

**GigaDevice Semiconductor Inc.**

**GD32F1x0**

**ARM® Cortex™ -M3 32-bit MCU**

**User Manual**

Revision 3.1.0

( Jan. 2018 )

# Table of Contents

<b>Table of Contents .....</b>	<b>1</b>
<b>List of Figures .....</b>	<b>16</b>
<b>List of Tables .....</b>	<b>22</b>
<b>1. System and memory architecture .....</b>	<b>1</b>
<b>1.1. ARM Cortex-M3 processor .....</b>	<b>1</b>
<b>1.2. System architecture .....</b>	<b>2</b>
<b>1.3. Memory map .....</b>	<b>4</b>
1.3.1. Bit-banding .....	7
1.3.2. On-chip SRAM memory .....	9
1.3.3. On-chip Flash memory .....	10
<b>1.4. Boot configuration .....</b>	<b>10</b>
<b>1.5. System configuration registers (SYSCFG) .....</b>	<b>12</b>
1.5.1. System configuration register 0 (SYSCFG_CFG0) .....	12
1.5.2. System configuration register 1 (SYSCFG_CFG1) .....	13
1.5.3. EXTI sources selection register 0 (SYSCFG_EXTISS0) .....	13
1.5.4. EXTI sources selection register 1 (SYSCFG_EXTISS1) .....	15
1.5.5. EXTI sources selection register 2 (SYSCFG_EXTISS2) .....	16
1.5.6. EXTI sources selection register 3 (SYSCFG_EXTISS3) .....	17
1.5.7. System configuration register 2 (SYSCFG_CFG2) .....	18
<b>1.6. Device electronic signature .....</b>	<b>19</b>
1.6.1. Memory density information .....	19
1.6.2. Unique device ID (96 bits) .....	20
<b>2. Flash memory controller (FMC) .....</b>	<b>22</b>
<b>2.1. Overview .....</b>	<b>22</b>
<b>2.2. Characteristics .....</b>	<b>22</b>
<b>2.3. Function overview .....</b>	<b>22</b>
2.3.1. Flash memory architecture.....	22
2.3.2. Read operations.....	23
2.3.3. Unlock the FMC_CTL register .....	23
2.3.4. Page erase .....	23
2.3.5. Mass erase.....	25
2.3.6. Main flash programming .....	26
2.3.7. Option bytes erase .....	28
2.3.8. Option bytes programming .....	28
2.3.9. Option bytes description .....	29

2.3.10.	Page erase/Program protection.....	30
2.3.11.	Security protection .....	30
<b>2.4.</b>	<b>Register definition .....</b>	<b>32</b>
2.4.1.	Wait state register (FMC_WS) .....	32
2.4.2.	Unlock key register (FMC_KEY).....	32
2.4.3.	Option bytes unlock key register (FMC_OBKEY).....	33
2.4.4.	Status register (FMC_STAT).....	33
2.4.5.	Control register (FMC_CTL) .....	34
2.4.6.	Address register (FMC_ADDR).....	35
2.4.7.	Option bytes status register (FMC_OBSTAT).....	36
2.4.8.	Write protection register (FMC_WP).....	36
2.4.9.	Wait state enable register (FMC_WSEN) .....	37
2.4.10.	Product ID register (FMC_PID).....	38
<b>3.</b>	<b>Power management unit (PMU).....</b>	<b>39</b>
3.1.	Overview .....	39
3.2.	Characteristics .....	39
3.3.	Function overview .....	39
3.3.1.	Battery backup domain .....	41
3.3.2.	V <sub>DD</sub> /V <sub>DDA</sub> power domain.....	42
3.3.3.	1.2V power domain for GD32F130xx and GD32F150xx devices.....	44
3.3.4.	1.8V power domain for GD32F170xx and GD32F190xx devices.....	44
3.3.5.	Power saving modes.....	45
3.4.	Register definition .....	47
3.4.1.	Control register (PMU_CTL).....	47
3.4.2.	Power control/status register (PMU_CS).....	49
<b>4.</b>	<b>Reset and clock unit (RCU).....</b>	<b>51</b>
4.1.	Reset control unit (RCTL).....	51
4.1.1.	Overview.....	51
4.1.2.	Function overview .....	51
4.2.	Clock control unit (CCTL) .....	52
4.2.1.	Overview.....	52
4.2.2.	Characteristics .....	55
4.2.3.	Function overview .....	55
4.3.	Register definition .....	61
4.3.1.	Control register 0 (RCU_CTL0) .....	61
4.3.2.	Configuration register 0 (RCU_CFG0) .....	62
4.3.3.	Interrupt register (RCU_INT).....	69
4.3.4.	APB2 reset register (RCU_APB2RST).....	75
4.3.5.	APB1 reset register (RCU_APB1RST).....	76
4.3.6.	AHB enable register (RCU_AHBEN) .....	81

4.3.7.	APB2 enable register (RCU_APB2EN) .....	83
4.3.8.	APB1 enable register (RCU_APB1EN) .....	84
4.3.9.	Backup domain control register (RCU_BDCTL) .....	89
4.3.10.	Reset source /clock register (RCU_RSTSCK).....	92
4.3.11.	AHB reset register (RCU_AHBRSRST).....	93
4.3.12.	Configuration register 1 (RCU_CFG1) .....	94
4.3.13.	Configuration register 2 (RCU_CFG2) .....	95
4.3.14.	Control register 1 (RCU_CTL1) .....	97
4.3.15.	Configuration register 3 (RCU_CFG3) of GD32F170xx and GD32F190xx devices .....	99
4.3.16.	APB1 additional enable register (RCU_ADDAPB1EN) .....	100
4.3.17.	APB1 additional reset register (RCU_ADDAPB1RST) .....	100
4.3.18.	Voltage key register (RCU_VKEY) .....	101
4.3.19.	Deep-sleep mode voltage register (RCU_DSV) .....	101
4.3.20.	Power down voltage select register (RCU_PDVSEL) of GD32F130xx and GD32F150xx devices .....	102
<b>5.</b>	<b>Interrupt/event controller (EXTI) .....</b>	<b>104</b>
5.1.	Overview .....	104
5.2.	Characteristics .....	104
5.3.	Interrupts function overview .....	104
5.4.	External interrupt and event (EXTI) block diagram .....	108
5.5.	External interrupt and Event function overview .....	109
5.6.	Register definition .....	112
5.6.1.	Interrupt Enable register (EXTI_INTEN) .....	112
5.6.2.	Event enable register (EXTI_EVEN) .....	112
5.6.3.	Rising edge trigger enable register (EXTI_RTEN) .....	113
5.6.4.	Falling edge trigger enable register (EXTI_FTEN).....	113
5.6.5.	Software interrupt event register (EXTI_SWIEV) .....	114
5.6.6.	Pending register (EXTI_PD) .....	115
<b>6.</b>	<b>General-purpose I/Os (GPIO) .....</b>	<b>116</b>
6.1.	Overview .....	116
6.2.	Characiteristics .....	116
6.3.	Function overview .....	116
6.3.1.	GPIO pin configuration .....	118
6.3.2.	Alternate functions (AF) .....	118
6.3.3.	Additional functions .....	119
6.3.4.	Input configuration .....	119
6.3.5.	Output configuration .....	119
6.3.6.	Analog configuration .....	120
6.3.7.	Alternate function (AF) configuration.....	121
6.3.8.	GPIO locking function .....	122
6.3.9.	GPIO single cycle toggle function .....	122

<b>6.4. Register definition .....</b>	<b>123</b>
6.4.1. Port control register (GPIOx_CTL) (x=A..D,F) .....	123
6.4.2. Port output mode register (GPIOx_OMODE) (x=A..D,F) .....	124
6.4.3. Port output speed register (GPIOx_OSPD) (x=A..D,F) .....	126
6.4.4. Port pull-up/down register (GPIOx_PUD) (x=A..D,F) .....	128
6.4.5. Port input status register (GPIOx_ISTAT) (x=A..D,F) .....	129
6.4.6. Port output control register (GPIOx_OCTL) (x=A..D,F).....	130
6.4.7. Port bit operate register (GPIOx_BOP) (x=A..D,F).....	130
6.4.8. Port configuration lock register (GPIOx_LOCK) (x=A, B) .....	131
6.4.9. Alternate function selected register0 (GPIOx_AFSEL0) (x=A, B, C).....	132
6.4.10. Alternate function selected register1 (GPIOx_AFSEL1) (x=A,B,C).....	133
6.4.11. Bit clear register (GPIOx_BC) (x=A..D,F).....	134
6.4.12. Port bit toggle register (GPIOx_TGx) (x=A..D,F) of GD32F170xx and GD32F190xx devices.....	135
<b>7. CRC calculation unit.....</b>	<b>136</b>
7.1. Overview .....	136
7.2. Characteristics .....	136
7.3. Function overview .....	137
7.4. Register definition .....	139
7.4.1. Data Register (CRC_DATA) .....	139
7.4.2. Free Data Register (CRC_FDATA).....	139
7.4.3. Control Register (CRC_CTL).....	140
7.4.4. Initialization Data Register (CRC_IDATA).....	140
<b>8. Direct memory access controller (DMA).....</b>	<b>142</b>
8.1. Overview .....	142
8.2. Characteristics .....	142
8.3. Block diagram .....	143
8.4. Function overview .....	143
8.4.1. DMA operation .....	143
8.4.2. Peripheral handshake .....	145
8.4.3. Arbitration .....	146
8.4.4. Address generation.....	146
8.4.5. Circular mode .....	146
8.4.6. Memory to memory mode .....	146
8.4.7. Channel configuration .....	146
8.4.8. Interrupt .....	147
8.4.9. DMA request mapping .....	148
8.5. Register definition .....	151
8.5.1. Interrupt flag register (DMA_INTF).....	151
8.5.2. Interrupt flag clear register (DMA_INTC).....	151
8.5.3. Channel x control register (DMA_CHxCTL) .....	152

8.5.4.	Channel x counter register (DMA_CHxCNT) .....	154
8.5.5.	Channel x peripheral base address register (DMA_CHxPADDR).....	155
8.5.6.	Channel x memory base address register (DMA_CHxMADDR) .....	155
<b>9.</b>	<b>Debug (DBG) .....</b>	<b>157</b>
9.1.	Introduction.....	157
9.2.	Serial Wire Debug port introduction .....	157
9.2.1.	Pin assignment .....	157
9.2.2.	JEDEC-106 ID code.....	157
9.3.	Debug hold function description .....	158
9.3.1.	Debug support for power saving mode .....	158
9.3.2.	Debug support for TIMER, I2C, RTC, WWDGT and FWDGT .....	158
9.4.	DBG registers .....	159
9.4.1.	ID code register (DBG_ID).....	159
9.4.2.	Control register 0(DBG_CTL0).....	159
9.4.3.	Control register 1 (DBG_CTL1).....	163
<b>10.</b>	<b>Analog to digital converter (ADC) .....</b>	<b>165</b>
10.1.	Overview.....	165
10.2.	Characteristics.....	165
10.3.	Pins and internal signals.....	167
10.4.	Function overview.....	168
10.4.1.	Calibration (ADC_CLB) .....	169
10.4.2.	Dual clock domain architecture.....	170
10.4.3.	ADCON switch.....	170
10.4.4.	Regular and inserted channel groups.....	170
10.4.5.	Conversion modes .....	170
10.4.6.	Inserted channel management.....	175
10.4.7.	Analog watchdog .....	176
10.4.8.	Data alignment .....	176
10.4.9.	Programmable sample time .....	178
10.4.10.	External trigger .....	178
10.4.11.	DMA request .....	179
10.4.12.	Temperature sensor and internal reference voltage $V_{REFINT}$ .....	179
10.4.13.	Battery voltage monitoring .....	180
10.4.14.	ADC interrupts.....	180
10.4.15.	Programmable resolution (DRES) - fast conversion mode for GD32F170xx and GD32F190xx devices .....	180
10.4.16.	On-chip hardware oversampling for GD32F170xx and GD32F190xx devices.....	181
<b>10.5.</b>	<b>Register definition.....</b>	<b>184</b>
10.5.1.	Status register (ADC_STAT) .....	184
10.5.2.	Control register 0 (ADC_CTL0).....	185

10.5.3.	Control register 1 (ADC_CTL1) .....	188
10.5.4.	Sampling time register 0 (ADC_SAMPT0) .....	190
10.5.5.	Sampling time register 1 (ADC_SAMPT1) .....	192
10.5.6.	Inserted channel data offset registers (ADC_IOFFx) (x=0..3) .....	193
10.5.7.	Watchdog high threshold register (ADC_WDHT).....	193
10.5.8.	Watchdog low threshold register (ADC_WDLT).....	194
10.5.9.	Regular sequence register 0 (ADC_RSQ0) .....	194
10.5.10.	Regular sequence register 1 (ADC_RSQ1) .....	195
10.5.11.	Regular sequence register 2 (ADC_RSQ2) .....	195
10.5.12.	Inserted sequence register (ADC_ISQ) .....	196
10.5.13.	Inserted data registers (ADC_IDATAx) (x= 0..3).....	197
10.5.14.	Regular data register (ADC_RDATA).....	197
10.5.15.	Oversampling control register (ADC_OVSAMPCTL) of GD32F170xx and GD32F190xx devices ....	197
<b>11.</b>	<b>Digital-to-analog converter (DAC) .....</b>	<b>200</b>
11.1.	Overview.....	200
11.2.	Characteristic .....	200
11.3.	Function overview.....	201
11.3.1.	DAC enable .....	201
11.3.2.	DAC output buffer .....	201
11.3.3.	DAC data configuration.....	202
11.3.4.	DAC trigger .....	202
11.3.5.	DAC conversion.....	203
11.3.6.	DAC output voltage .....	203
11.3.7.	DMA request .....	203
11.3.8.	DAC concurrent conversion for GD32F190xx devices.....	204
11.4.	Register definition.....	206
11.4.1.	Control register (DAC_CTL) .....	206
11.4.2.	Software trigger register (DAC_SWT) .....	209
11.4.3.	DAC0 12-bit right-aligned data holding register (DAC0_R12DH) .....	210
11.4.4.	DAC0 12-bit left-aligned data holding register (DAC0_L12DH).....	210
11.4.5.	DAC0 8-bit right-aligned data holding register (DAC0_R8DH) .....	211
11.4.6.	DAC1 12-bit right-aligned data holding register (DAC1_R12DH) of GD32F190xx devices.....	211
11.4.7.	DAC1 12-bit left-aligned data holding register (DAC1_L12DH) of GD32F190xx devices .....	212
11.4.8.	DAC1 8-bit right-aligned data holding register (DAC1_R8DH) of GD32F190xx devices.....	212
11.4.9.	DAC concurrent mode 12-bit right-aligned data holding register (DACC_R12DH) of GD32F190xx devices .....	213
11.4.10.	DAC concurrent mode 12-bit left-aligned data holding register (DACC_L12DH) of GD32F190xx devices .....	213
11.4.11.	DAC concurrent mode 8-bit right-aligned data holding register (DACC_R8DH) of GD32F190xx devices .....	214
11.4.12.	DAC0 data output register (DAC0_DO).....	214
11.4.13.	DAC1 data output register (DAC1_DO) of GD32F190xx devices .....	215

