microcontroller under any circumstances. This identification cannot be modified by the user under any circumstances.

This 96-bit product unique identifier can be read in bytes (8-bit), half-words (16-bit) or full-words (32-bit), depending on the user's usage.

Base address: 0x1FFF F7E8

Address offset: 0x00

Read-only, its value is written at the factory

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

Bit	notation	clarification
31:16	U_ID[15:0]	U_ID[15:0]: unique identity flag bits 15:0

Address offset: 0x02

Read-only, its value is written at the factory

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
U_ID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit	notation	clarification
31:16	U_ID[31:16]	U_ID[31:16]: unique identifier 31:16 bits The value of this field is also reserved for other future functions.

Address offset: 0x04

Read-only, its value is written at the factory

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
U_ID[63:48]															
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
U_ID[47:32]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit	notation	clarification
31:16	U_ID[63:32]	U_ID[63:32]: unique identifier 31:16 bits
Address offset: 0x08		
Read-only, its value is written at the factory		
31	30	29
28	27	26
25	24	23
22	21	20
19	18	17
16	U_ID[95:80]	
r	r	r
r	r	r
r	r	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	U_ID[79:64]	
r	r	r
r	r	r
r	r	r
r	r	r
r	r	r
r	r	r
r	r	r
r	r	r
Bit	notation	clarification
31:16	U_ID[95:64]	U_ID[95:64]: Unique Identifier 95:64 bits

## 31 Debugging Support (DBG)

### 31.1 General Situation

The W55MH32 uses the Cortex™-M3 core, which includes a hardware debug module to support complex debugging operations. The hardware debug module allows the kernel to stop when fetching instructions (instruction breakpoints) or accessing data (data breakpoints). When the kernel is stopped, both the internal state of the kernel and the external state of the system are queried. Upon completion of the query, the kernel and peripherals can be recovered and the program will continue to execute.

When the W55MH32 microcontroller is connected to the debugger and debugging is started, the debugger will use the kernel's hardware debug module to perform debugging operations.

Two debugging interfaces are supported:

- serial interface
- JTAG Debug Interface

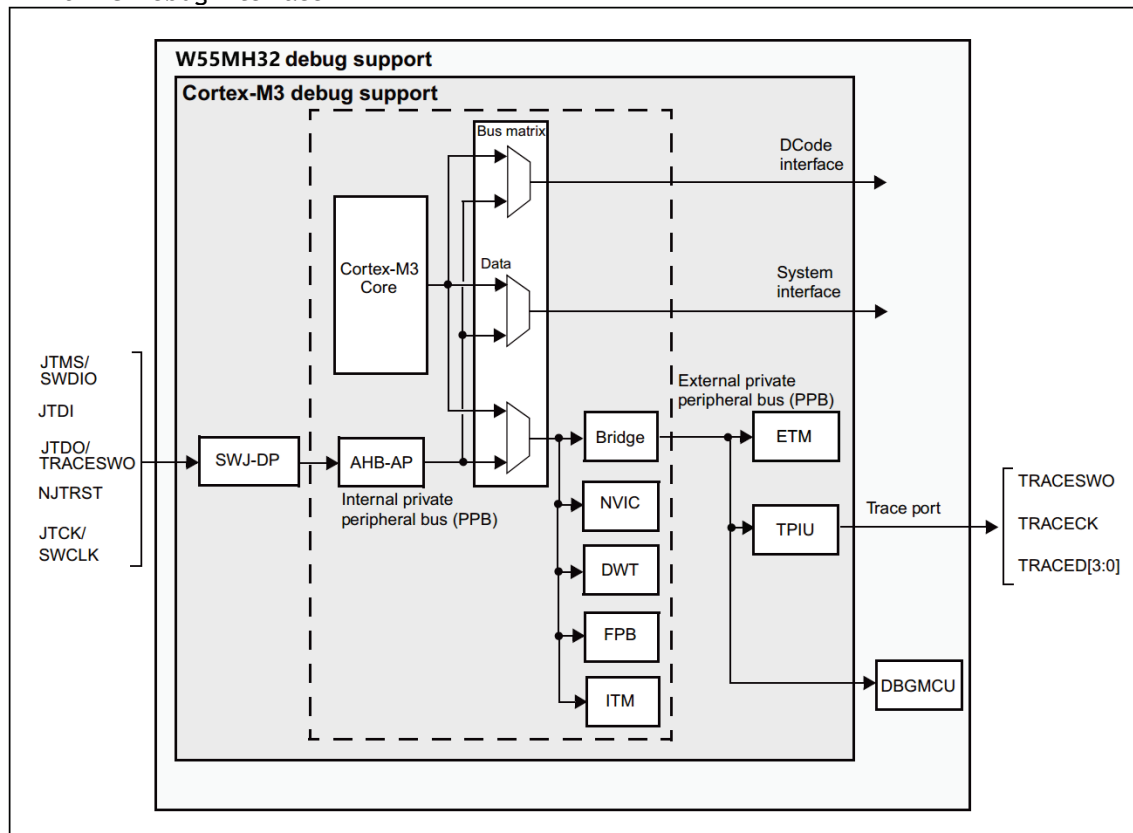


Figure 296 Debugging Block Diagram for W55MH32 level and Cortex™-M3 level

**Notes:** The hardware debugging module included with the Cortex™-M3 core is a subset of the ARM CoreSight development toolset.

The ARM Cortex™-M3 core provides integrated on-chip debugging capabilities. It consists of the following components:

- SWJ-DP: Serial/JTAG Debug Port
- AHP-AP: AHB access port
- ITM: Implementation Tracking Module
- FPB: Flash Instruction Breakpoint
- DWT: Data Trigger
- TPIU: Trace unit interface (only supported by chips in larger packages)
- ETM: Embedded Trace Microcell (pins supporting this feature are available on larger packages) dedicated to the debugging features of the W55MH32
- Flexible debug pin assignment
- MCU debug box (support low power mode, control peripheral clock, etc.)

Notes: For more information on the debugging features of the ARM Cortex™ -M3 core, refer to the Cortex™ -M3 (r1p1 Edition) Technical Reference Manual (TRM) and CoreSight Development Toolset (r1p0 Edition) TRM.

## 31.2 ARM References

- Cortex™ -M3 (r1p1 Edition) Technical Reference Manual (TRM)
- ARM Debug Interface V5
- ARM CoreSight Development Toolset (r1p0 Edition) Technical Reference Manual

## 31.3 SWJ Debug Port (Serial Wire and JTAG)

The W55MH32 core integrates a serial/JTAG debug interface (SWJ-DP). This is the standard ARM CoreSight debug interface and includes the JTAG-DP interface (5 pins) and the SW-DP interface (2 pins).

- The JTAG Debug Interface (JTAG-DP) provides a 5-pin standard JTAG interface for the AHP-AP module.
- The Serial Debug Interface (SW-DP) provides a 2-pin (clock + data) interface to the AHP-AP module.

In the SWJ-DP interface, 2 pins of the SW-DP interface and some of the 5 pins of the JTAG interface are multiplexed.

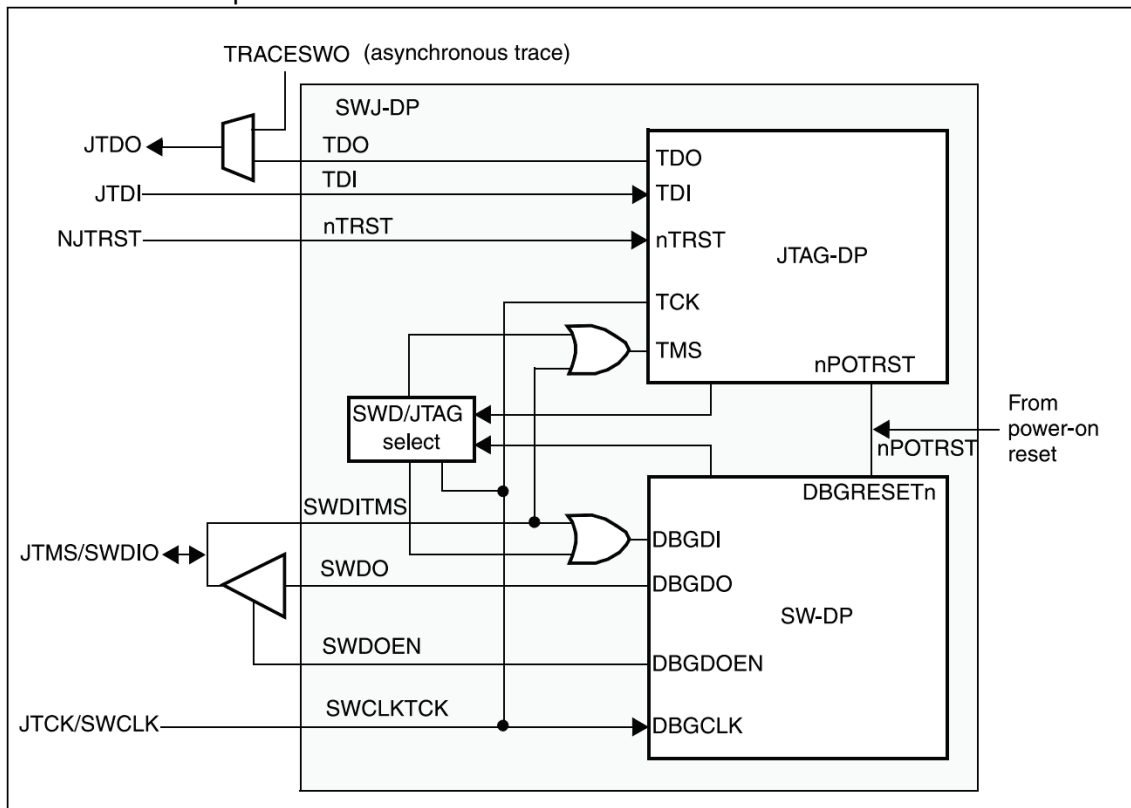


Figure 297 SWJ Debug Port

The diagram above shows that the asynchronous trace output pin (TRACE SWO) and TDO are multiplexed, so the asynchronous trace function can only be implemented on the SW-DP debug interface, not on the JTAG-DP debug interface.

### 31.3.1 Mechanism for JTAG-DP and SW-DP Switching

The JTAG debug interface is the default debug interface.

If the debugger wants to switch to SW-DP, it must output a specified JTAG sequence on the TMS/TCK (mapped to SWDIO and SWCLK, respectively) that disables JTAG-DP and activates SW-DP. This method can be used only through the SWCLK and the SWDIO two pins to activate the SW-DP interface. The specified sequence is:

1. Outputs TMS (SWDIO) = 1 signals for more than 50 TCK cycles

2. Output 16 TMS (SWDIO) signals 011110011111100111 (MSB)
3. Outputs TMS (SWDIO) = 1 signals for more than 50 TCK cycles

## 31.4 Pinouts and Debug Port Pins

### 31.4.1 SWJ Debug Port Pin

The W55MH32's five common I/O ports can be used as SWJ-DP interface pins. These pins are present in all packages.

Table161 SWJ Debug Port Pins

SWJ-DP Port Pin Name	JTAG Debug Interface		SW debugging interface		Pin Assignment
	typology	descriptive	typology	debugging function	
JTMS/SWDIO	importation	JTAG mode selection	Input/Output	Serial data input/output	PA13
JTCK/SWCLK	importation	JTAG Clock	importation	serial clock	PA14
JTDI	importation	JTAG data input	--	--	PA15
JTDO/TRACESWO	exports	JTAG data output	--	TRACESWO signal when tracking	PB3
JNTRST	importation	JTAG module reset	--	--	PB4

### 31.4.2 Flexible SWJ-DP Pin Assignment

Immediately after reset (SYSRESETn or PORESETn), all five pins belonging to the SWJ-DP are initialized as dedicated pins that can be used by the debugger (note that the trace output pin is not initialized unless the debugger defines this pin).

However, the W55MH32 microcontroller can disable the functionality of some or all of the pins of the SWJ-DP interface with the Multiplexed Remapping and Debugging I/O Configuration Register (AFIO\_MAPR) register (see section7.4.2 ), and these dedicated pins are released for use as general I/O ports. This register is mapped to the APB bridge connected to the Cortex™-M3 system bus. The setting of this register will be done by user code and not by the debugger. Three control bits are used to configure the pins of the SWJ-DP interface, and these three bits are reset at system reset.

- AFIO\_MAPR (address is 0x4001 0004)
  - Read: APB, no wait state
  - Write: APB, if the write buffer of the AHB-APB bridge is full, a wait state

Bits 26:24 = SWJ\_CFG[2:0]

Set and reset by software

These 3 bits are used to set the number of dedicated pins assigned to the SWJ debug interface, with the goal of freeing up as many pins as possible for use as general I/O ports when using a different debug interface.

The initial value after reset is 000 (all pins are set to JTAG-DP interface-specific pins), and only one of the three bits can be set at the same time (setting more than one bit at the same time is prohibited).

Table162 Flexible SWJ\_DP Pin Assignments

SWJ-CFG [2:0]	Pins configured for debugging	I/O port assignment for SWJ interface				
		PA13/ JTMS/ SWDIO	PA14/ JTCK/ SWCLK	PA15/ JTDI	PB3/ JTDO	PB4/ JNTRST
000	All SWJ pins (JTAG-DP+SW-DP) reset state	categorical	categorical	categorical	categorical	categorical
001	All SWJ pins (JTAG-DP+SW-DP) In addition to the JNTRST pin	categorical	categorical	categorical	categorical	liberate (a prisoner)
010	JTAG-DP interface disabled, SW-DP interface allowed	categorical	categorical	liberate (a prisoner)		
100	Both the JTAG-DP interface and the SW- DP interface are disabled	liberate (a prisoner)				

(sth. or sb) else	prohibited	
----------------------	------------	--

**Notes:** When the APB bridge's write buffer is full, one more APB cycle is required when writing the AFIO\_MAPR register. This is because the release of the JTAGSW pin requires 2 APB cycles to ensure smooth nTRST and TCK signals into the core.

- Cycle 1: Input 1/O JTAGSW signals to the core (nTRST, TDI and TMS are 1, TCK is 0).
- Cycle 2: The GPI/O controller gets control signals for the SWJTAG I/O pins (e.g., control of direction, pull-up/down, Schmitt trigger, etc.).

### 31.4.3 Internal Pull-ups and Pull-downs on the JTAG Pin

It is essential to ensure that the JTAG input pins are not dangling, as they are directly connected to the D-trigger controlling the debug mode. Special attention must be paid to the SWCLK/TCK pins as they are directly connected to the clock side of some of the D flip-flops.

To avoid any uncontrolled I/O levels, the W55MH32 embeds internal pull-ups and pull-downs on the JTAG input pins.

- JINTRST: internal pull-up
- JTDI: internal pull-up
- JTMS/SWDIO: Internal pull-ups
- TCK/SWCLK: internal pull-downs

Once the JTAG I/O is released by the user code, the GPIO controller gains control again. The state of these I/O ports will be restored to the state they were in at reset.

- JNTRST: Input with pull-up
- JTDI: Inputs with pull-ups
- JTMS/SWDIO: Inputs with pull-ups
- JICK/SWCLK: Inputs with drop-downs
- JTDO: Floating Input

The software can use these I/O ports as normal I/O ports.

**Notes:** The JTAG IEEE standard recommends pull-ups for TDI, TMS, and nTRST, and has no specific recommendation for TCK. However, there is no specific recommendation for TCK in the W55MH32, the JTCK pin has a pull-down.

Embedded pull-ups and pull-downs eliminate the need for external resistors on the chip.

### 31.4.4 Utilizes the Serial Interface and Releases the Unused Debug Pins as Normal I/O Ports.

In order to utilize the serial debugging interface to free up some of the common I/O ports, the user software must set SWJ\_CFG=010 after a reset, thus freeing up PA15, PB3, and PB4 to be used as common I/O ports.