11.4.14.	Status register (DAC_STAT) .....	215
<b>12.</b>	<b>Comparator (CMP) .....</b>	<b>217</b>
12.1.	Overview .....	217
12.2.	Characteristics .....	217
12.3.	Function overview .....	217
12.3.1.	CMP clock and reset .....	219
12.3.2.	CMP inputs and outputs .....	219
12.3.3.	CMP power mode .....	219
12.3.4.	CMP hysteresis .....	219
12.3.5.	CMP register write protection .....	220
12.4.	Register definition .....	221
12.4.1.	Control/status register (CMP_CS) .....	221
<b>13.</b>	<b>Watchdog timer (WDGT) .....</b>	<b>228</b>
13.1.	Free watchdog timer (FWDGT) .....	228
13.1.1.	Overview .....	228
13.1.2.	Characteristics .....	228
13.1.3.	Function overview .....	228
13.1.4.	Register definition .....	231
13.2.	Window watchdog timer (WWDGT) .....	235
13.2.1.	Overview .....	235
13.2.2.	Characteristics .....	235
13.2.3.	Function overview .....	235
13.2.4.	Register definition .....	238
<b>14.</b>	<b>Real-time clock(RTC).....</b>	<b>240</b>
14.1.	Overview .....	240
14.2.	Characteristics .....	240
14.3.	Function overview .....	241
14.3.1.	Block diagram .....	241
14.3.2.	Clock source and prescalers .....	242
14.3.3.	Real-time clock and calendar .....	242
14.3.4.	Configurable and field maskable alarm .....	242
14.3.5.	RTC initialization and configuration .....	243
14.3.6.	Calendar reading .....	244
14.3.7.	Resetting the RTC .....	245
14.3.8.	RTC shift function .....	246
14.3.9.	RTC reference clock detection .....	246
14.3.10.	RTC smooth digital calibration .....	247
14.3.11.	Time-stamp function .....	249
14.3.12.	Tamper detection .....	249

14.3.13.	Calibration clock output .....	250
14.3.14.	Alarm output .....	251
14.3.15.	RTC power saving mode management .....	251
14.3.16.	RTC interrupts.....	251
<b>14.4.</b>	<b>Register definition .....</b>	<b>253</b>
14.4.1.	Time register (RTC_TIME) .....	253
14.4.2.	Date register (RTC_DATE) .....	253
14.4.3.	Control register (RTC_CTL).....	254
14.4.4.	Status register (RTC_STAT) .....	255
14.4.5.	Prescaler register (RTC_PSC).....	257
14.4.6.	Alarm 0 time and date register (RTC_ALRMO TD) .....	257
14.4.7.	Write protection key register (RTC_WPK).....	258
14.4.8.	Sub second register (RTC_SS) .....	258
14.4.9.	Shift function control register (RTC_SHIFTCTL) .....	259
14.4.10.	Time of time stamp register (RTC_TTS) .....	259
14.4.11.	Date of time stamp register (RTC_DTS) .....	260
14.4.12.	Sub second of time stamp register (RTC_SSTS) .....	260
14.4.13.	High resolution frequency compensation register (RTC_HRFC) .....	261
14.4.14.	Tamper register (RTC_TAMP).....	261
14.4.15.	Alarm 0 sub second register (RTC_ALRMOSS) .....	263
14.4.16.	Backup registers (RTC_BKPx) (x=0..4) .....	264
<b>15.</b>	<b>Timer (TIMERx) .....</b>	<b>265</b>
<b>15.1.</b>	<b>Advanced timer (TIMERx,x=0) .....</b>	<b>266</b>
15.1.1.	Overview.....	266
15.1.2.	Characteristics .....	266
15.1.3.	Block diagram .....	266
15.1.4.	Function overview .....	268
15.1.5.	TIMERx registers(x=0).....	296
<b>15.2.</b>	<b>General level0 timer (TIMERx, x=1, 2).....</b>	<b>321</b>
15.2.1.	Overview.....	321
15.2.2.	Characteristics .....	321
15.2.3.	Block diagram .....	321
15.2.4.	Function overview .....	323
15.2.5.	TIMERx registers(x=1, 2).....	340
<b>15.3.</b>	<b>General level2 timer (TIMERx, x=13).....</b>	<b>364</b>
15.3.1.	Overview.....	364
15.3.2.	Characteristics .....	364
15.3.3.	Block diagram .....	364
15.3.4.	Function overview .....	366
15.3.5.	TIMERx registers(x=13).....	373
<b>15.4.</b>	<b>General level3 timer (TIMERx, x=14).....</b>	<b>382</b>

15.4.1.	Overview.....	382
15.4.2.	Characteristics .....	382
15.4.3.	Block diagram .....	382
15.4.4.	Function overview .....	383
15.4.5.	TIMERx registers(x=14).....	400
<b>15.5.</b>	<b>General level4 timer (TIMERx, x=15,16) .....</b>	<b>418</b>
15.5.1.	Overview.....	418
15.5.2.	Characteristics .....	418
15.5.3.	Block diagram .....	418
15.5.4.	Function overview .....	419
15.5.5.	TIMERx registers(x=15,16).....	434
<b>15.6.</b>	<b>Basic timer (TIMERx, x=5) .....</b>	<b>449</b>
15.6.1.	Overview.....	449
15.6.2.	Characteristics .....	449
15.6.3.	Block diagram .....	449
15.6.4.	Function overview .....	449
15.6.5.	TIMERx registers(x=5).....	454
<b>16.</b>	<b>Infrared ray port (IFRP) .....</b>	<b>459</b>
16.1.	Overview.....	459
16.2.	Characteristics.....	459
16.3.	Function overview.....	459
<b>17.</b>	<b>Universal synchronous asynchronous receiver transmitter (USART).....</b>	<b>461</b>
17.1.	Overview.....	461
17.2.	Characteristics .....	461
17.3.	Function overview.....	463
17.3.1.	USART frame format.....	464
17.3.2.	Baud rate generation .....	465
17.3.3.	USART transmitter .....	465
17.3.4.	USART receiver .....	466
17.3.5.	Use DMA for data buffer access .....	468
17.3.6.	Hardware flow control.....	470
17.3.7.	Multi-processor communication .....	471
17.3.8.	LIN mode .....	472
17.3.9.	Synchronous mode.....	473
17.3.10.	IrDA SIR ENDEC mode.....	474
17.3.11.	Half-duplex communication mode.....	475
17.3.12.	Smartcard (ISO7816) mode .....	475
17.3.13.	Auto baudrate detection .....	477
17.3.14.	ModBus communication .....	478
17.3.15.	Wakeup from Deep-sleep mode.....	478

17.3.16.	USART interrupts .....	479
<b>17.4.</b>	<b>Register definition .....</b>	<b>481</b>
17.4.1.	Control register 0 (USART_CTL0) .....	481
17.4.2.	Control register 1 (USART_CTL1) .....	483
17.4.3.	Control register 2 (USART_CTL2) .....	486
17.4.4.	Baud rate generator register (USART_BAUD) .....	489
17.4.5.	Prescaler and guard time configuration register (USART_GP).....	489
17.4.6.	Receiver timeout register (USART_RT) .....	490
17.4.7.	Command register (USART_CMD) .....	491
17.4.8.	Status register (USART_STAT) .....	492
17.4.9.	Interrupt status clear register (USART_INTC) .....	496
17.4.10.	Receive data register (USART_RDATA).....	497
17.4.11.	Transmit data register (USART_TDATA) .....	497
<b>18.</b>	<b>Inter-integrated circuit interface(I2C).....</b>	<b>499</b>
<b>18.1.</b>	<b>Overview.....</b>	<b>499</b>
<b>18.2.</b>	<b>Characteristics.....</b>	<b>499</b>
<b>18.3.</b>	<b>Function overview.....</b>	<b>499</b>
18.3.1.	SDA and SCL lines .....	500
18.3.2.	Data validation.....	501
18.3.3.	START and STOP condition.....	501
18.3.4.	Clock synchronization .....	501
18.3.5.	Arbitration .....	502
18.3.6.	I2C communication flow.....	502
18.3.7.	Programming model.....	503
18.3.8.	SCL line stretching .....	512
18.3.9.	Use DMA for data transfer.....	513
18.3.10.	Packet error checking.....	513
18.3.11.	SMBus support .....	513
18.3.12.	Status, errors and interrupts .....	515
<b>18.4.</b>	<b>Register definition.....</b>	<b>517</b>
18.4.1.	Control register 0 (I2C_CTL0).....	517
18.4.2.	Control register 1 (I2C_CTL1).....	518
18.4.3.	Slave address register 0 (I2C_SADDR0).....	519
18.4.4.	Slave address register 1 (I2C_SADDR1).....	520
18.4.5.	Transfer buffer register (I2C_DATA) .....	520
18.4.6.	Transfer status register 0 (I2C_STAT0).....	521
18.4.7.	Transfer status register 1 (I2C_STAT1).....	523
18.4.8.	Clock configure register (I2C_CKCFG) .....	524
18.4.9.	Rise time register (I2C_RT) .....	525
18.4.10.	SAM control and status register (I2C_SAMCS) of GD32F170xx and GD32F190xx devices .....	525
<b>19.</b>	<b>Serial peripheral interface/Inter-IC sound (SPI/I2S).....</b>	<b>527</b>

<b>19.1.</b>	<b>Overview.....</b>	<b>527</b>
<b>19.2.</b>	<b>Characteristics.....</b>	<b>527</b>
19.2.1.	SPI characteristics .....	527
19.2.2.	I2S characteristics .....	527
<b>19.3.</b>	<b>SPI block diagram .....</b>	<b>528</b>
<b>19.4.</b>	<b>SPI signal description.....</b>	<b>529</b>
19.4.1.	Normal configuration .....	529
19.4.2.	Quad-SPI configuration for GD32F170xx and GD32F190xx devices .....	529
<b>19.5.</b>	<b>SPI function overview.....</b>	<b>530</b>
19.5.1.	SPI clock timing and data format .....	530
19.5.2.	NSS function .....	531
19.5.3.	SPI operation modes .....	531
19.5.4.	DMA function .....	538
19.5.5.	CRC function .....	538
<b>19.6.</b>	<b>SPI interrupts .....</b>	<b>539</b>
19.6.1.	Status flags.....	539
19.6.2.	Error conditions .....	539
<b>19.7.</b>	<b>I2S block diagram .....</b>	<b>540</b>
<b>19.8.</b>	<b>I2S signal description.....</b>	<b>540</b>
<b>19.9.</b>	<b>I2S function overview.....</b>	<b>541</b>
19.9.1.	I2S audio standards .....	541
19.9.2.	I2S clock.....	549
19.9.3.	Operation.....	550
19.9.4.	DMA function .....	553
<b>19.10.</b>	<b>I2S interrupts .....</b>	<b>553</b>
19.10.1.	Status flags .....	553
19.10.2.	Error conditions.....	554
<b>19.11.</b>	<b>Register definition.....</b>	<b>555</b>
19.11.1.	Control register 0 (SPI_CTL0).....	555
19.11.2.	Control register 1 (SPI_CTL1).....	556
19.11.3.	Status register (SPI_STAT) .....	557
19.11.4.	Data register (SPI_DATA).....	559
19.11.5.	CRC polynomial register (SPI_CRCPOLY).....	559
19.11.6.	RX CRC register (SPI_RCRC) .....	560
19.11.7.	TX CRC register (SPI_TCRC).....	560
19.11.8.	I2S control register (SPI_I2SCTL).....	561
19.11.9.	I2S clock prescaler register (SPI_I2SPSC).....	563
19.11.10.	Quad-SPI mode control register (SPI_QCTL) of GD32F170xx and GD32F190xx devices .....	563
<b>20.</b>	<b>HDMI-CEC controller(HDMI-CEC).....</b>	<b>565</b>

<b>20.1.</b>	<b>Overview.....</b>	<b>565</b>
<b>20.2.</b>	<b>Characteristics.....</b>	<b>565</b>
<b>20.3.</b>	<b>Function overview.....</b>	<b>566</b>
20.3.1.	CEC bus pin .....	566
20.3.2.	Message description.....	566
20.3.3.	Bit timing description .....	567
20.3.4.	Arbitration .....	568
20.3.5.	SFT option bit description.....	569
20.3.6.	Error definition .....	569
20.3.7.	HDMI-CEC interrupt.....	572
<b>20.4.</b>	<b>Register definition.....</b>	<b>574</b>
20.4.1.	Control register (CEC_CTL).....	574
20.4.2.	Configuration register (CEC_CFG).....	574
20.4.3.	Transmit data register (CEC_TDATA) .....	576
20.4.4.	Receive data register (CEC_RDATA) .....	577
20.4.5.	Interrupt Flag Register (CEC_INTF) .....	577
20.4.6.	Interrupt enable register (CEC_INTEN).....	579
<b>21.</b>	<b>Touch sensing interface(TSI).....</b>	<b>582</b>
<b>21.1.</b>	<b>Overview.....</b>	<b>582</b>
<b>21.2.</b>	<b>Characteristics.....</b>	<b>582</b>
<b>21.3.</b>	<b>Function overview.....</b>	<b>582</b>
21.3.1.	TSI block diagram.....	582
21.3.2.	Touch sensing technique overview.....	583
21.3.3.	Charge transfer sequence.....	583
21.3.4.	Charge transfer sequence FSM.....	586
21.3.5.	Clock and duration time of states.....	587
21.3.6.	PIN mode control of TSI.....	588
21.3.7.	ASW and hysteresis mode .....	588
21.3.8.	TSI operation flow .....	588
21.3.9.	TSI flags and interrupts.....	588
21.3.10.	TSI GPIOs .....	589
<b>21.4.</b>	<b>Register definition.....</b>	<b>590</b>
21.4.1.	Control register (TSI_CTL) .....	590
21.4.2.	Interrupt enable register (TSI_INTEN) .....	592
21.4.3.	Interrupt flag clear register (TSI_INTC) .....	592
21.4.4.	Interrupt flag register (TSI_INTF) .....	593
21.4.5.	Pin hysteresis mode register (TSI_PHM).....	594
21.4.6.	Analog switch register (TSI_ASW).....	594
21.4.7.	Sample configuration register (TSI_SAMP_CFG) .....	595
21.4.8.	Channel configuration register (TSI_CHCFG) .....	595
21.4.9.	Group control register (TSI_GCTL).....	596