During debugging, the debugger performs the following operations:

- At system reset, all SWJ pins are assigned as dedicated pins (JTAG-DP+SW-DP).
- In the system reset state, the debugger sends the specified JTAG sequence to switch from JTAG-DP to SW-DP.
- Still in the system reset state, the debugger sets a breakpoint at the reset address
- The reset signal is released and the kernel stops at the reset address.
- From here on, all debug communication will use the SW-DP interface, and other JTAG pins can be reassigned by user code to normal I/O ports.

**Notes:** For user software design, care should be taken:

After a reset, these dedicated pins remain in the pull-up input (nTRST, TMS, TDI), pull-down input (TCK), and output (TDO) states for a period of time until the user code releases these pins.

When these pins are configured as dedicated pins (JTAG or SW or TRACE), modifying the corresponding general I/O port configuration registers is not effective.

## 31.5 W55MH32JTAG TAP Connection

The W55MH32 microcontroller has two internal JTAG TAPs in series. the boundary-scan TAP is dedicated to testing (IR register is 5 bits wide) and the Cortex™-M3 TAP (IR register is 4 bits wide).

In order to access the Cortex™-M3 TAP to debug the chip, it is necessary to:

1. First, the BYPASS instruction must be shifted into TMCTAP.
2. Second, when shifting the input IR, each scan chain contains 9 bits (=5+4), and for unused TAPs, the BYPASS command must be entered
3. When shifting the input data, the unused TAP is in BYPASS mode, so an extra bit is added to the data scan chain.

**Notes:** *IMPORTANT: Once the serial debug interface is selected using the specified JTAG sequence, TMC TAP is automatically disabled (JTMS is forced high).*

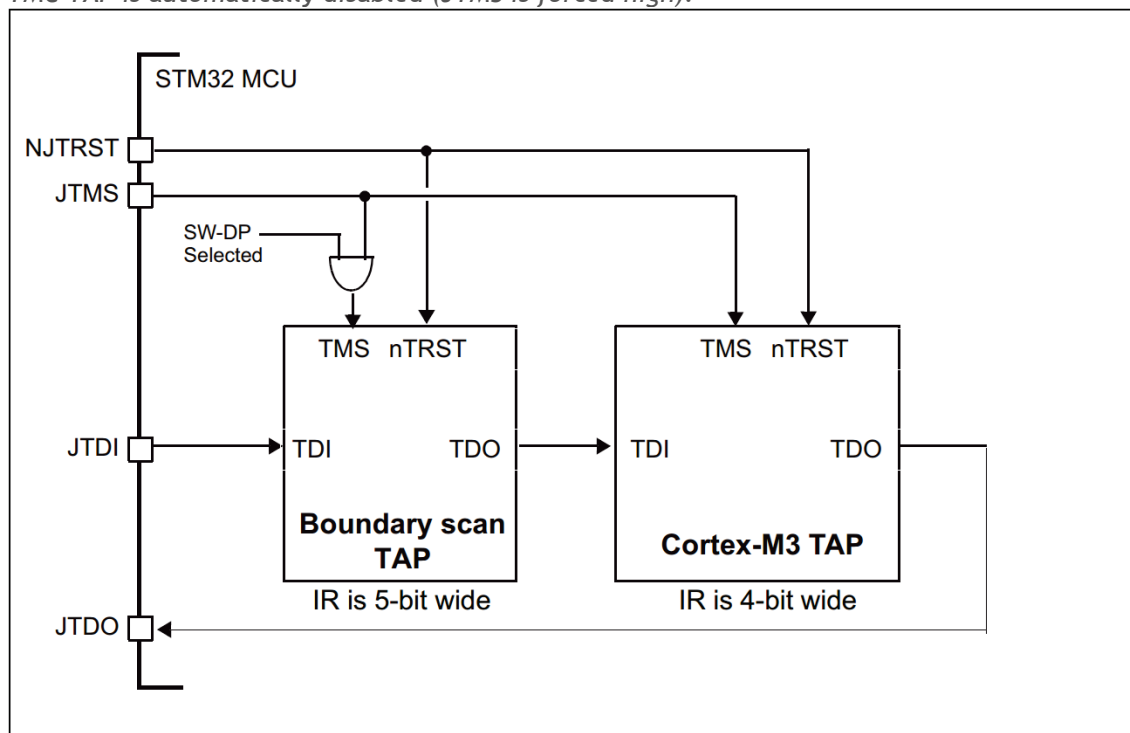


Figure298 JTAG TAP connection

## 31.6 ID Codes and Locking Mechanisms

There are multiple ID codes within the W55MH32 microcontroller. It is strongly recommended that the tool designer use the MCU DEVICE ID mapped on the external PPB memory at address 0xE004 2000 to lock the debugger.

### 31.6.1 Microcontroller Device ID Code

The microcontroller W55MH32 contains an MCUID code. This ID defines the MCU part number and silicon version. It is a component of DBG\_MCU and is mapped to the external PPB bus (see section 31.16). This code can be accessed using the JTAG debug port (4 to 5 pins) or the SW debug port (2 pins) or through user code. This code can be accessed even when the MCU is in system reset.

DBGMCU\_IDCODE

Address: 0xE004 2000 supports 32-bit access only

Read-only= 0XXXXX X410, where X is a bit with indeterminate content

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



Bit	notation	clarification
31:16	REV_ID[15:0]	REV_ID[15:0]: version identification This field identifies the version of the product 0x1000 = version A 0x1001 = version Z
15:12	Reserved	Reserved
11:0	DEV_ID[11:0]	DEV_ID[11:0]: Device Identification This section indicates the device code. The device code is 0x414;

### 31.6.2 Boundary Scan TAP

JTAGID code

W55MH32 boundary-scan TAP with integrated JTAGID code: 0x0641 4041 = version A

### 31.6.3 Cortex-M3 TAP

The ARM Cortex-M3 TAP has a JTAG ID code. This ID code is the ARM default and has not been modified and can only be accessed through the JTAG debug port. This code is 0x3BA0 0477 (corresponding to Cortex-M3 r1p1).

The debugger/programming tool should only recognize the chip by DEV\_ID(11:0).

### 31.6.4 Cortex-M3 JEDEC-106 ID Code

ARM's Cortex-M3 has a JEDEC-106ID code. It is located in the 4KB ROM table mapped to the internal PPB bus address 0xE00F F000\_0xE00F FFFF.

## 31.7 JTAG Debug Port

The standard JTAG state machine is implemented with a 4-bit instruction register (IR) and 5 data registers (see Cortex-M3 r1p1 Technical Reference Manual for details).

Table163 JTAG Debug Port Data Registers

IR(3:0)	data register	descriptive
1111	BYPASS [1 bit]	
1110	IDCODE [32 bits]	ID Code Register 0x3BA0 0477 (ARM Cortex-M3 r1p1-01rel0 ID code)
1010	DPACC [32 bits]	Debug Interface Register Initializes the debug port and allows access to the debug interface registers -When entering data: Bits34:3=DATA[31:0]: 32-bit data bits corresponding to the write operation Bits2:1=A[3:2]: 2-bit address values of the debug interface registers Bit0=RnW: Read operation (1) or write operation (0) -When outputting data: Bits34:3=DATA[31:0]: 32-bit data result of previous read operation Bits2:0=ACK[2:0]: 3-bit answer 010 = Success/Failure 001 = Waiting Other = Undefined For the definition of A(3:2), please refer to 0
1011	APACC [35 bits]	Access Interface Registers Initializes the access interface and allows access to the access interface registers -When entering data: Bits34:3=DATA[31:0]: 32 data bits corresponding to write operations Bits2:1=A[3:2]: 2-bit address (partial address of AP register) Bit0=RnW: read operation (1) or write operation (0) -When outputting data: Bits34:3=DATA[31:0]: 32-bit data result of previous read operation Bits2:0=ACK[2:0]: 3-bit answer 010 = Success/Failure 001 = Waiting Other = Undefined Refer to the AHB-AP section for the AP registers, which are addressed by A[3:2] -Shift values A[3:2] -Current value of the DPSELECT register
1000	ABORT [35 bits]	abort register -Bits31:1 undefined -Bit0 = DAPABORT: Write 1 to generate a DAP abort

Table164 32-bit Debug Interface Register Addresses Defined by A[3:2]

address	A(3:2) value	descriptive
0x0	00	undefined
0x4	01	DPCTRL/STAT registers -Request a system or debug power-up operation -Configure the mode of operation for AP access -Controls comparison, calibration operations -Read some status bits (overflow, power-up response)
0x8	10	DPSELECT register Used to select the current access port and valid 4-word long register window -Bits31:24: APSEL selects the current AP -Bits23:8: undefined -Bits7:4: APBANKSEL: Selects a 4-word-length register window on the current AP -Bits3:0: undefined
0xC	11	DPDRDBUFF register: used to enable the debugger to get the final result of the previous operation (without having to request a new JTAG-DP operation)

## 31.8 SW Debug Port

### 31.8.1 Introduction to SW Protocol

This synchronous serial protocol uses 2 pins:

- SWCLK: clock signal from host to target
- SWDIO: Bidirectional data signal

The protocol allows reading and writing of 2 register sets (DPACC and APACC register sets). Data bits are transferred on an LSB basis.