21.4.10.	Group x cycle number registers (TSI_GxCYCN) (x= 0..5) .....	596
<b>22.</b>	<b>Universal Serial Bus full-speed device interface (USB D).....</b>	<b>598</b>
22.1.	Overview.....	598
22.2.	Main features.....	598
22.3.	Block diagram .....	598
22.4.	Signal description .....	599
22.5.	Clock configuration .....	599
22.6.	Function overview.....	600
22.6.1.	USB endpoints .....	600
22.6.2.	Operation procedure .....	602
22.6.3.	USB events and interrupts.....	605
22.6.4.	Operation guide.....	607
22.7.	Register definition.....	609
22.7.1.	USB D control register (USB D_CTL) .....	609
22.7.2.	USB D interrupt flag register (USB D_INTF) .....	610
22.7.3.	USB D status register (USB D_STAT).....	611
22.7.4.	USB D device address register (USB D_DADDR) .....	612
22.7.5.	USB D buffer address register (USB D_BADDR) .....	612
22.7.6.	USB D endpoint x control and status register (USB D_EPxCS), x=[0..7] .....	613
22.7.7.	USB D endpoint x transmission buffer address register (USB D_EPxTBADDR), x=[0..7] .....	615
22.7.8.	USB D endpoint x transmission buffer byte count register (USB D_EPxTBCNT), x=[0..7] .....	615
22.7.9.	USB D endpoint x reception buffer address register (USB D_EPxRBADDR), x=[0..7] .....	616
22.7.10.	USB D endpoint x reception buffer byte count register (USB D_EPxRBCNT), x=[0..7] .....	616
<b>23.</b>	<b>Segment LCD controller (SLCD).....</b>	<b>617</b>
23.1.	Overview.....	617
23.2.	Characteristics.....	617
23.3.	Function overview.....	617
23.3.1.	SLCD Architecture .....	617
23.3.2.	Clock generator .....	618
23.3.3.	Blink control.....	619
23.3.4.	SEG/COM Driver .....	619
23.3.5.	Double buffer memory .....	622
23.3.6.	ANALOG matrix.....	622
23.4.	Register definition.....	624
23.4.1.	Control register (SLCD_CTL).....	624
23.4.2.	Configuration register (SLCD_CFG) .....	625
23.4.3.	Status flag register (SLCD_STAT) .....	627
23.4.4.	Status flag clear register (SLCD_STATC).....	628
23.4.5.	Display data registers (SLCD_DATAx, x=0~7).....	629

<b>24. Operational amplifiers (OPA)/ Programmable current and voltage reference (IVREF)</b> .....	<b>630</b>
<b>24.1. Operational amplifiers (OPA)</b> .....	<b>630</b>
24.1.1. Overview.....	630
24.1.2. Characteristics .....	630
24.1.3. Function overview .....	630
24.1.4. Register definition .....	634
<b>24.2. Programmable Current and Voltage Reference (IVREF)</b> .....	<b>639</b>
24.2.1. Overview.....	639
24.2.2. Characteristics .....	639
24.2.3. Function overview .....	639
24.2.4. Register definition .....	641
<b>25. Controller area network (CAN)</b> .....	<b>643</b>
<b>25.1. Overview</b> .....	<b>643</b>
<b>25.2. Characteristics</b> .....	<b>643</b>
<b>25.3. Function overview</b> .....	<b>644</b>
25.3.1. Working mode .....	644
25.3.2. Communication modes .....	645
25.3.3. Data transmission .....	646
25.3.4. Data reception .....	648
25.3.5. Filtering Function.....	650
25.3.6. Time-triggered communication .....	653
25.3.7. Communication parameters .....	653
25.3.8. Error flags .....	655
25.3.9. CAN interrupts .....	656
25.3.10. CAN PHY mode .....	657
<b>25.4. Register definition</b> .....	<b>658</b>
25.4.1. Control register (CAN_CTL).....	658
25.4.2. Status register (CAN_STAT) .....	659
25.4.3. Transmit status register (CAN_TSTAT).....	661
25.4.4. Receive message FIFO0 register (CAN_RFIFO0).....	663
25.4.5. Receive message FIFO1 register (CAN_RFIFO1).....	664
25.4.6. Interrupt enable register (CAN_INTEN) .....	665
25.4.7. Error register (CAN_ERR) .....	667
25.4.8. Bit timing register (CAN_BT).....	668
25.4.9. Transmit mailbox identifier register (CAN_TMIX) (x=0..2) .....	669
25.4.10. Transmit mailbox property register (CAN_TMPx) (x=0..2).....	669
25.4.11. Transmit mailbox data0 register (CAN_TMDATA0x) (x=0..2) .....	670
25.4.12. Transmit mailbox data1 register (CAN_TMDATA1x) (x=0..2) .....	671
25.4.13. Receive FIFO mailbox identifier register (CAN_RFIFOMIX) (x=0,1) .....	671
25.4.14. Receive FIFO mailbox property register (CAN_RFIFOMPx) (x=0,1).....	672

25.4.15.	Receive FIFO mailbox data0 register (CAN_RFIFOMDATA0x) (x=0,1) .....	672
25.4.16.	Receive FIFO mailbox data1 register (CAN_RFIFOMDATA1x) (x=0,1) .....	673
25.4.17.	Filter control register (CAN_FCTL) .....	673
25.4.18.	Filter mode configuration register (CAN_FMCFG).....	674
25.4.19.	Filter scale configuration register (CAN_FSCFG).....	674
25.4.20.	Filter associated FIFO register (CAN_FAFIFO).....	675
25.4.21.	Filter working register (CAN_FW).....	675
25.4.22.	Filter x data y register (CAN_FxDATAy) (x=0..27, y=0,1) .....	676
25.4.23.	PHY control register (CAN_PHYCTL) .....	676
<b>26.</b>	<b>Revision history.....</b>	<b>678</b>

## List of Figures

Figure 1-1. The structure of the Cortex™-M3 processor.....	2
Figure 1-2. Series system architecture of GD32F130xx and GD32F150xx devices .....	3
Figure 1-3. Series system architecture of GD32F170xx and GD32F190xx devices .....	4
Figure 2-1. Process of page erase operation .....	25
Figure 2-2. Process of the mass erase operation .....	26
Figure 2-3. Process of the word programming operation.....	27
Figure 3-1. Power supply overview of GD32F130xx and GD32F150xx devices .....	40
Figure 3-2. Power supply overview of GD32F170xx and GD32F190xx devices .....	41
Figure 3-3. Waveform of the POR/PDR .....	43
Figure 3-4. Waveform of LVD threshold.....	44
Figure 4-1. The system reset circuit .....	52
Figure 4-2. Clock tree of GD32F130xx and GD32F150xx devices.....	53
Figure 4-3. Clock tree of GD32F170xx and GD32F190xx devices.....	54
Figure 4-4. HXTAL clock source .....	55
Figure 5-1. Block diagram of EXTI. ....	108
Figure 6-1. Basic structure of a standard I/O port bit .....	118
Figure 6-2. Input floating/pull up/pull down configurations .....	119
Figure 6-3. Output configuration .....	120
Figure 6-4. High impedance-analog configuration .....	121
Figure 6-5. Alternate function configuration .....	122
Figure 7-1. Block Diagram of CRC Calculation Unit.....	137
Figure 8-1. Block diagram of DMA .....	143
Figure 8-2. Handshake mechanism.....	145
Figure 8-3. DMA interrupt logic.....	148
Figure 8-4. DMA request mapping.....	149
Figure 10-1. ADC module block diagram of GD32F130xx and GD32F150xx devices .....	168
Figure 10-2. ADC module block diagram of GD32F170xx and GD32F190xx devices .....	169
Figure 10-3. Single conversion mode.....	171
Figure 10-4. Continuous conversion mode, scan disable .....	172
Figure 10-5. Scan conversion mode, continuous disable.....	173
Figure 10-6. Scan conversion mode, continuous enable.....	173
Figure 10-7. Discontinuous conversion mode.....	174
Figure 10-8. Auto-insertion, CTN = 1 .....	175
Figure 10-9. Triggered insertion .....	176
Figure 10-10. Data alignment of 12-bit resolution.....	177
Figure 10-11. Data alignment of 10-bit resolution.....	177
Figure 10-12. Data alignment of 8-bit resolution .....	177
Figure 10-13. Data alignment of 6-bit resolution .....	178
Figure 10-14. 20-bit to 16-bit result truncation.....	182
Figure 10-15. Numerical example with 5-bits shift and rounding .....	182

Figure 11-1. DAC block diagram .....	201
Figure 12-1. CMP block diagram of GD32F130xx and GD32F150xx devices .....	218
Figure 12-2. CMP block diagram of GD32F170xx and GD32F190xx devices .....	218
Figure 13-1. Free watchdog timer block diagram.....	229
Figure 13-2. Window watchdog timer block diagram.....	236
Figure 13-3. Window watchdog timer timing diagram .....	237
Figure 14-1. Block diagram of RTC .....	241
Figure 15-1. Advanced timer block diagram.....	267
Figure 15-2. Normal mode, internal clock divided by 1 .....	268
Figure 15-3. Counter timing diagram with prescaler division change from 1 to 2 .....	269
Figure 15-4. Up-counter timechart, PSC=0/1 .....	270
Figure 15-5. Up-counter timechart, change TIMERx_CAR on the go.....	271
Figure 15-6. Down-counter timechart, PSC=0/1.....	272
Figure 15-7. Down-counter timechart, change TIMERx_CAR on the go .....	273
Figure 15-8. Center-aligned counter timechart .....	274
Figure 15-9. Repetition timecart for center-aligned counter .....	275
Figure 15-10. Repetition timechart for up-counter.....	276
Figure 15-11. Repetition timechart for down-counter .....	276
Figure 15-12. Input capture logic.....	277
Figure 15-13. Output-compare under three modes .....	279
Figure 15-14. EAPWM timechart .....	280
Figure 15-15. CAPWM timechart.....	280
Figure 15-16. Complementary output with dead-time insertion.....	283
Figure 15-17. Output behavior in response to a break(The break high active).....	284
Figure 15-18. Example of counter operation in encoder interface mode.....	285
Figure 15-19. Example of encoder interface mode with CIOFE0 polarity inverted .....	285
Figure 15-20. Hall sensor is used to BLDC motor.....	286
Figure 15-21. Hall sensor timing between two timers.....	287
Figure 15-22. Restart mode.....	288
Figure 15-23. Pause mode.....	289
Figure 15-24. Event mode.....	289
Figure 15-25. Single pulse mode TIMERx_CHxCV = 0x04 TIMERx_CAR=0x60.....	290
Figure 15-26. TIMER0 Master/Slave mode timer example.....	291
Figure 15-27. Triggering TIMER0 with Enable of TIMER1 .....	292
Figure 15-28. Triggering TIMER0 with update signal of TIMER1 .....	292
Figure 15-29. Pause TIMER0 with enable of TIMER1.....	293
Figure 15-30. Pause TIMER0 with O0CPREof TIMER1 .....	294
Figure 15-31. Triggering TIMER0 and TIMER1 with TIMER1's CIO input .....	295
Figure 15-32. General Level 0 timer block diagram .....	322
Figure 15-33. Normal mode, internal clock divided by 1 .....	323
Figure 15-34. Counter timing diagram with prescaler division change from 1 to 2 .....	324
Figure 15-35. Up-counter timechart, PSC=0/1 .....	325
Figure 15-36. Up-counter timechart, change TIMERx_CAR on the go. ....	326
Figure 15-37. Down-counter timechart, PSC=0/1 .....	327

Figure 15-38. Down-counter timechart, change <code>TIMERx_CAR</code> on the go.....	327
Figure 15-39. Center-aligned counter timechart.....	329
Figure 15-40. Input capture logic.....	330
Figure 15-41. Output-compare under three modes .....	332
Figure 15-42. EAPWM timechart .....	333
Figure 15-43. CAPWM timechart.....	333
Figure 15-44. Example of counter operation in encoder interface mode.....	335
Figure 15-45. Example of encoder interface mode with <code>CI0FE0</code> polarity inverted .....	335
Figure 15-46. Restart mode.....	336
Figure 15-47. Pause mode.....	337
Figure 15-48. Event mode.....	337
Figure 15-49. Single pulse mode <code>TIMERx_CHxCV = 0x04</code> <code>TIMERx_CAR=0x60</code> .....	338
Figure 15-50. General level2 timer block diagram .....	365
Figure 15-51. Normal mode, internal clock divided by 1 .....	366
Figure 15-52. Counter timing diagram with prescaler division change from 1 to 2.....	367
Figure 15-53. Up-counter timechart, <code>PSC=0/1</code> .....	368
Figure 15-54. Up-counter timechart, change <code>TIMERx_CAR</code> on the go .....	368
Figure 15-55. Input capture logic.....	369
Figure 15-56. Output-compare under three modes .....	371
Figure 15-57. PWM mode timechart.....	372
Figure 15-58. General level3 timer block diagram .....	383
Figure 15-59. Normal mode, internal clock divided by 1 .....	384
Figure 15-60. Counter timing diagram with prescaler division change from 1 to 2.....	385
Figure 15-61. Up-counter timechart, <code>PSC=0/1</code> .....	386
Figure 15-62. Up-counter timechart, change <code>TIMERx_CAR</code> on the go .....	387
Figure 15-63. Repetition timechart for up-counter.....	388
Figure 15-64. Input capture logic.....	389
Figure 15-65. Output-compare under three modes .....	391
Figure 15-66. PWM mode timechart.....	392
Figure 15-67. Complementary output with dead-time insertion.....	394
Figure 15-68. Output behavior in response to a break(The break high active).....	395
Figure 15-69. Restart mode.....	396
Figure 15-70. Pause mode.....	397
Figure 15-71. Event mode.....	397
Figure 15-72. Single pulse mode <code>TIMERx_CHxCV = 0x04</code> <code>TIMERx_CAR=0x60</code> .....	398
Figure 15-73. General level4 timer block diagram .....	419
Figure 15-74. Normal mode, internal clock divided by 1 .....	420
Figure 15-75. Counter timing diagram with prescaler division change from 1 to 2.....	421
Figure 15-76. Up-counter timechart, <code>PSC=0/1</code> .....	422
Figure 15-77. Up-counter timechart, change <code>TIMERx_CAR</code> on the go .....	423
Figure 15-78. Repetition timechart for up-counter.....	424
Figure 15-79. Input capture logic.....	425
Figure 15-80. Output-compare under three modes .....	427
Figure 15-81. PWM mode timechart.....	428

Figure 15-82. Complementary output with dead-time insertion.....	430
Figure 15-83. Output behavior in response to a break(The break high active).....	431
Figure 15-84. Single pulse mode $TIMERx\_CHxCV = 0x04$ $TIMERx\_CAR=0x60$ .....	432
Figure 15-85. Basic timer block diagram.....	449
Figure 15-86. Normal mode, internal clock divided by 1.....	450
Figure 15-87. Counter timing diagram with prescaler division change from 1 to 2.....	451
Figure 15-88. Up-counter timechart, $PSC=0/1$ .....	452
Figure 15-89. Up-counter timechart, change $TIMERx\_CAR$ on the go.....	453
Figure 16-1. IFRP output timechart 1.....	459
Figure 16-2. IFRP output timechart 2.....	460
Figure 16-3. IFRP output timechart 3.....	460
Figure 17-1. USART module block diagram.....	464
Figure 17-2. USART character frame (9 bits data and 1 stop bit).....	464
Figure 17-3. USART transmit procedure.....	466
Figure 17-4. Oversampling method of a receive frame bit ( $OSB=0$ ).....	467
Figure 17-5. Configuration step when using DMA for USART transmission.....	469
Figure 17-6. Configuration step when using DMA for USART reception.....	470
Figure 17-7. Hardware flow control between two USARTs.....	470
Figure 17-8. Hardware flow control.....	471
Figure 17-9. Break frame occurs during idle state.....	472
Figure 17-10. Break frame occurs during a frame.....	473
Figure 17-11. Example of USART in synchronous mode.....	473
Figure 17-12. 8-bit format USART synchronous waveform ( $CLEN=1$ ).....	474
Figure 17-13. IrDA SIR ENDEC module.....	474
Figure 17-14. IrDA data modulation.....	475
Figure 17-15. ISO7816-3 frame format.....	476
Figure 17-16. USART interrupt mapping diagram.....	480
Figure 18-1. I2C module block diagram.....	500
Figure 18-2. Data validation.....	501
Figure 18-3. START and STOP condition.....	501
Figure 18-4. Clock synchronization.....	502
Figure 18-5. SDA Line arbitration.....	502
Figure 18-6. I2C communication flow with 7-bit address.....	503
Figure 18-7. I2C communication flow with 10-bit address.....	503
Figure 18-8. Programming model for slave transmitting.....	505
Figure 18-9. Programming model for slave receiving.....	506
Figure 18-10. Programming model for master transmitting.....	508
Figure 18-11. Programming model for master receiving using Solution A.....	510
Figure 18-12. Programming model for master receiving using Solution B.....	512
Figure 19-1. Block diagram of SPI for GD32F130xx and GD32F150xx devices.....	528
Figure 19-2. Block diagram of SPI for GD32F170xx and GD32F190xx devices.....	528
Figure 19-3. SPI timing diagram in normal mode.....	530
Figure 19-4. SPI timing diagram in Quad-SPI mode ( $CKPL=1$ , $CKPH=1$ , $LF=0$ ) for GD32F170xx and GD32F190xx devices.....	530

Figure 19-5. A typical Full-duplex connection .....	532
Figure 19-6. A typical simplex connection (Master: Receive, Slave: Transmit).....	533
Figure 19-7. A typical simplex connection (Master: Transmit only, Slave: Receive).....	533
Figure 19-8. A typical bidirectional connection .....	533
Figure 19-9. Timing diagram of quad write operation in Quad-SPI mode .....	536
Figure 19-10. Timing diagram of quad read operation in Quad-SPI mode .....	537
Figure 19-11. Block diagram of I2S .....	540
Figure 19-12. I2S Phillips standard timing diagram (DTLEN=00, CHLEN=0, CKPL=0) .....	541
Figure 19-13. I2S Phillips standard timing diagram (DTLEN=00, CHLEN=0, CKPL=1) .....	542
Figure 19-14. I2S Phillips standard timing diagram (DTLEN=10, CHLEN=1, CKPL=0) .....	542
Figure 19-15. I2S Phillips standard timing diagram (DTLEN=10, CHLEN=1, CKPL=1) .....	542
Figure 19-16. I2S Phillips standard timing diagram (DTLEN=01, CHLEN=1, CKPL=0) .....	542
Figure 19-17. I2S Phillips standard timing diagram (DTLEN=01, CHLEN=1, CKPL=1) .....	542
Figure 19-18. I2S Phillips standard timing diagram (DTLEN=00, CHLEN=1, CKPL=0) .....	543
Figure 19-19. I2S Phillips standard timing diagram (DTLEN=00, CHLEN=1, CKPL=1) .....	543
Figure 19-20. MSB justified standard timing diagram (DTLEN=00, CHLEN=0, CKPL=0) ..	543
Figure 19-21. MSB justified standard timing diagram (DTLEN=00, CHLEN=0, CKPL=1) ..	543
Figure 19-22. MSB justified standard timing diagram (DTLEN=10, CHLEN=1, CKPL=0) ..	544
Figure 19-23. MSB justified standard timing diagram (DTLEN=10, CHLEN=1, CKPL=1) ..	544
Figure 19-24. MSB justified standard timing diagram (DTLEN=01, CHLEN=1, CKPL=0) ..	544
Figure 19-25. MSB justified standard timing diagram (DTLEN=01, CHLEN=1, CKPL=1) ..	544
Figure 19-26. MSB justified standard timing diagram (DTLEN=00, CHLEN=1, CKPL=0) ..	544
Figure 19-27. MSB justified standard timing diagram (DTLEN=00, CHLEN=1, CKPL=1) ..	544
Figure 19-28. LSB justified standard timing diagram (DTLEN=01, CHLEN=1, CKPL=0) ..	545
Figure 19-29. LSB justified standard timing diagram (DTLEN=01, CHLEN=1, CKPL=1) ..	545
Figure 19-30. LSB justified standard timing diagram (DTLEN=00, CHLEN=1, CKPL=0) ..	545
Figure 19-31. LSB justified standard timing diagram (DTLEN=00, CHLEN=1, CKPL=1) ..	545
Figure 19-32. PCM standard short frame synchronization mode timing diagram (DTLEN=00, CHLEN=0, CKPL=0) .....	546
Figure 19-33. PCM standard short frame synchronization mode timing diagram (DTLEN=00, CHLEN=0, CKPL=1) .....	546
Figure 19-34. PCM standard short frame synchronization mode timing diagram (DTLEN=10, CHLEN=1, CKPL=0) .....	546
Figure 19-35. PCM standard short frame synchronization mode timing diagram (DTLEN=10, CHLEN=1, CKPL=1) .....	546
Figure 19-36. PCM standard short frame synchronization mode timing diagram (DTLEN=01, CHLEN=1, CKPL=0) .....	546
Figure 19-37. PCM standard short frame synchronization mode timing diagram (DTLEN=01, CHLEN=1, CKPL=1) .....	547
Figure 19-38. PCM standard short frame synchronization mode timing diagram (DTLEN=00, CHLEN=1, CKPL=0) .....	547
Figure 19-39. PCM standard short frame synchronization mode timing diagram (DTLEN=00, CHLEN=1, CKPL=1) .....	547
Figure 19-40. PCM standard long frame synchronization mode timing diagram (DTLEN=00,	

CHLEN=0, CKPL=0).....	547
Figure19-41. PCM standard long frame synchronization mode timing diagram (DTLEN=00, CHLEN=0, CKPL=1).....	547
Figure 19-42. PCM standard long frame synchronization mode timing diagram (DTLEN=10, CHLEN=1, CKPL=0).....	548
Figure 19-43. PCM standard long frame synchronization mode timing diagram (DTLEN=10, CHLEN=1, CKPL=1).....	548
Figure 19-44. PCM standard long frame synchronization mode timing diagram (DTLEN=01, CHLEN=1, CKPL=0).....	548
Figure 19-45. PCM standard long frame synchronization mode timing diagram (DTLEN=01, CHLEN=1, CKPL=1).....	548
Figure 19-46. PCM standard long frame synchronization mode timing diagram (DTLEN=00, CHLEN=1, CKPL=0).....	548
Figure 19-47. PCM standard long frame synchronization mode timing diagram (DTLEN=00, CHLEN=1, CKPL=1).....	549
Figure 19-48. Block diagram of I2S clock generator .....	549
Figure 20-1. HDMI-CEC Controller Block Diagram.....	566
Figure 21-1. Block diagram of TSI module.....	582
Figure 21-2. Block diagram of Sample pin and Channel Pin.....	583
Figure 21-3. Voltage of a sample pin during charge-transfer sequence .....	585
Figure 21-4. FSM flow of a charge-transfer sequence .....	586
Figure 22-1. USB D block diagram .....	598
Figure 22-2. An example with buffer descriptor table usage (USB D_BADDR = 0).....	601
Figure 23-1. SLCD Block Diagram.....	618
Figure 24-1. OPA0 Signal Route.....	631
Figure 24-2. OPA1 Signal Route.....	631
Figure 24-3. OPA2 Signal Route.....	631
Figure 25-1. CAN module block diagram.....	644
Figure 25-2. Transmission register .....	646
Figure 25-3. State of transmission mailbox.....	647
Figure 25-4. Reception register .....	649
Figure 25-5. 32-bit filter .....	650
Figure 25-6. 16-bit filter .....	650
Figure 25-7. 32-bit mask mode filter .....	650
Figure 25-8. 16-bit mask mode filter .....	650
Figure 25-9. 32-bit list mode filter.....	651
Figure 25-10. 16-bit list mode filter .....	651
Figure 25-11. The bit time .....	654
Figure 25-12. CAN PHY connection .....	657

## List of Tables

Table 1-1. Memory map of GD32F130xx and GD32F150xx devices .....	5
Table 1-2. Memory map of GD32F170xx and GD32F190xx devices .....	5
Table 1-3. Flash module organization .....	10
Table 1-4. Boot modes.....	10
Table 2-1. Base address and size for flash memory.....	23
Table 2-2. Option bytes .....	29
Table 2-3. OB_WP bit for pages protected .....	30
Table 3-1. Power saving mode summary.....	46
Table 4-1. Clock source select .....	58
Table 4-2. Clock source select .....	58
Table 4-3. Core domain voltage selected in Deep-sleep mode - .....	59
Table 4-4. Core domain voltage selected in Deep-sleep mode - .....	59
Table 5-1. NVIC exception types in Cotrex-M3 .....	105
Table 5-2. Interrupt vector table of GD32F130xx and GD32F150xx devices .....	105
Table 5-3. Interrupt vector table of GD32F170xx and GD32F190xx devices .....	106
Table 5-4. EXTI source of GD32F130xx and GD32F150xx devices .....	110
Table 5-5. EXTI source of GD32F170xx and GD32F190xx devices .....	110
Table 6-1. GPIO configuration table.....	117
Table 8-1. DMA transfer operation .....	144
Table 8-2. Interrupt events .....	147
Table 8-3. DMA requests for each channel .....	150
Table 10-1. ADC internal signals .....	167
Table 10-2. ADC pins definition of GD32F130xx and GD32F150xx devices.....	167
Table 10-3. ADC pins definition of GD32F170xx and GD32F190xx devices.....	168
Table 10-4. External trigger for regular channels of ADC .....	178
Table 10-5. External trigger for inserted channels of ADC .....	179
Table 10-6. t <sub>CONV</sub> timings depending on resolution.....	181
Table 10-7. Maximum output results vs N and M (Grayed values indicates truncation) ...	182
Table 11-1. DAC pins.....	201
Table 11-2. External triggers of DAC.....	202
Table 13-1. Min/max FWDGT timeout period at 40 kHz (IRC40K) .....	230
Table 13-2. Min-max timeout value at 36 MHz (fPCLK1) .....	237
Table 14-1. RTC power saving mode management .....	251
Table 14-2. RTC interrupts control.....	251
Table 15-1. Timers (TIMERx) are divided into six sorts.....	265
Table 15-2. Complementary outputs controlled by parameters .....	281
Table 15-3. Counting direction versus encoder signals.....	284
Table 15-4. Slave mode example table.....	288
Table 15-5. Counting direction versus encoder signals.....	334
Table 15-6. Counting direction versus encoder signals.....	336

Table 15-7. TIMERx(x=1,2) interconnection .....	338
Table 15-8. Complementary outputs controlled by parameters .....	393
Table 15-9. Slave mode example table .....	396
Table 15-10. TIMERx(x=14) interconnection .....	398
Table 15-11. Complementary outputs controlled by parameters .....	429
Table 17-1. USART important pins description .....	463
Table 17-2. Stop bits configuration .....	464
Table 17-3. USART interrupt requests .....	479
Table 18-1. Definition of I2C-bus terminology (refer to the I2C specification of Philips semiconductors) .....	500
Table 18-2. Event status flags .....	515
Table 18-3. I2C error flags .....	515
Table 19-1. SPI signal description .....	529
Table 19-2. Quad-SPI signal description .....	529
Table 19-3. SPI operation modes .....	531
Table 19-4. SPI interrupt requests .....	540
Table 19-5. I2S bitrate calculation formulas .....	549
Table 19-6. Audio sampling frequency calculation formulas .....	550
Table 19-7. Direction of I2S interface signals for each operation mode .....	550
Table 19-8. I2S interrupt .....	554
Table 20-1. Data Bit Timing Parameter Table .....	567
Table 20-2. Error Handling Timing Parameter Table .....	571
Table 20-3. TERR Timing Parameter Table .....	572
Table 21-1. Pin and analog switch state in a charge-transfer sequence .....	584
Table 21-2. Duration time of Extend Charge state in each cycle .....	587
Table 21-3. TSI errors and flags .....	588
Table 21-4. TSI pins .....	589
Table 22-1. USB D signal description .....	599
Table 22-2. Double-buffering buffer flag definition .....	602
Table 22-3. Double buffer usage .....	602
Table 22-4. Reception status encoding .....	614
Table 22-5. Endpoint type encoding .....	614
Table 22-6. Endpoint kind meaning .....	614
Table 22-7. Transmission status encoding .....	615
Table 23-1. The odd frame voltage .....	619
Table 23-2. The even frame voltage .....	620
Table 23-3. The all common signal driver .....	620
Table 24-1. Operating mode and calibration .....	632
Table 25-1. 32-bit filter number .....	651
Table 25-2. Filtering index .....	652
Table 26-1. Revision history .....	678

## 1. System and memory architecture

The GD32F1x0 series are 32-bit general-purpose microcontrollers based on the ARM® Cortex™-M3 processor. The Cortex™-M3 processor includes three AHB buses known as I-Code, D-Code and System buses. All memory accesses of the Cortex™-M3 processor are executed on the three buses according to the different purposes and the target memory spaces. The memory organization uses a Harvard architecture, pre-defined memory map and up to 4 GB of memory space, making the system flexible and extendable.