Since SWDIO is a bi-directional port, a pull-up is required on this pin (ARM recommends a 100KΩ resistor).

By protocol a transition time is inserted each time the SWDIO direction changes. Neither the host nor the target drives this signal line during this period. The default value for the transition time is 1 bit, but can be adjusted by configuring the SWCLK frequency.

### 31.8.2 SW Protocol Sequence

Each sequence consists of 3 phases:

1. Host send packet request (8 bits)
2. Target sends acknowledgement response (3 bits)
3. Host or target send data (33 bits)

Table165 Request Packet (8 bits)

byte (computing)	name (of a thing)	descriptive
0	originate	Must be 1
1	APnDP	0: Access to DP1: Access to AP
2	RnW	0: Write request 1: Read request
4:3	A (3:2)	DP or AP register address (refer to 0)
5	Parity	Check digit of the preceding bit
6	Stop	0
7	Park	Cannot be driven by host, target always reads 1 due to pull-ups

Refer to the Cortex-M3 r1p1 Technical Reference Manual for details on DPACC and APACC register descriptions. Packet requests are always followed by a (default 1-bit) transition time, when neither the host nor the target drives the line.

Table166 ACK Definitions (3 bits)

byte (computing)	name (of a thing)	descriptive
0..2	ACK	001: Failure 010: Waiting 100: Success

When the ACK is a failure or wait, or an ACK replying to a read operation, this ACK is followed by a transition time.

Table167 Transmission data (33 bits)

byte (computing)	name (of a thing)	descriptive
0..31	WDATA/RDATA	Data written or read
32	Parity	Parity bit for 32-bit data

The read operation has a conversion time after the data transfer operation.

### 31.8.3 SW-DP State Machine (Reset, Idle States, IDcode)

The SW-DP state machine has an internal ID code used to identify the SW-DP, which adheres to the JEP-106 standard. This ID code is the ARM default code with a value of 0x1BA0 1477 (corresponding to Cortex-M3 r1p1).

*Notes: The SW-DP's state machine does not work until the debugger reads this ID code.*

- The SW-DP state machine will be in the RESET state, after a power-on reset, or after the DP has switched from JTAG to SWD, or has been high for more than 50 cycles.
- When the state machine is in the RESET state, the state machine will switch to the IDLE state if there are at least 2 cycles of low level.
- When the state machine is in the RESET state, it must first enter the IDLE state and perform an operation to read the DP-SWID register. Otherwise, the debugger can only get a failed ACK response when performing other transfers.

For more detailed SW-DP state machine information, please refer to the Cortex-M3 r1p1 Technical Reference Manual and Core Sight Design Kit r1p0 Technical Reference Manual.

### 31.8.4 DP and AP Read/Write Access

- There is no delay for read operations to the DP: the debugger will either get the data directly (if ACK = success), or wait (if ACK = wait).
- Read operations on the AP have a delay. That is, the result of the previous read operation can only be obtained at the next operation. If the next operation is not an access to the AP, the DP-RDBUFF register must be read to obtain the result of the previous read operation.
- The READOK flag bit in the DP-CTRL/STAT register is updated after each AP read operation and RDBUFF read operation to notify the debugger whether the AP read operation was successful.
- SW-DP has a write buffer (both DP and AP have write buffers), which allows write operations to still be accepted while other transfers are in progress. If the write buffer is full, the debugger will get a waiting ACK response. Read IDCODE register, read CTRL/STAT register and write ABORT register operations are still accepted when the write buffer is full.
- Due to the asynchronous nature of SWCLK and HCLK, 2 additional SWCLK cycles need to be inserted after the write operation (after the parity bit) to ensure that the internal write operation completes correctly. These two extra clock cycles need to be inserted when the line is low (in the IDLE state). This operation step is especially important when writing the CTRL/STAT register to make a power-up request, otherwise the next operation (one that is not valid until after the kernel is powered up) will be executed immediately, which will result in a failure.

### 31.8.5 SW-DP Register

When APnDP=0, these following registers can be accessed.

Table168 SW-DP registers

A (3:2)	Read/Write	SELECT registers CTRLSEL bit	Register	descriptive
00	phrase marked by pause		IDCODE	Fixed to 0x1BA0 1477 (for SW-DP identification).
00	write		ABORT	
01	Read/Write	0	DP-CTRL/STAT	-Requests a system or debug power-up operation; -Configure the mode of operation for AP access; -Controls comparison, calibration operations; -Read some status bits (overflow, power-up response).

01	Read/Write	1	WIRE CONTROL	Configure the serial communications physical layer protocol (e.g., conversion time length, etc.).
10	phrase marked by pause		READ RESEND	Allows data to be recovered from an erroneous debug transfer without repeating the initial AP transfer.
10	write		SELECT	Selects the current access port and the valid 4-word long register window.
11	Read/Write		READ BUFFER	This register is necessary because of the pass-through nature of AP accesses (the result of the current AP read operation is passed on during the next AP transfer). This register captures the data result of the last read operation from the AP, so the data can be obtained without having to start a new AP transfer.

### 31.8.6 SW-AP Register

The following registers can be accessed when APnDP=1. The access address of the AP registers consists of the following two parts:

- The value of A[3:2].
- Current value of the DPSELECT register

## 31.9 AHB-AP (AHB Access Port) Valid for Either JTAG-DP or SWDP

Function:

- System access is independent of processor state.
- Both JTAG-DP and SW-DP can access the AHB-AP.
- The AHB-AP is the AHB master device of the bus matrix. As such, it has access to all data buses (Dcode bus, System bus, internal and external PPB buses) with the exception of the ICode bus.
- Supports bit-addressable transfers
- AHB-AP transmission of bypassed FPBs

The 32-bit AHB-AP register is addressed 6-bits wide (up to 64 words or 256 bytes) and consists of the following parts:

- Bits [8:4] = bits [7:4] of the DPSELECT register APBANKSEL
- Bit bits [3:2] = A(3:2) in the 35-bit SW-DP packet request. the Cortex-M3's AHB-AP has nine 32-bit registers

Table169 Cortex-M3AHB-AP Registers

address offset	register name	descriptive
0x00	AHB-AP Control and Status Word	Configure the transmission characteristics of the AHB interface (length, address self-add mode, current transmission state, privileged mode, etc.).
0x04	AHB-AP Transfer Address	
0x0C	AHB-AP Data Read/Write	
0x10	AHB-AP Banked Data0	Direct access to 4 connected words without rewriting the access address.
0x14	AHB-AP Banked Data1	
0x18	AHB-AP Banked Data2	
0x1C	AHB-AP Banked Data3	
0xF8	AHB-AP Debug ROM Address	The base address of the debug interface.
0xFC	AHB-AP ID Register	

For more information, please refer to the Cortex-M3r1p1 Technical Reference Manual

## 31.10 Kernel Debugging

Kernel debugging can be performed by manipulating the kernel debug registers. Access to these registers is through the Advanced High Performance Bus (AHB-AP). The processor can access these registers directly through the internal private peripheral bus (PPB). It includes 4 registers.