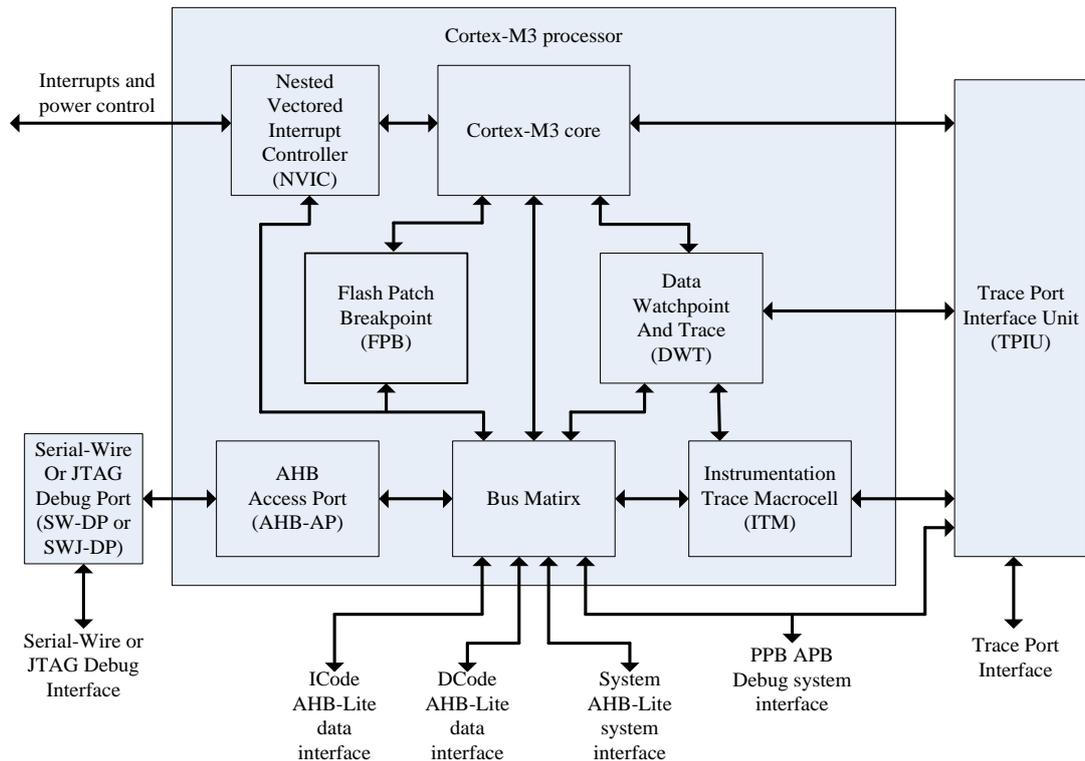
### 1.1. ARM Cortex-M3 processor

The Cortex™-M3 processor is a 32-bit processor including the features of low interrupt latency and low-cost debug. Integrated and advanced features make the Cortex™-M3 processor suitable for market products that require microcontrollers with high performance and low power consumption. The Cortex™-M3 processor is based on the ARMv7 architecture and supports a powerful and scalable instruction set including general data processing I/O control tasks and advanced data processing bit field manipulations. Some system peripherals listed below are also provided by Cortex™-M3:

- Internal Bus Matrix connected with I-Code bus, D-Code bus, System bus, Private Peripheral Bus (PPB) and debug accesses (AHB-AP)
- Nested Vectored Interrupt Controller (NVIC)
- Flash Patch and Breakpoint (FPB)
- Data Watchpoint and Trace (DWT)
- Instrumentation Trace Macrocell (ITM)
- Serial Wire JTAG Debug Port (SWJ-DP)
- Trace Port Interface Unit (TPIU)

The following figure shows the Cortex™-M3 processor block diagram. For more information, refer to the ARM® Cortex™-M3 Technical Reference Manual.

Figure 1-1. The structure of the Cortex™-M3 processor



## 1.2. System architecture

The system architecture of GD32F1x0 series is shown in the following figure. The AHB matrix based on AMBA 3.0 AHB-LITE is a multi-layer AHB, which enables parallel access paths between multiple masters and slaves in the system. There are four masters on the AHB matrix, including I-Code, D-Code, system bus of the Cortex™-M3 core and DMA. The I-Code bus is the instruction bus and also used for vector fetches from the Code region (0x0000 0000 ~ 0x1FFF FFFF) to the Cortex™-M3 core. The D-Code bus is used for loading/storing data and also for debugging access of the Code region. Similarly, the System bus is used for instruction/vector fetches, data loading/storing and debugging access of the system regions. The System regions include the internal SRAM region and the Peripheral region. The AHB matrix consists of five slaves, including I-Code and D-Code interfaces of the flash memory controller, internal SRAM, AHB1 and AHB2.

The AHB2 connects with the GPIO ports. The AHB1 connects with the AHB peripherals including two AHB-to-APB bridges which provide full synchronous connections between the AHB1 and the two APB buses. The two APB buses connect with all the APB peripherals.

Figure 1-2. Series system architecture of GD32F130xx and GD32F150xx devices

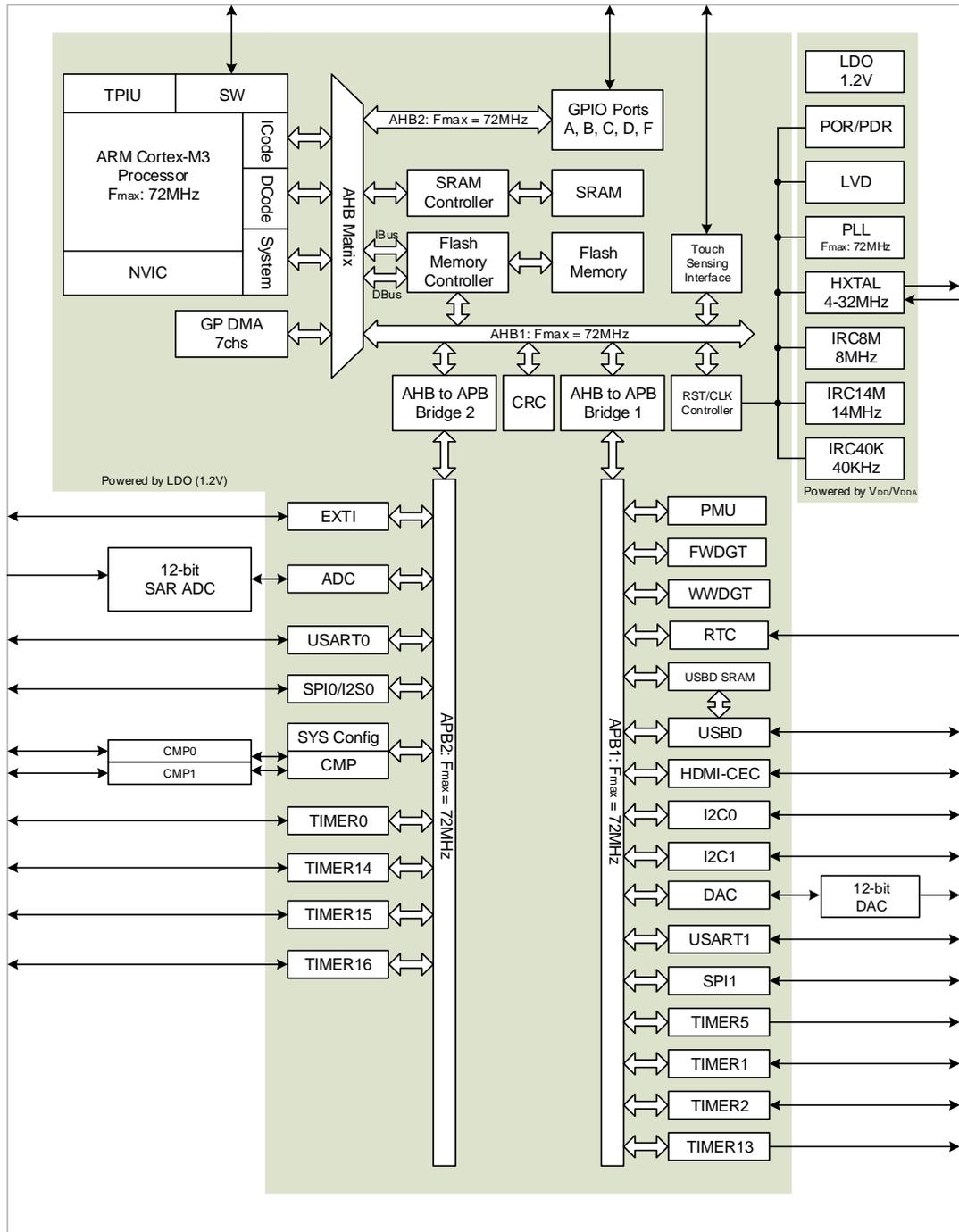
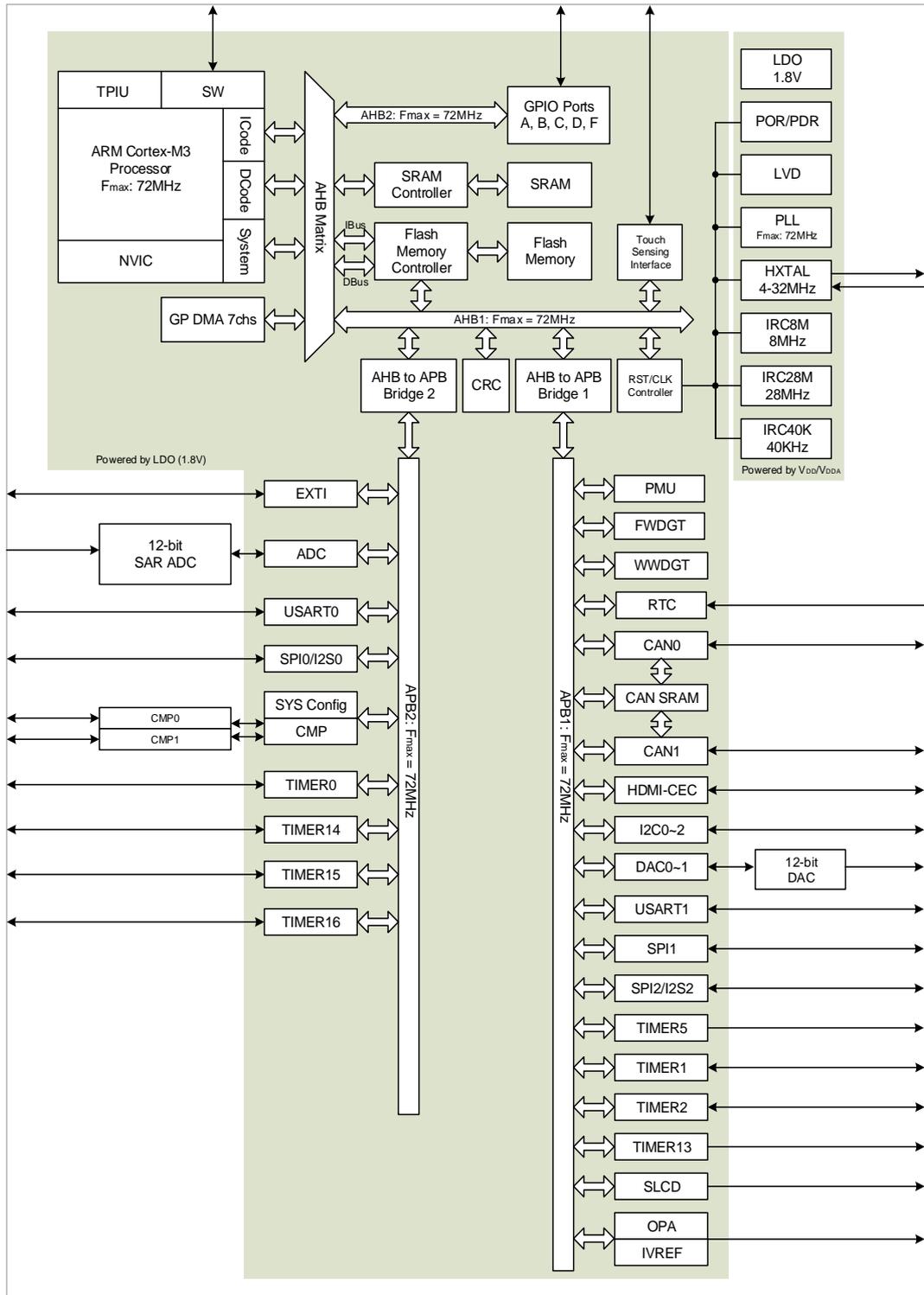


Figure 1-3. Series system architecture of GD32F170xx and GD32F190xx devices



### 1.3. Memory map

The ARM<sup>®</sup> Cortex<sup>™</sup>-M3 processor is structured in Harvard architecture which can use separate buses to fetch instructions and load/store data. The instruction code and data are

both located in the same memory address space but in different address ranges. Program memory, data memory, registers and I/O ports are organized within the same linear 4-Gbyte address space which is the maximum address range of the Cortex™-M3 since it has a 32-bit bus address width. Additionally, a pre-defined memory map is provided by the Cortex™-M3 processor to reduce the software complexity of repeated implementation of different device vendors. However, some regions are used by the ARM® Cortex™-M3 system peripherals. The following figure shows the memory map of GD32F1x0 series, including Code, SRAM, peripheral, and other pre-defined regions. Each peripheral of either type is allocated 1KB of space. This allows simplifying the address decoding for each peripheral.