Table170 Kernel Debug Registers

Register	descriptive
DHCSR	32-bit debug control and status registers This register provides kernel status information, allows the kernel to enter debug mode, and provides single-step functionality.
DCRSR	17-bit kernel register debug select registers This register selects the kernel registers that require read and write operations.
DCRDR	32-bit kernel registers debug data registers This register holds data that is read from or needs to be written to the kernel register selected by the DCRSR.
DEMCR	32-bit exception debug and monitor control registers This register provides vector transfer and monitor debug control functions. the TRCENA bit initiates the TRACE function.

Notes: **IMPORTANT:** These registers are not reset on system reset, only on power-on reset.

Refer to the Cortex-M3r1p1 Technical Reference Manual for more detailed information. In order to put the kernel into the debug state immediately after reset, it is necessary to:

- Enables bit 0 (VC\_CORRESET) of the Debug and Exception Monitor Control .
- Enables bit 0 (C\_DEBUGEN) of the Debug Halting Control and Status .

## 31.11 Debugger Host Connectivity under System Reset

The reset system of the W55MH32 microcontroller consists of the following reset sources:

- POR (Power-On Reset), initiates a reset at every power-up
- Internal Watchdog Reset
- software reset
- external reset (computing)

The Cortex-M3 distinguishes between a reset of the debug section (typically PORRESETn) and other resets (SYSRESETn). Thus, when the kernel is in system reset, the debugger can connect to the kernel, configure the kernel debug registers, enable the debug allow bit, and operate in such a way that the kernel enters the debug state immediately upon release of the system reset without executing any instructions. Similarly, debugging features can be configured while the kernel is in the reset state.

Notes: *It is strongly recommended that the debugger connects to the kernel during a system reset (with a breakpoint at the reset vector).*

## 31.12 FPB (Flash Patch Breakpoint)

FPB unit:

- Implementing hardware breakpoints
- Replaces code and data in the code area with code and data in the system area. This feature can be used to correct software errors in the code area.

The software patch function and the hardware breakpoint function cannot be used at the same time.FPB consists of the following parts:

- 2 content comparators to compare the content obtained in the code area and remap it to the relevant address in the system area.
- 6 instruction comparators to compare instructions in code areas. These comparators can be used to implement software patches or hardware breakpoint functions.

## 31.13 DWT (Data Watch Point )

The DWT module consists of four comparators which are:

- A hardware data comparator
- An ETM trigger
- A PC value sampler
- A data address sampler

The DWT can also be used to obtain certain side information. The following data can be obtained with some counters:

- clock cycle
- branching instruction
- access unit operation
- sleep cycle
- CPI (execution time per instruction)
- interrupt overhead

## 31.14 ITM (Instruction Trace Microcell Instrumentation Trace Macrocell)

### 31.14.1 Summarize

ITM is an application-driven trace source that supports printf-type debugging to trace operating system (OS) and application events, and to publish adjudicated system information. ITM publishes trace information in the form of a package, which consists of the following components:

- Software trace: software can publish package information by writing directly to the ITM excitation register.
- Hardware Trace: The ITM publishes information packets generated by the DWT.
- Timestamps: Timestamps are posted to the appropriate packets. The ITM contains a 21-bit counter to generate the timestamps. The clock of the Cortex-M3 or the bit clock rate of the Serial Wire Viewer provides the clock to the counter.

The information packets sent by the ITM are output to the TPIU (Trace Port Interface Unit), which adds some additional packets (refer to TPIU) and then outputs the complete packet sequence to the debugger.

The user must enable the TRCEN bit of the Debug Exception and Monitor Control before setting or using the ITM.

### 31.14.2 Timestamp Packets, Synchronization and Overflow Packets

The timestamp packet contains timestamp information, common control and synchronization information. It uses a 21-bit timestamp counter (and possibly a prescaler), and this counter is reset at the time each timestamp packet is issued. The clock for the counter can be either the CPU clock or the SWV clock.

The synchronization packet is 0x80\_00\_00\_00\_00\_00 and is sent to the TPIU at 00 00 00 00 00 80 (LSB comes first). The synchronization packet is the control signal for the timestamp packet.

It is also sent on each DWT trigger, so the DWT must be configured to trigger the ITM: Bit 0 (CYCCNTENA) of the DWT Control must be set. Bit 2 (SYNCENA) of the ITM Trace Control Register must also be set.

*Notes: If the SYNENA bit is not set, the DWT generates a synchronization trigger to the TPIU that will send only TPIU synchronization packets and not ITM synchronization packets.*

An overflow packet is a special timestamped packet that indicates that data has been written but the FIFO is full.

Table171 Primary ITM Registers

address	Register	descriptive
@E0000FB0	ITM Lock Access	Write 0xC5AC CE55 allows writing of other ITM registers
@E0000E80	ITM Trace Control	Bit 31-24 = always 0 Bit 23 = busy Bits 22-16 = 7 bits of the ATBID to identify the trace data source Bits 15-10 = always 0 Bits 9:8 = prescaled bits 7-5 of the timestamp = undefined Bit 4 = enable SWV function i.e. timestamp counter uses SWV clock Bit 3 = enable DWT excitation function Bit 2 = This bit must be set to 1 to enable the DWT's Generate Synchronization Trigger function to enable the TPIU to send synchronization packets. Bit 1 = timestamp enable Bit 0 = global enable bit for ITMs
@E0000E40	ITM Trace Privilege	Bit 3: Set '1' to enable trace ports 31:24 Bit 2: Set '1' to enable trace port 23:16 Bit 1: Set '1' to enable trace port 15:8 Bit 0: Set '1' to enable trace port 7:0
@E0000E00	ITM Trace Enable	Each bit enables the corresponding trigger port to generate a trace.
@E0000000-E000007C	Stimulus Port Registers 0-31	Write 32-bit data to selected trigger ports (32) that generate traces

**Example on configuration:**

Outputs a simple value to the TUIU:



- Configure the TPIU and enable I/O\_TRACEN to enable the MCU to assign TRACE pins (see Section31.17.2 - Trace Pin Assignments, Section31.16.3 - Debug MCU Configuration Registers);
- Write 0xC5AC CE55 to the ITM Lock Access register to allow writes to other ITM registers;
- Write 0x0001 0005 to the Trace Control register to enable the synchronization packet for the TPIU and enable the entire ITM function with an ATBID of 0x01 in the register;
- Write 0x1 to the ITM Trace Enable register to enable trigger port 0;
- Write 0x1 to the ITM Trace Privilege register to turn off masking of trigger ports 7:0;
- Write the value to be output to the trigger port 0 register, this step can be done by software (using the printf function).

## 31.15 ETM Module (Embedded Trace Macrocell)

### 31.15.1 Summarize

The ETM can reproduce the running process of a program. Changes in data can be tracked using a Data Watchpoint Trigger Module (DWT) or an Instruction Trace Microcell (ITM), but rather the execution of instructions can be tracked using an Embedded Trace Microcell (ETM).

The ETM is triggered and transmits information in packets by built-in resources that can be configured individually by the software, and the trigger source can be selected using the trigger event register (0xE004 1008). The trigger event can be a simple event (e.g. an address match from an address comparator) or the trigger event can be the result of a logical operation between 2 events. The trigger source is one of the four comparators in the DWT module. The following events can be observed:

- Clock period matching
- data address matching

For more information on trigger sources, refer to Section31.13 .

The packets of information transmitted by the ETM are output through the TPIU (Trace Port Interface Unit), which also needs to add some additional information (see Section0 for details) before outputting the complete packet sequence to the debug host.

### 31.15.2 Signaling Protocols and Packet Types

Refer to Chapter 7, "ETM v3 Singal Protocol" of the ARM IHI 0014N document for a description of this section.

### 31.15.3 Main ETM Registers

For a detailed description of the registers, refer to Chapter 3 of the ARMIHI0014N documentation.

Table172 Primary ETM registers

address	Register	clarification
0xE004 1FB0	ETMLockAccess	Write 0xC5AC CE55 to unprotect other ETM registers from writes
0xE004 1000	ETMControl	Controls the operation of the ETM, e.g. how to enable tracking
0xE004 1010	ETMStatus	Provides information on the current state of tracking and triggering logic
0xE004 1008	ETMTriggerEvent	Define the events that will be triggered by the control
0xE004 101C	ETMTraceEnableControl	Define the selected comparator
0xE004 1020	ETMTraceEnableEvent	Define the trace enable event
0xE004 1024	ETMTraceStart/Stop	Define the trace events used by the trigger source to set the trace to start or stop

### 31.15.4 Configuration Example

Output a simple value from TPIU:

- Configure the TPIU and enable I/O\_TRACEN to allocate TRACEI/O in the debug configuration registers;
- Write 0xC5AC CE55 to the ETMLockAccess register to unprotect the ITM register from writing;
- Write 0x0000 1D1E in the control register (configuration trace);
- Write 0x0004 06F (defining the trigger event) in the Trigger Event register (Trigger Event);
- Write 0x0000 006F in the Trace Enable Event register (Trace Enable Event) (defines the start and stop of the event);

- Write 0x0000 0001 (Trace Enable) in the Trace Start/Stop register (Trace Start/Stop);
- Write 0x0000 191E (end of configuration) in the ETM control register (Control).

## 31.16 MCU Debug Module (MCUDBG)

The MCU debug module assists the debugger in providing the following functions:

- Low Power Mode
- Provides clock control of timer, watchdog, I2C and bxCAN at breakpoints
- Control over tracking foot assignment

### 31.16.1 Debugging Support for Low-Power Modes

Low power mode can be accessed using WFI and WFE.

The MCU supports a variety of low-power modes, which can turn off the CPU clock or reduce the power consumption of the CPU, respectively.

The kernel does not allow FCLK or HCLK to be turned off during debugging. these clocks are necessary for debugging operations, so they must work during debugging. the MCU uses a special approach that allows the user to debug code in low-power mode.

To accomplish this, the debugger must first set some configuration registers to change the characteristics of the low-power mode.

- In sleep mode, the debugger must first reset the DBG\_SLEEP bit of the DBGMCU\_CR register. This will provide the same clock for HCLK as FCLK (the system clock configured by the code).
- In stop mode, the debugger must first set the DBG\_STOP bit. This activates the internal RC oscillator that provides the clock for FCLK and HCLK in stop mode.

### 31.16.2 Supports Timer, Watchdog, bxCAN and I2C Debugging

When generating breakpoints, it is necessary to select the operating mode of the counter according to the different uses of the timer and watchdog:

- The counter continues to count when a breakpoint is generated. This is often used when outputting a PWM to control a motor.
- The counter stops counting when a breakpoint is generated. This is necessary for watchdog counters.

For bxCAN, the user has the option to prevent receiving register updates during breakpoints.

For I2C, the user has the option to block the SMBUS timeout during the breakpoint.

### 31.16.3 Debugging MCU Configuration Registers

This register allows the MCU to be down-configured in the debug state. including:

- Supports low power consumption mode
- Timer and watchdog counter support
- Supports bxCAN communication
- Assigning trace pins

The DBGMCU\_CR register is mapped to the external PPB bus with base address 0xE004 2000.

The register is reset asynchronously (not by a system reset) by PORESET. The debugger can write this register when the kernel is in the reset state.

If the debugger does not support these features, the user software can still write these registers.

DBGMCU\_CR

Address: 0xE004 2004 supports 32-bit access only

POR reset: 0x0000 0000 (not reset by system reset)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	DBG_TI M11_ST OP	DBG_TI M10_ST OP	DBG_TI M9_ST OP	DBG_TI M14_ST OP	DBG_TI M13_ST OP	DBG_TI M12_ST OP	Reserved			DBG_C AN2_ST OP	DBG_TI M8 STOP	DBG_TI M7 STOPT	DBG_TI M6 STOPT	DBG_TI M5 STOP	DBG_I2 C2_SMB USTIME OUT
	rw	rw	rw	rw	rw	rw				rw	rw	rw	rw	rw	rw
	15	14	13	12	11	10				9	8	7	6	5	4
DBG_I2 C1_SMB USTIME EOUT	DBG_C AN1_ST OP	DBG_TI M4_ST OP	DBG_TI M3_ST OP	DBG_TI M2_ST OP	DBG_TI M1_ST OP	DBG_W WDG_S TOP	DBG_I WDG_S TOP	TRACE_MODE[1 :0]		TRACE _IOEN	Reserved		DBG_ STAND BY	DBG_S TOP	DBG_SL EEP
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	res	res	rw	rw	rw



Bit	notation	clarification
31	Reserved	Reserved, always reads 0.
30:25	DBG_TIMx_STOP	<b>DBG_TIMx_STOP:</b> TIMx counter stops at core stop (x=9..14) 0: Even if the core is stopped, the clocks of the involved timing counters are input and the output behavior is normal. 1: When the core stops, the clocks of the timer counters involved are stopped and the outputs are disabled (as if there were an emergency stop for interrupt events).
24:22	Reserved	Reserved, always reads 0.
21	DBG_CAN2_STOP	<b>DBG_CAN2_STOP:</b> CAN2 stops running when the kernel enters the debug state. 0: CAN2 is still operating normally; 1: The CAN2 receive register does not continue to receive data.
20:17	DBG_TIMx_STOP	<b>DBG_TIMx_STOP:</b> stops the timer counter when the core stops (x=8..5) 0: When the core is stopped, the clock is still supplied to the counter of the associated timer and the timer outputs work normally; 1: When the core stops, cut off the clock of the counter of the relevant timer, and at the same time turn off the output of the timer (as if it were an emergency response to a pause event, stopping the timer).
16	DBG_I2C2_SMBUS_TIMEOUT	<b>DBG_I2C2_SMBUS_TIMEOUT:</b> stops SMBUS timeout mode when the core stops. 0: Same as normal mode operation; 1: Freeze SMBUS timeout control.
15	DBG_I2C1_SMBUS_TIMEOUT	<b>DBG_I2C1_SMBUS_TIMEOUT:</b> stops SMBUS timeout mode when the core stops. 0: Same as normal mode operation; 1: Freeze SMBUS timeout control.
14	DBG_CAN1_STOP	<b>DBG_CAN1_STOP:</b> CAN1 stops running when the kernel enters the debug state. 0: CAN1 is still operating normally; 1: The receive register of CAN1 does not continue to receive data.
13:10	DBG_TIMx_STOP	<b>DBG_TIMx_STOP:</b> Counter stops working when the kernel enters debug state x=4..1. 0: The counter of the selected timer is still working normally; 1: The counter of the selected timer stops working.
9	DBG_WWDG_STOP	<b>DBG_WWDG_STOP:</b> The debug window watchdog stops working when the kernel enters the debug state. 0: The window watchdog counter is still functioning normally; 1: The window watchdog counter stops working.
8	DBG_IWDG_STOP	<b>DBG_IWDG_STOP:</b> watchdog stops working when the kernel enters debug state 0: The watchdog counter is still functioning normally; 1: The watchdog counter stops working.
7:5	TRACE_MODE[1:0] and TRACE_IOEN	<b>TRACE_MODE[1:0] and TRACE_IOEN:</b> trace pin assignment control -When TRACE_IOEN = 0: TRACE_MODE = xx: No trace pins are assigned (default state). -When TRACE_IOEN=1: TRACE_MODE=00: The trace pin uses asynchronous mode; TRACE_MODE=01: The trace pin uses synchronization mode and has a data length of 1; TRACE_MODE=10: The trace pin uses synchronization mode and has a data length of 2; TRACE_MODE=11: The trace pin uses synchronization mode and has a data length of 4.
4:3	Reserved	Reserved, must be held to 0.
2	DBG_STANDBY	<b>DBG_STANDBY:</b> Debug standby mode. 0: (FCLK off, HCLK off) The entire digital circuit section is powered down. From a software point of view, exiting STANDBY mode is the same as a reset (except that some status bits indicate that the microcontroller has just exited from the STANDBY state). 1: (FCLK on, HCLK on) The digital circuitry section is not powered down and the FCLK and HCLK clocks are clocked by the internal RL oscillator. In addition, the microcontroller exits STANDBY mode by generating a system reset is the same as a reset.
1	DBG_STOP	<b>DBG_STOP:</b> Debugging stop mode. 0: (FCLK off, HCLK off) When in STOP mode, the clock controller disables all clocks (including HCLK and FCLK). When exiting from STOP mode, the clock configuration is the same as after a reset (the microcontroller is clocked by an 8MHz internal RC oscillator (HIS)). Therefore, software must reconfigure the clock control system to start the PLL, crystal, etc. 1: (FCLK on, HCLK on) In stop mode, the FCLK and HCLK clocks are provided by the internal RC oscillator. When exiting stop mode, software must reconfigure the clock system to start the PLL, crystal, etc. (same operation as when configuring this bit to 0).
0	DBG_SLEEP	<b>DBG_SLEEP:</b> Debug Sleep Mode 0: (FCLK on, HCLK off) In sleep mode, FCLK is provided by the originally configured system clock and HCLK is off Closed. Since sleep mode does not reset the configured clock system, the

software does not need to reconfigure the clock system when exiting from sleep mode.  
1: (FCLK on, HCLK on) In sleep mode, both FCLK and HCLK clocks are provided by the originally configured system clock.