**Table 1-1. Memory map of GD32F130xx and GD32F150xx devices**

Pre-defined Regions	Bus	ADDRESS	Peripherals
		0xE000 0000 - 0xE00F FFFF	Cortex-M3 internal peripherals
External Device		0xA000 0000 - 0xDFFF FFFF	Reserved
External RAM		0x6000 0000 - 0x9FFF FFFF	Reserved
Peripherals	AHB1	0x5000 0000 - 0x5FFF FFFF	Reserved
	AHB2	0x4800 1800 - 0x4FFF FFFF	Reserved
		0x4800 1400 - 0x4800 17FF	GPIOF
		0x4800 1000 - 0x4800 13FF	Reserved
		0x4800 0C00 - 0x4800 0FFF	GPIOD
		0x4800 0800 - 0x4800 0BFF	GPIOC
		0x4800 0400 - 0x4800 07FF	GPIOB
		0x4800 0000 - 0x4800 03FF	GPIOA
	AHB1	0x4002 4400 - 0x47FF FFFF	Reserved
		0x4002 4000 - 0x4002 43FF	TSI
		0x4002 3400 - 0x4002 3FFF	Reserved
		0x4002 3000 - 0x4002 33FF	CRC
		0x4002 2400 - 0x4002 2FFF	Reserved
		0x4002 2000 - 0x4002 23FF	FMC
		0x4002 1400 - 0x4002 1FFF	Reserved
		0x4002 1000 - 0x4002 13FF	RCU
		0x4002 0400 - 0x4002 0FFF	Reserved
		0x4002 0000 - 0x4002 03FF	DMA
	APB2	0x4001 4C00 - 0x4001 FFFF	Reserved
		0x4001 4800 - 0x4001 4BFF	TIMER16
		0x4001 4400 - 0x4001 47FF	TIMER15
		0x4001 4000 - 0x4001 43FF	TIMER14
		0x4001 3C00 - 0x4001 3FFF	Reserved
		0x4001 3800 - 0x4001 3BFF	USART0
		0x4001 3400 - 0x4001 37FF	Reserved
		0x4001 3000 - 0x4001 33FF	SPI0/I2S0
	0x4001 2C00 - 0x4001 2FFF	TIMER0	

Pre-defined Regions	Bus	ADDRESS	Peripherals
		0x4001 2800 - 0x4001 2BFF	Reserved
		0x4001 2400 - 0x4001 27FF	ADC
		0x4001 0800 - 0x4001 23FF	Reserved
		0x4001 0400 - 0x4001 07FF	EXTI
		0x4001 0000 - 0x4001 03FF	SYSCFG+CMP
	APB1	0x4000 C400 - 0x4000 FFFF	Reserved
		0x4000 C000 - 0x4000 C3FF	Reserved
		0x4000 8000 - 0x4000 BFFF	Reserved
		0x4000 7C00 - 0x4000 7FFF	Reserved
		0x4000 7800 - 0x4000 7BFF	CEC
		0x4000 7400 - 0x4000 77FF	DAC
		0x4000 7000 - 0x4000 73FF	PMU
		0x4000 6400 - 0x4000 6FFF	Reserved
		0x4000 6000 - 0x4000 63FF	USB SRAM
		0x4000 5C00 - 0x4000 5FFF	USB registers
		0x4000 5800 - 0x4000 5BFF	I2C1
		0x4000 5400 - 0x4000 57FF	I2C0
		0x4000 4800 - 0x4000 53FF	Reserved
		0x4000 4400 - 0x4000 47FF	USART1
		0x4000 4000 - 0x4000 43FF	Reserved
		0x4000 3C00 - 0x4000 3FFF	Reserved
		0x4000 3800 - 0x4000 3BFF	SPI1
		0x4000 3400 - 0x4000 37FF	Reserved
		0x4000 3000 - 0x4000 33FF	FWDGT
		0x4000 2C00 - 0x4000 2FFF	WWDGT
		0x4000 2800 - 0x4000 2BFF	RTC
		0x4000 2400 - 0x4000 27FF	Reserved
		0x4000 2000 - 0x4000 23FF	TIMER13
		0x4000 1400 - 0x4000 1FFF	Reserved
		0x4000 1000 - 0x4000 13FF	TIMER5
		0x4000 0800 - 0x4000 0FFF	Reserved
		0x4000 0400 - 0x4000 07FF	TIMER2
0x4000 0000 - 0x4000 03FF	TIMER1		
SRAM		0x2000 2000 - 0x3FFF FFFF	Reserved
		0x2000 0000 - 0x2000 1FFF	SRAM
Code		0x1FFF F810 - 0x1FFF FFFF	Reserved
		0x1FFF F800 - 0x1FFF F80F	Option bytes
		0x1FFF EC00 - 0x1FFF F7FF	System memory
		0x0801 0000 - 0x1FFF EBFF	Reserved
		0x0800 0000 - 0x0800 FFFF	Main Flash memory

Pre-defined Regions	Bus	ADDRESS	Peripherals
		0x0000 0000 - 0x07FF FFFF	Aliased to Flash or system memory

**Table 1-2. Memory map of GD32F170xx and GD32F190xx devices**

Pre-defined Regions	Bus	ADDRESS	Peripherals
		0xE000 0000 - 0xE00F FFFF	Cortex-M3 internal peripherals
External Device		0xA000 0000 - 0xDFFF FFFF	Reserved
External RAM		0x6000 0000 - 0x9FFF FFFF	Reserved
Peripherals	AHB1	0x5000 0000 - 0x5FFF FFFF	Reserved
	AHB2	0x4800 1800 - 0x4FFF FFFF	Reserved
		0x4800 1400 - 0x4800 17FF	GPIOF
		0x4800 1000 - 0x4800 13FF	Reserved
		0x4800 0C00 - 0x4800 0FFF	GPIOD
		0x4800 0800 - 0x4800 0BFF	GPIOC
		0x4800 0400 - 0x4800 07FF	GPIOB
		0x4800 0000 - 0x4800 03FF	GPIOA
	AHB1	0x4002 4400 - 0x47FF FFFF	Reserved
		0x4002 4000 - 0x4002 43FF	TSI
		0x4002 3400 - 0x4002 3FFF	Reserved
		0x4002 3000 - 0x4002 33FF	CRC
		0x4002 2400 - 0x4002 2FFF	Reserved
		0x4002 2000 - 0x4002 23FF	FMC
		0x4002 1400 - 0x4002 1FFF	Reserved
		0x4002 1000 - 0x4002 13FF	RCU
		0x4002 0400 - 0x4002 0FFF	Reserved
		0x4002 0000 - 0x4002 03FF	DMA
	APB2	0x4001 4C00 - 0x4001 FFFF	Reserved
		0x4001 4800 - 0x4001 4BFF	TIMER16
		0x4001 4400 - 0x4001 47FF	TIMER15
		0x4001 4000 - 0x4001 43FF	TIMER14
		0x4001 3C00 - 0x4001 3FFF	Reserved
		0x4001 3800 - 0x4001 3BFF	USART0
		0x4001 3400 - 0x4001 37FF	Reserved
		0x4001 3000 - 0x4001 33FF	SPI0/I2S0
		0x4001 2C00 - 0x4001 2FFF	TIMER0
		0x4001 2800 - 0x4001 2BFF	Reserved
		0x4001 2400 - 0x4001 27FF	ADC
		0x4001 0800 - 0x4001 23FF	Reserved
		0x4001 0400 - 0x4001 07FF	EXTI
		0x4001 0000 - 0x4001 03FF	SYSCFG+CMP

Pre-defined Regions	Bus	ADDRESS	Peripherals
	APB1	0x4000 C400 - 0x4000 FFFF	Reserved
		0x4000 C000 - 0x4000 C3FF	I2C2
		0x4000 8000 - 0x4000 BFFF	Reserved
		0x4000 7C00 - 0x4000 7FFF	OPAMP+IVREF
		0x4000 7800 - 0x4000 7BFF	CEC
		0x4000 7400 - 0x4000 77FF	DAC0~1
		0x4000 7000 - 0x4000 73FF	PMU
		0x4000 6C00 - 0x4000 6FFF	Reserved
		0x4000 6800 - 0x4000 6BFF	CAN1
		0x4000 6400 - 0x4000 67FF	CAN0
		0x4000 6000 - 0x4000 63FF	CAN SRAM
		0x4000 5C00 - 0x4000 5FFF	Reserved
		0x4000 5800 - 0x4000 5BFF	I2C1
		0x4000 5400 - 0x4000 57FF	I2C0
		0x4000 4800 - 0x4000 53FF	Reserved
		0x4000 4400 - 0x4000 47FF	USART1
		0x4000 4000 - 0x4000 43FF	Reserved
		0x4000 3C00 - 0x4000 3FFF	SPI2/I2S2
		0x4000 3800 - 0x4000 3BFF	SPI1
		0x4000 3400 - 0x4000 37FF	Reserved
		0x4000 3000 - 0x4000 33FF	FWDGT
		0x4000 2C00 - 0x4000 2FFF	WWDGT
		0x4000 2800 - 0x4000 2BFF	RTC
		0x4000 2400 - 0x4000 27FF	SLCD
		0x4000 2000 - 0x4000 23FF	TIMER13
		0x4000 1400 - 0x4000 1FFF	Reserved
		0x4000 1000 - 0x4000 13FF	TIMER5
		0x4000 0800 - 0x4000 0FFF	Reserved
0x4000 0400 - 0x4000 07FF	TIMER2		
0x4000 0000 - 0x4000 03FF	TIMER1		
SRAM		0x2000 5000 - 0x3FFF FFFF	Reserved
		0x2000 0000 - 0x2000 4FFF	SRAM
Code		0x1FFF F80F - 0x1FFF FFFF	Reserved
		0x1FFF F800 - 0x1FFF F80E	Option bytes
		0x1FFF EC00 - 0x1FFF F7FF	System memory
		0x0800 FFFF - 0x1FFF EBFF	Reserved
		0x0800 0000 - 0x0800 FFFE	Main Flash memory
		0x0000 0000 - 0x07FF FFFF	Aliased to Flash or system memory

### 1.3.1. Bit-banding

In order to reduce the time of read-modify-write operations, the Cortex™-M3 processor provides a bit-banding function to perform a single atomic bit operation. The memory map includes two bit-band regions. These occupy the SRAM and Peripherals respectively. These bit-band regions map each word in an alias region of memory to a bit in a bit-band region of memory.

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

$$\text{bit\_word\_addr} = \text{bit\_band\_base} + (\text{byte\_offset} \times 32) + (\text{bit\_number} \times 4) \quad (1-1)$$

where:

- bit\_word\_addr is the address of the word in the alias memory region that maps to the targeted bit.
- bit\_band\_base is the starting address of the alias region.
- byte\_offset is the number of the byte in the bit-band region that contains the targeted bit.
- bit\_number is the bit position (0-7) of the targeted bit.

For example, to access bit 7 of address 0x2000 0200, the bit-band alias is:

$$\text{bit\_word\_addr} = 0x2200\ 0000 + (0x200 * 32) + (7 * 4) = 0x2200\ 401C \quad (1-2)$$

Writing to address 0x2200 401C will cause bit 7 of address 0x2000 0200 change while a read to address 0x2200 401C will return 0x01 or 0x00 according to the value of bit 7 at the SRAM address 0x2000 0200.

### 1.3.2. On-chip SRAM memory

The GD32F1x0 series contain up to 8KB of on-chip SRAM which starts at the address 0x2000 0000. It supports byte, half-word (16 bits), and word (32 bits) accesses. In order to increase memory robustness, parity check is supported. The user can enable the parity check function using the bit SRAM\_PARITY\_CHECK in the user option bytes (refer to Chapter 2.3.9 [Option bytes description](#)). When enabled, an NMI is generated if the parity check fails. The SRAM parity check error flag is implemented in the system configuration register 2 (SYSCFG\_CFG2). The error flag can be connected to the break input of TIMER 0/ TIMER 14/ TIMER 15/ TIMER 16, if the SRAM\_PARITY\_ERROR\_LOCK control bit in the system configuration register 2 (SYSCFG\_CFG2) is set to 1.

The real data width of the SRAM is 36 bits, including 32 bits for data and 4 bits for parity (1 bit per byte). When writing, the parity bits are computed and stored into the SRAM. When reading, the parity bits are also computed using the stored data in SRAM. The computed parity bits are compared with the stored parity bits which are computed during the writing access. If they are different, the parity check fails.

**Note:** When enabling the SRAM parity check, it is recommended to initialize the whole SRAM memory by software at the beginning of the code, in order to avoid getting parity check errors

when reading non-initialized locations.

### 1.3.3. On-chip Flash memory

The devices provide up to 64 KB of on-chip flash memory. The flash memory consists of up to 64 KB main flash organized into 64 pages with 1 KB capacity per page and a 3 KB information block for the boot loader. The following table shows details.

**Table 1-3. Flash module organization**

Block	Name	Address	Size
Main Flash Block	Page 0	0x0800 0000 - 0x0800 03FF	1 Kbytes
	Page 1	0x0800 0400 - 0x0800 07FF	1 Kbytes
	Page 2	0x0800 0800 - 0x0800 0BFF	1 Kbytes
	.	.	.
	.	.	.
	Page 63	0x0800 FC00 - 0x0800 FFFF	1 Kbytes
Information Block	System memory	0x1FFF EC00 - 0x1FFF F7FF	3 Kbytes
Option Bytes Block	Option Bytes	0x1FFF F800 - 0x1FFF F80F	16 bytes

Read accesses to the preceding 32 pages can be performed 32 bits per cycle without any wait state. All of byte, half-word (16 bits) and word (32 bits) read accesses are supported. The flash memory can be programmed half-word (16 bits) or word (32 bits) at a time. Each page of the flash memory can be erased individually. The whole flash memory space except information blocks can be erased at a time.

## 1.4. Boot configuration

The GD32F1x0 series provides three kinds of boot sources which can be selected using the bit BOOT1\_n in the user option bytes (refer to Chapter 2.3.9 [Option bytes description](#)) and the BOOT0 pins. The value on the BOOT0 pin is latched on the 4th rising edge of SYSCLK after a reset. It is up to the user to set the BOOT1\_n and BOOT0 after a power-on reset or a system reset to select the required boot source. The details are shown in the following table.

**Table 1-4. Boot modes**

Selected boot source	Boot mode selection pins	
	BOOT1	BOOT0
Main Flash Memory	x	0
System Memory	0	1
On-chip SRAM	1	1

**Note:** The BOOT1 value is the opposite of the BOOT1\_n value.

After power-on sequence or a system reset, the ARM® Cortex™-M3 processor fetches the top-of-stack value from address 0x0000 0000 and the base address of boot code from 0x0000 0004 in sequence. Then, it starts executing code from the base address of boot code.

According to the selected boot source, either the main flash memory (original memory space beginning at 0x0800 0000) or the system memory (original memory space beginning at 0x1FFF EC00) is aliased in the boot memory space which begins at the address 0x0000 0000. When the on-chip SRAM whose memory space is beginning at 0x2000 0000 is selected as the boot source, in the application initialization code, you have to relocate the vector table in SRAM using the NVIC exception table and offset register.

The embedded boot loader is located in the System memory, which is used to reprogram the Flash memory. The boot loader can be activated through one of the following serial interfaces: USART0 or USART1.