## 31.17 TPIU (Trace Port Interface )

### 31.17.1 Introductory

The TPIU acts as a bridge between on-chip data tracking and ITM.

The output data stream is encapsulated into a trace source ID, which is then captured by the Trace Port Analyzer (TPA).

The kernel embeds a simple TPIU (consisting of a special version of the CoreSight TPIU) designed specifically for low-cost debugging.

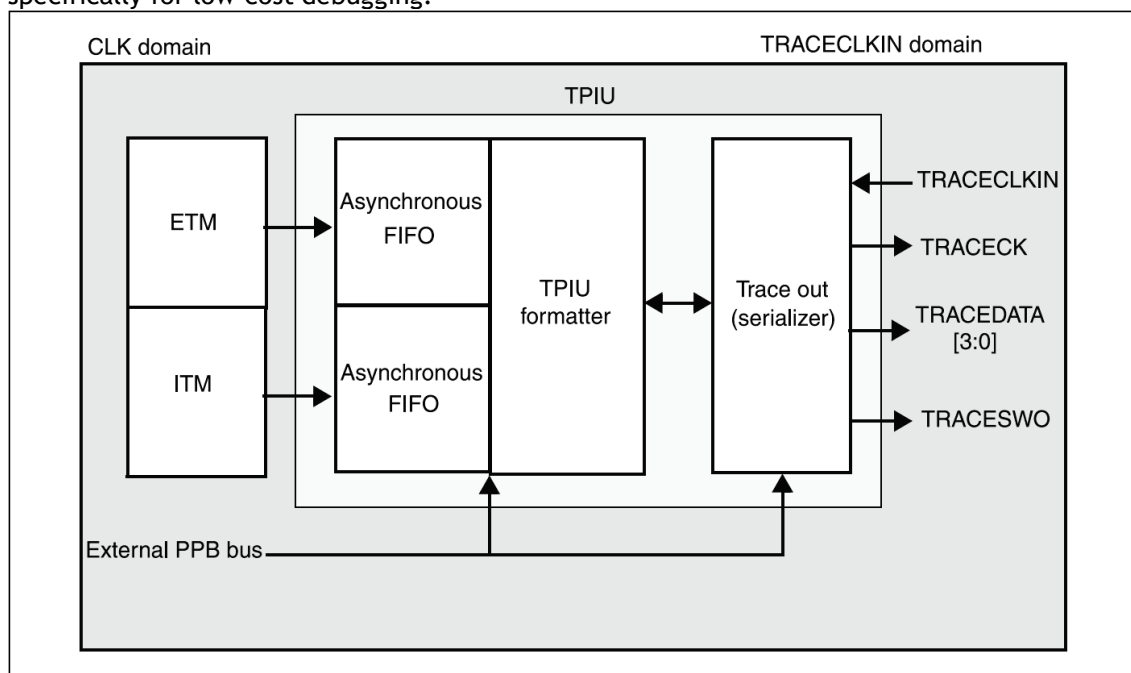


Figure299 TPIU Block Diagram

### 31.17.2 Tracking Pin Assignment

- asynchronous mode

Asynchronous mode requires 1 extra pin and is present in all packages. This mode is only available on the serial debug interface (JTAG debug interface is not supported).

Table173 Asynchronous Tracking Pin Assignments

TPUI Pins	Synchronous tracking mode		W55MH32 Pin Assignment
	typology	descriptive	
TRACESWO	exports	Asynchronous trace data output	PB3

- synchronous mode

Synchronous mode uses two to six extra pins depending on the trace data length and is only found in large package chips. Additionally, this mode works with both JTAG debug interfaces and serial debug interfaces and provides better data output than asynchronous traces.

Table174 Synchronous Tracking Pin Assignments

TPUI Pin Name	Synchronous tracking mode		W55MH32 Pin Assignment
	typology	descriptive	
TRACECK	exports	Tracking Clock	PE2
TRACED[3:0]	exports	Synchronized trace data output, length can be 1,2, or 4	PE[6:3]

### TPUI trace pin assignment

These pins are not dedicated pins by default. These pins can be assigned by setting the TRACE\_IOEN and TRACE\_MODE bits of the MCU Debug Component Configuration register. The setting must be done by the debugger.

In addition, the configuration of the trace determines the number of pins assigned (asynchronous or synchronous).

- Asynchronous mode: 1 extra pin required
- Synchronized mode: 2 to 5 additional pins are used depending on the length of data (1, 2 or 4) on the trace port
  - TRACECK
  - TRACED(0) if the data length is configured as 1, 2 or 4
  - TRACED(1) if the data length is configured as 2 or 4
  - TRACED(2) if the data length is configured as 4
  - TRACED(3) if the data length is configured as 4

The debugger needs to set the TRACE\_IOEN and TRACE\_MODE[1:0] bits of the Debug MCU Configuration register to assign the trace pins. By default, the trace pins are not assigned. This register is mapped to an external PPB and is reset by PORESET (system reset does not reset this register). The debugger can write this register in the system reset state.

Table175 Flexible Tracking Pin Assignments

DBGMCU_CR registers		Pin Usage	Tracking Pin Assignment					
TRACE_IO EN	TRACE_MODE[1:0]		PB3/JTDO/TRACESWO	PE2/TRACECK	PE3/TRACE D[0]	PE4/TRACE D[1]	PE5/TRACE D[2]	PE6/TRACE D[3]
0	XX	untracked (default state)	Release <sup>(1)</sup>	Release (can be used as a normal I/O port)				
1	00	asynchronous trace	TRACESWO					
1	01	Synchronous tracking 1 bit	Release <sup>(1)</sup>	TRACECK	TRACE D[0]	Release (can be used as a normal I/O port)		
1	10	Synchronous tracking 2 bits		TRACECK	TRACE D[0]	TRACE D[1]	liberate (a prisoner) (Can be used as a general I/O port)	
1	11	Synchronized tracking 4-bit		TRACECK	TRACE D[0]	TRACE D[1]	TRACE D[2]	TRACE D[3]

Note (1): This pin is released when using the serial debug interface, and is used as JTDO when using the JTAG debug interface.

Notes: The input clock TRACECLKIN of TUIP is grounded by default. So after the bit TRACE\_IOEN is set, HCLK needs two clock cycles.

The debugger can then configure the trace mode by writing the PROTOCOL[1:0] bits of the SPP\_R (Selected Pin Protocol) register of the TPIU.

- PROTOCOL=00: Tracking mode (synchronization)
  - PROTOCOL=01 or 10: Serial mode (Manchester or NRZ encoding). The default state is 01
- The trace port size is then configured by writing bits [3:0] of the CPSPS\_R (Current Sync Port Size) register of the TPIU.