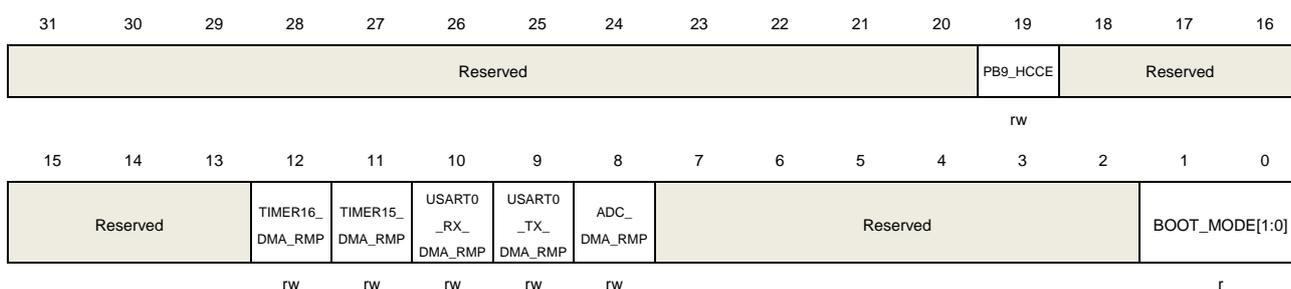
## 1.5. System configuration registers (SYSCFG)

### 1.5.1. System configuration register 0 (SYSCFG\_CFG0)

Address offset: 0x00

Reset value: 0x0000 000X (X indicates BOOT\_MODE[1:0] may be any value according to the BOOT0 pin and the BOOT1\_n option bit after reset)

This register has to be accessed by word(32-bit)



Bits	Fields	Descriptions
31:20	Reserved	Must be kept at reset value
19	PB9_HCCE	PB9 pin high current capability enable When it is set, the PB9 pin can be used to control an infrared LED directly. 0: High current capability on the PB9 pin is disabled. 1: High current capability on the PB9 pin is enabled, and the speed control of the pin is bypassed.
18:13	Reserved	Must be kept at reset value
12	TIMER16_DMA_RM P	TIMER 16 DMA request remapping enable 0: Not remap (TIMER16_CH0 and TIMER16_UP DMA requests are mapped on DMA channel 0) 1: Remap (TIMER16_CH0 and TIMER16_UP DMA requests are mapped on DMA channel 1)
11	TIMER15_DMA_RM P	TIMER 15 DMA request remapping enable 0: Not remap (TIMER15_CH0 and TIMER15_UP DMA requests are mapped on DMA channel 2) 1: Remap (TIMER15_CH0 and TIMER15_UP DMA requests are mapped on DMA channel 3)
10	USART0_RX_DMA_RMP	USART0_RX DMA request remapping enable 0: Not remap (USART0_RX DMA requests are mapped on DMA channel 2) 1: Remap (USART0_RX DMA requests are mapped on DMA channel 4)
9	USART0_TX_DMA_RMP	USART0_TX DMA request remapping enable 0: Not remap (USART0_TX DMA requests are mapped on DMA channel 1)

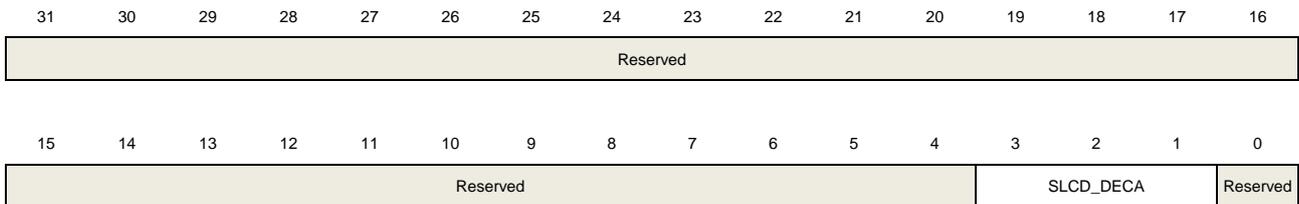
		1: Remap (USART0_TX DMA requests are mapped on DMA channel 3)
8	ADC_DMA_RMP	ADC DMA request remapping enable 0: Not remap (ADC DMA requests are mapped on DMA channel 0) 1: Remap (ADC DMA requests are mapped on DMA channel 1)
7:2	Reserved	Must be kept at reset value
1:0	BOOT_MODE[1:0]	Boot mode (Refer to Chapter 1.4 <a href="#">Boot configuration</a> for details) Bit0 is mapping to the BOOT0 pin; the value of bit1 is the opposite of the BOOT1_n option bit value. x0: Boot from the Main Flash 01: Boot from the system memory 11: Boot from the embedded SRAM

## 1.5.2. System configuration register 1 (SYSCFG\_CFG1)

Address offset: 0x04

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)



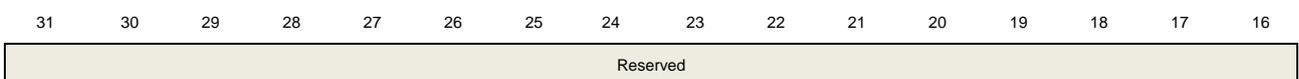
Bits	Fields	Descriptions
31:4	Reserved	Must be kept at reset value
3:1	SLCD_DECA	Decoupling capacitance connection for SLCD Bit1: Decoupling capacitance connection to PB2 or not Bit2: Decoupling capacitance connection to PB12 or not Bit3: Decoupling capacitance connection to PB0 or not
0	Reserved	Must be kept at reset value

## 1.5.3. EXTI sources selection register 0 (SYSCFG\_EXTISS0)

Address offset: 0x08

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)



15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI3_SS [3:0]				EXTI2_SS [3:0]				EXTI1_SS [3:0]				EXTI0_SS [3:0]			
rw				rw				rw				rw			

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:12	EXTI3_SS[3:0]	EXTI 3 sources selection X000: PA3 pin X001: PB3 pin X010: PC3 pin X011: Reserved X100: Reserved X101: Reserved X110: Reserved X111: Reserved
11:8	EXTI2_SS[3:0]	EXTI 2 sources selection X000: PA2 pin X001: PB2 pin X010: PC2 pin X011: PD2 pin X100: Reserved X101: Reserved X110: Reserved X111: Reserved
7:4	EXTI1_SS[3:0]	EXTI 1 sources selection X000: PA1 pin X001: PB1 pin X010: PC1 pin X011: Reserved X100: Reserved X101: PF1 pin X110: Reserved X111: Reserved
3:0	EXTI0_SS[3:0]	EXTI 0 sources selection X000: PA0 pin X001: PB0 pin X010: PC0 pin X011: Reserved X100: Reserved X101: PF0 pin X110: Reserved

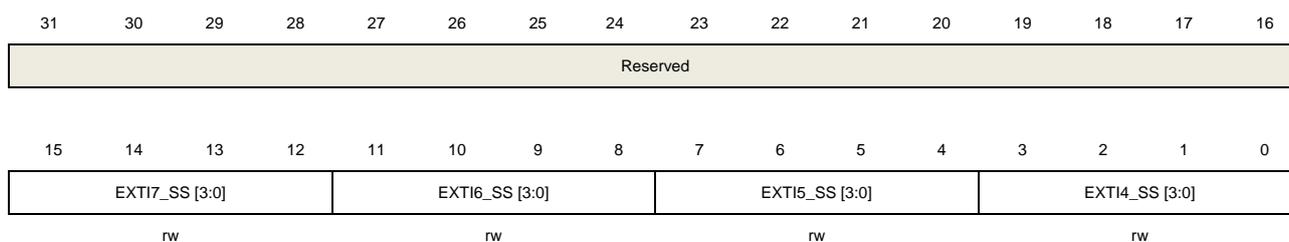
X111: Reserved

### 1.5.4. EXTI sources selection register 1 (SYSCFG\_EXTISS1)

Address offset: 0x0C

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)



Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:12	EXTI7_SS[3:0]	EXTI 7 sources selection X000: PA7 pin X001: PB7 pin X010: PC7 pin X011: Reserved X100: Reserved X101: PF7 pin X110: Reserved X111: Reserved
11:8	EXTI6_SS[3:0]	EXTI 6 sources selection X000: PA6 pin X001: PB6 pin X010: PC6 pin X011: Reserved X100: Reserved X101: PF6 pin X110: Reserved X111: Reserved
7:4	EXTI5_SS[3:0]	EXTI 5 sources selection X000: PA5 pin X001: PB5 pin X010: PC5 pin X011: Reserved X100: Reserved X101: PF5 pin

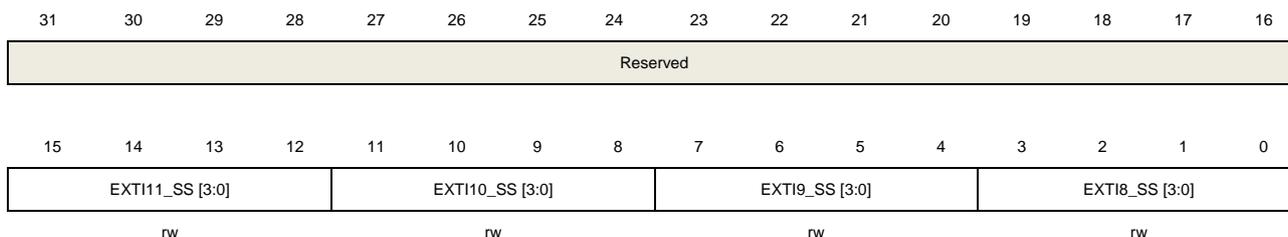
		X110: Reserved
		X111: Reserved
3:0	EXTI4_SS[3:0]	EXTI 4 sources selection
		X000: PA4 pin
		X001: PB4 pin
		X010: PC4 pin
		X011: Reserved
		X100: Reserved
		X101: PF4 pin
		X110: Reserved
		X111: Reserved

### 1.5.5. EXTI sources selection register 2 (SYSCFG\_EXTISS2)

Address offset: 0x10

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)



Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:12	EXTI11_SS[3:0]	EXTI 11 sources selection X000: PA11 pin X001: PB11 pin X010: PC11 pin X011: Reserved X100: Reserved X101: Reserved X110: Reserved X111: Reserved
11:8	EXTI10_SS[3:0]	EXTI 10 sources selection X000: PA10 pin X001: PB10 pin X010: PC10 pin X011: Reserved X100: Reserved

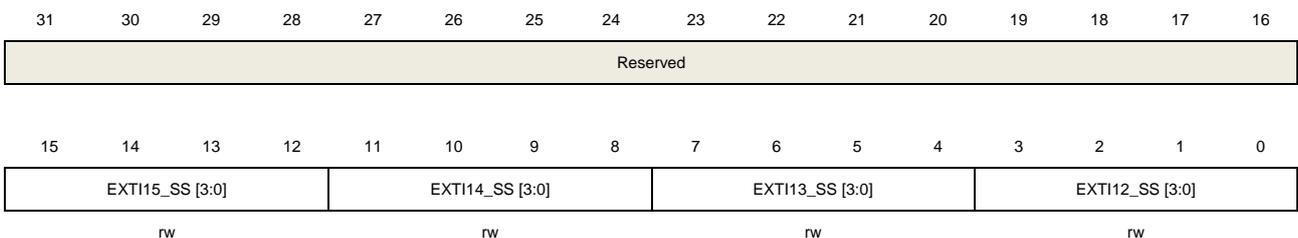
		X101: Reserved
		X110: Reserved
		X111: Reserved
7:4	EXTI9_SS[3:0]	EXTI 9 sources selection X000: PA9 pin X001: PB9 pin X010: PC9 pin X011: Reserved X100: Reserved X101: Reserved X110: Reserved X111: Reserved
3:0	EXTI8_SS[3:0]	EXTI 8 sources selection X000: PA8 pin X001: PB8 pin X010: PC8 pin X011: Reserved X100: Reserved X101: Reserved X110: Reserved X111: Reserved

### 1.5.6. EXTI sources selection register 3 (SYSCFG\_EXTISS3)

Address offset: 0x14

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)



Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:12	EXTI15_SS[3:0]	EXTI 15 sources selection X000: PA15 pin X001: PB15 pin X010: PC15 pin X011: Reserved

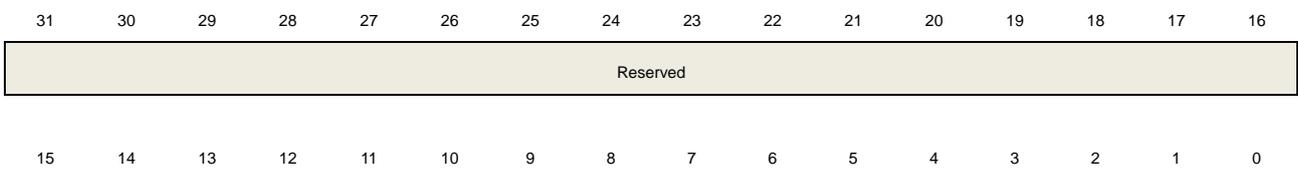
		X100: Reserved
		X101: Reserved
		X110: Reserved
		X111: Reserved
11:8	EXTI14_SS[3:0]	EXTI 14 sources selection
		X000: PA14 pin
		X001: PB14 pin
		X010: PC14 pin
		X011: Reserved
		X100: Reserved
		X101: Reserved
		X110: Reserved
		X111: Reserved
7:4	EXTI13_SS[3:0]	EXTI 13 sources selection
		X000: PA13 pin
		X001: PB13 pin
		X010: PC13 pin
		X011: Reserved
		X100: Reserved
		X101: Reserved
		X110: Reserved
		X111: Reserved
3:0	EXTI12_SS[3:0]	EXTI 12 sources selection
		X000: PA12 pin
		X001: PB12 pin
		X010: PC12 pin
		X011: Reserved
		X100: Reserved
		X101: Reserved
		X110: Reserved
		X111: Reserved

### 1.5.7. System configuration register 2 (SYSCFG\_CFG2)

Address offset: 0x18

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)



Reserved	SRAM_P CEF	Reserved	LVD_ LOCK	SRAM_ PARITY_ ERROR_ LOCK	LOCK UP_ LOCK
	rw		rw	rw	rw

Bits	Fields	Descriptions
31:9	Reserved	Must be kept at reset value
8	SRAM_PCEF	SRAM parity check error flag This bit is set by hardware when an SRAM parity check error occurs. It is cleared by software by writing 1. 0: No SRAM parity check error detected 1: SRAM parity check error detected
7:3	Reserved	Must be kept at reset value
2	LVD_LOCK	LVD lock This bit is set by software and cleared by a system reset. 0: The LVD interrupt is disconnected from the break input of TIMER0/14/15/16. LVDEN and LVDT[2:0] in the PMU_CTL register can be programmed. 1: The LVD interrupt is connected from the break input of TIMER0/14/15/16. LVDEN and LVDT[2:0] in the PMU_CTL register are read only.
1	SRAM_PARITY_ERROR_LOCK	SRAM parity check error lock This bit is set by software and cleared by a system reset. 0: The SRAM parity check error is disconnected from the break input of TIMER0/14/15/16 1: The SRAM parity check error is connected from the break input of TIMER0/14/15/16
0	LOCKUP_LOCK	Cortex-M3 LOCKUP output lock This bit is set by software and cleared by a system reset. 0: The Cortex-M3 LOCKUP output is disconnected from the break input of TIMER0/14/15/16 1: The Cortex-M3 LOCKUP output is connected from the break input of TIMER0/14/15/16

## 1.6. Device electronic signature

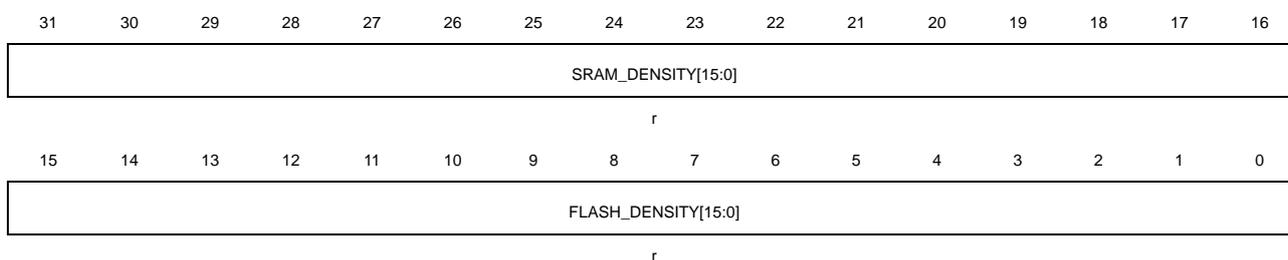
The device electronic signature contains memory density information and the 96-bit unique device ID. It is stored in the information block of the Flash memory. The 96-bit unique device ID is unique for any device. It can be used as serial numbers, or part of security keys, etc.