- 0x1: 1 pin (default)
- 0x2: 2 pins
- 0x8: 4 pins

### 31.17.3 TPUI Formatter

The protocol formatter outputs a frame consisting of 16 bytes:

- 7 data bytes
- 8 multi-purpose bytes, consisting of the following components:
  - Bit 1 (LSB) is used to distinguish between the data byte (0) and the ID byte (1).
  - 7 bits (MSB) can be used as data or to track source ID changes.
- 1 auxiliary byte in which each bit corresponds to one of the 8 multi-purpose bytes:

- If the corresponding multipurpose byte is a data byte, then this bit is the bit 0 bit of the data.
- If the corresponding byte is an ID byte, this bit indicates when the ID change takes effect.

*Notes: For more information, see ARM CoreSight Architecture Specification v1.0 (ARM IHI 0029B)*

### 31.17.4 TPUI Frame Asynchronous Package

TPUI generates two types of synchronization packets:

- Frame Synchronization Package (or Full Word Synchronization Package)

The packet is 0x7F\_FF\_FF\_FF (LSB first), a sequence that can only be used if 0x7F is not encoded as the ID source.

This packet is output periodically between frames.

In continuous mode, the TPA must discard all synchronization frames as soon as they are discovered.

- Halfword Synchronization Pack

The packet is: 0x7F\_FF (LSB first).

It is output periodically between or within frames.

These packets exist only in continuous mode and enable the TPA to detect TRACE ports in IDLE mode (no TRACE is captured). When detected by the TPA, they must be discarded.

### 31.17.5 Synchronized Frame Packet Delivery

Since there is no synchronization counter register in the TPIU of the kernel, the trigger for synchronization can only be generated by the DWT. Refer to the DWT Control Register (bits SYNCTAP[11:10]) and the DWT Current PC Sampler Cycle Count Register.

The TPUI frame synchronization packet (0x7F\_FF\_FF\_FF) is sent in the following cases:

- After each TPIU reset is released. The reset signal is released synchronously to the rising edge of the TRACECLKIN clock, which means that the synchronization packet is sent when the TRACE\_IOEN bit of the DBGMCU\_CFG register is set. In this case, packet 0x7F\_FF\_FF\_FF is not followed by a packet of any format.
- At the time each DWT is triggered (assuming the DWT has been set up beforehand), there are two scenarios:
  - If the SYNENA bit of the ITM = 0, only the word 0x7F\_FF\_FF\_FF is sent, followed by no packet in any format.
  - If ITM's SYNENA bit = 1, the ITM synchronization packet will follow (0x80\_00\_00\_00\_00\_00\_00\_00), formatted by the TPUI (plus the trace source ID).

### 31.17.6 Synchronous Mode

The number of pins tracking output data can be 4, 2 or 1 by TRACED(3:0).

Configuration Clock Output to Debugger (TRACECK) TRACECLKIN is driven internally and is connected to HCLK only when TRACE is used.

*Notes: In this type of synchronization mode, it is not necessary to provide a stable clock frequency.*

The I/O ports of TRACE (including TRACECK) are driven by the rising edge of TRACLKIN (equivalent to HCLK). Therefore, the output frequency of TRACECK is equal to HCLK/2.

### 31.17.7 Asynchronous Mode

The debug module provides a low-cost, one-pin trace data output capability using the asynchronous output pin TRACESWO. However, it is clear that the bandwidth of such output data is limited.

The TRACESWO pin is multiplexed with the JTDO pin and is only valid for the SW-DP debug interface, so all packages of the W55MH32 provide this feature.

Asynchronous mode requires a smooth frequency availability on the TRACECLKIN pin. For the standard UART (NRZ) capture mechanism, 5% correctness is required. Manchester encoding can be relaxed to 10%.

### 31.17.8 TRACECLKIN Connections inside the W55MH32

The TRACECLKIN input is internally connected to HCLK in the W55MH32. This means that when using the asynchronous tracking mode, the application should limit the use of time frames to ensure a stable CPU frequency.

*Notes: IMPORTANT: Caution is required when using the asynchronous tracking feature:*

The initial clock of the W55MH32 microcontroller is an internal RC oscillator, and the frequency of this oscillator in the reset state is different from the frequency after reset. This is due to the fact that the RC calibration uses an initial value in the reset state, and this value is updated after the reset is released.

Therefore, the trace function of the Trace Port Analyzer (set TRACE\_IOEN) should not be activated in the system reset state. This is because the bit width of a synchronized frame packet in the reset state is different from the packet after reset.

### 31.17.9 TPIU Register

The TPIUAPB register can be read or written only if the TRCENA bit of the Debug Exception and Monitor Control (DEMCR) register is set. Otherwise the register reads a value of 0 (the output of this bit enables the PCLK of the TPIU).

Table176 Important TPIU Registers

address	Register	descriptive
0xE004 0004	Current port size	Tracks the length of the port: bit 0: port length is 1 bit 1: port length is 2 Bit 2: port length 3, not supported Bit 3: port length 4 Only one of the four bits can be set at the same time. By default, the port length is 1 (0x0000 0001)
0xE004 00F0	Selected pin protocol	Trace port protocol selection: bits 1:0 = 00: Synchronized tracking mode 01: Serial output - Manchester code (default) 10: Serial output - NRZ 11: Undefined
0xE004 0304	Formatter and flush control	Bits 31-9: always 0 Bit 8 = TriglN: always 1, indicating trigger bits 7-4: always 0 Bit 3-2: always 0 Bit 1 = EnFCnt: In synchronous mode (Bit1:0 of the SelectPinProtocol register is 00), this bit is forced to 1 and the formatter is automatically enabled in continuous mode. Asynchronous mode (Bit1:0 of the SelectPinProtocol register is not 00), this bit can be set or reset to select whether to enable the formatter or not. Bit 0: always 0 The default value is 0x102 Note: In synchronized mode, the formatter is automatically enabled in continuous mode since the TRACECTL signal has no external pin. This means that the formatter inserts some control packets to identify the source of the trace packet.
0xE004 0300	Formatter and flush status	Not used in Cortex-M3, the readout value is always 0x0000 0008

### 31.17.10 Examples of Configurations

- Sets the TRCENA bit of the Debug Exception and Monitor Control register;
- Write the desired value in the TPIU Current Port Size register (default is 0x1, indicating a port length of 1 bit);
- Write 0x102 (default value) to the TPIU Formatter and Flush Control register;
- Write the TPIU Select Pin Protocol register to select synchronous or asynchronous mode. For example, writing 0x2 selects asynchronous mode for NRZ encoding (similar to URAT);
- Write 0x20 to the DBGMCU Control register (set IO\_TRACEN) to assign the I/O port of TRACE for asynchronous mode. At this point the TPIU will send a synchronization packet (FF\_FF\_FF\_7F);
- Configure the ITM and write the ITMStimulus register to output data.

## 31.18 DBG Register Address Map

The following table summarizes the debug registers.

Table177 DBG-Register and Reset values

address	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
0xE004 2000	DBGMCU_IDCODE	REV_ID																Reserved				DEV_ID															
	Reset value <sup>(1)</sup>	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x					x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
0xE004 2004	DBGMCU_CR	Reserved																DBG_CAN2_STOP	DBG_TIM8_STOP	DBG_TIM7_STOP	DBG_TIM6_STOP	DBG_TIM5_STOP	DBG_I2C2_SMBUS_TIMEOUT	DBG_I2C1_SMBUS_TIMEOUT	DBG_CAN1_STOP	DBG_TIM4_STOP	DBG_TIM3_STOP	DBG_TIM2_STOP	DBG_TIM1_STOP	DBG_WWDG_STOP	DBG_IWDG_STOP	TRACE_MODE [1:0]	TRACE_IOEN	Reserved	DBG_STANDBY	DBG_STOP	DBG_SLEEP
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0

(1) See Section 26.6.1 - Microcontroller Device ID Coding for Reset value details.

---

## Document History

releases	dates	clarification
Ver. 1.0.0	2025-05-20	First Release
Ver. 1.0.1	2026-03-15	Fix the error in the description of the interrupted chapter add TRNG,SYSCFG and OTP register description modify CRC and RCC register description

---

---

## copyright statement

Copyright 2025 WIZnet.  
Technical Support. [support@wiznet.hk](mailto:support@wiznet.hk)  
Sales & Agents. [sales@wiznet.hk](mailto:sales@wiznet.hk)  
For more information, visit <http://www.wiznet.io>