### 1.6.1. Memory density information

Base address: 0x1FFF F7E0

The value is factory programmed and can never be altered by user.

This register has to be accessed by word(32-bit)



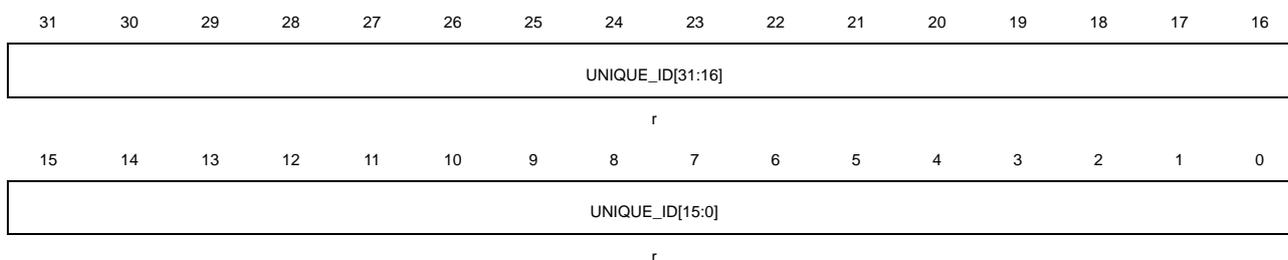
Bits	Fields	Descriptions
31:16	SRAM_DENSITY [15:0]	SRAM density The value indicates the on-chip SRAM density of the device in Kbytes. Example: 0x0008 indicates 8 Kbytes.
15:0	FLASH_DENSITY [15:0]	Flash memory density The value indicates the Flash memory density of the device in Kbytes. Example: 0x0020 indicates 32 Kbytes.

## 1.6.2. Unique device ID (96 bits)

Base address: 0x1FFF F7AC

The value is factory programmed and can never be altered by user.

This register has to be accessed by word(32-bit)

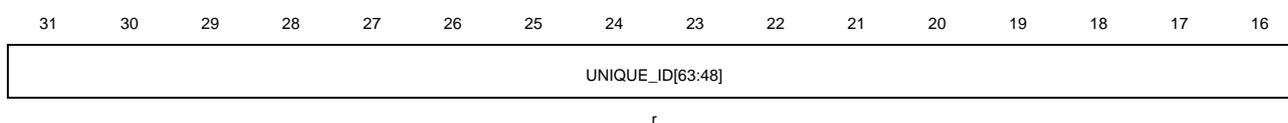


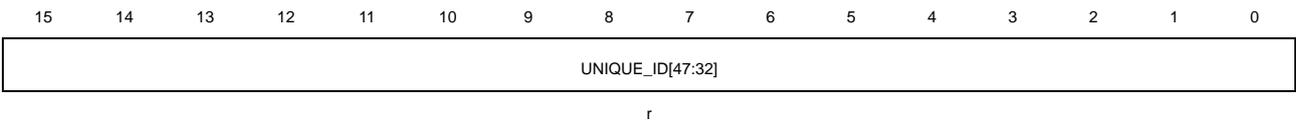
Bits	Fields	Descriptions
31:0	UNIQUE_ID[31:0]	Unique device ID

Base address: 0x1FFF F7B0

The value is factory programmed and can never be altered by user.

This register has to be accessed by word(32-bit)



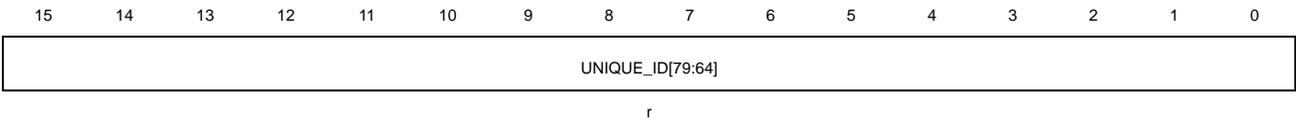
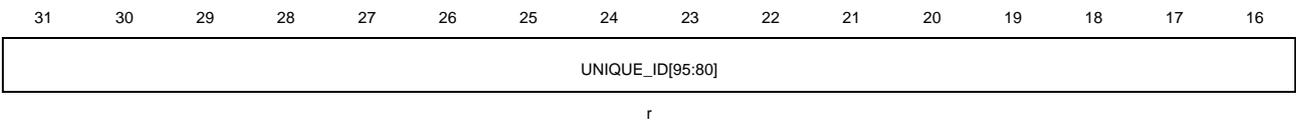


Bits	Fields	Descriptions
31:0	UNIQUE_ID[63:32]	Unique device ID

Base address: 0x1FFF F7B4

The value is factory programmed and can never be altered by user.

This register has to be accessed by word(32-bit)



Bits	Fields	Descriptions
31:0	UNIQUE_ID[95:64]	Unique device ID

## 2. Flash memory controller (FMC)

### 2.1. Overview

The Flash Memory Controller, FMC, provides all the necessary functions for the on-chip flash memory. There is no waiting time within 32K bytes while CPU executes instruction. It also provides page erase, mass erase, and word/half word program for flash memory.

### 2.2. Characteristics

- Up to 64KB of on-chip flash memory for storing instruction/data
- No waiting time within 32K bytes when CPU executes instruction
- A long delay when fetch 32K ~ 64K bytes data from flash
- 3K bytes information block for boot loader
- 16 bytes option bytes block for user application requirements
- 1K bytes page size
- Word or half word programming, page erase and mass erase capability
- Flash read protection to prevent illegal code/data access
- Page erase/program protection to prevent unexpected operation

### 2.3. Function overview

#### 2.3.1. Flash memory architecture

The flash memory consists of up to 64 KB main flash organized into 64 pages with 1 KB capacity per page and a 3 KB Information Block for the Boot Loader. The main flash memory contains a total of up to 64 pages which can be erased individually. The following table shows the base address and size.

**Table 2-1. Base address and size for flash memory**

Block	Name	Address	size(bytes)
Main Flash Block	Page 0	0x0800 0000 - 0x0800 03FF	1KB
	Page 1	0x0800 0400 - 0x0800 07FF	1KB
	Page 2	0x0800 0800 - 0x0800 0BFF	1KB
	.	.	.
	.	.	.
	Page 63	0x0800 FC00 - 0x0800 FFFF	1KB
Information Block	Boot loader	0x1FFF EC00 - 0x1FFF F7FF	3KB
Option Bytes Block	Option bytes	0x1FFF F800 - 0x1FFF F80F	16B

**Note:** The Information Block stores the bootloader - this block cannot be programmed or erased by user.

### 2.3.2. Read operations

The flash can be addressed directly as a common memory space. Any instruction fetch and the data access from the flash are through the IBUS or DBUS from the CPU.

### 2.3.3. Unlock the FMC\_CTL register

After reset, the FMC\_CTL register is not accessible in write mode, except for the OBRDL bit, which is used for reloading the option bytes, and the LK bit in FMC\_CTL register is 1. An unlocking sequence consists of two write operations to the FMC\_KEY register can open the access to the FMC\_CTL register. The two write operations are writing 0x45670123 and 0xCDEF89AB to the FMC\_KEY register. After the two write operations, the LK bit in FMC\_CTL register is set to 0 by hardware. The software can lock the FMC\_CTL again by setting the LK bit in FMC\_CTL register to 1. If there is any wrong operations on the FMC\_KEY register, the LK bit in FMC\_CTL register will be set, and the FMC\_CTL register will be locked, then it will generate a bus error.

The OBPG bit and OBER bit in FMC\_CTL are also protected by FMC\_OBKEY register. The unlocking sequence includes two write operations, which are writing 0x45670123 and 0xCDEF89AB to FMC\_OBKEY register. And then set the OBWEN bit in FMC\_CTL register to 1. The software can set OBWEN bit to 0 to protect the OBPG bit and OBER bit in FMC\_CTL register again.

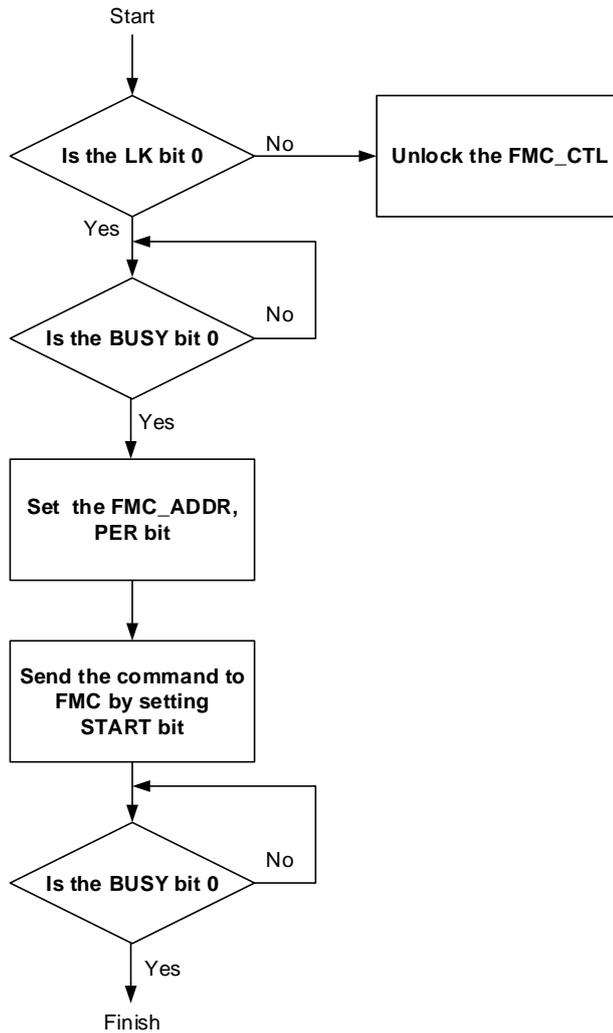
### 2.3.4. Page erase

The FMC provides a page erase function which is used for initializing the contents of a main flash memory page to a high state. Each page can be erased independently without affecting the contents of other pages. The following steps show the access sequence of the register for a page erase operation.

- Unlock the FMC\_CTL register if necessary.
- Check the BUSY bit in FMC\_STAT register to confirm that no flash memory operation is in progress (BUSY equal to 0). Otherwise, wait until the operation has been finished.
- Write the page address into the FMC\_ADDR register.
- Write the page erase command into PER bit in FMC\_CTL register.
- Send the page erase command to the FMC by setting the START bit in FMC\_CTL register.
- Wait until all the operations have been completed by checking the value of the BUSY bit in FMC\_STAT register.
- Read and verify the page if required using a DBUS access.

When the operation is executed successfully, an interrupt will be triggered by FMC if the ENDIE bit in the FMC\_CTL register is set, and the ENDF in FMC\_STAT register is set. Note that a correct target page address must be confirmed. Or the software may run out of control if the target erase page is being used for fetching codes or accessing data. The FMC will not provide any notification when this occurs. Additionally, the page erase operation will be ignored on protected pages. A Flash Operation Error interrupt will be triggered by the FMC if the ERRIE bit in the FMC\_CTL register is set. The software can check the PGERR bit in the FMC\_STAT register to detect this condition in the interrupt handler. The end of this operation is indicated by the ENDF bit in the FMC\_STAT register. The following figure shows the page erase operation flow.

Figure 2-1. Process of page erase operation



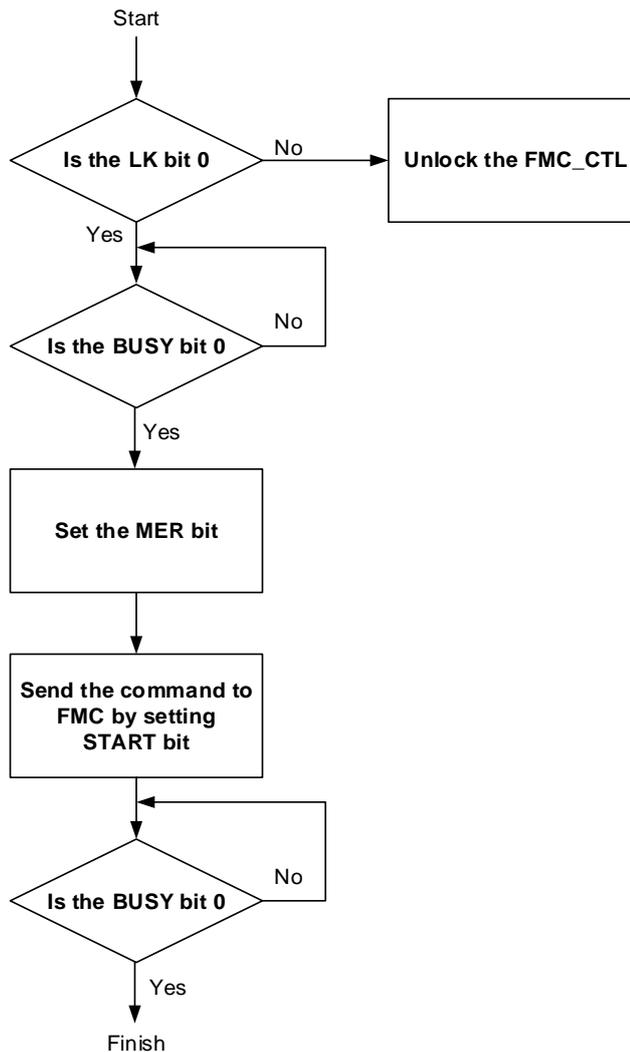
### 2.3.5. Mass erase

The FMC provides a complete erase function which is used for initializing the Main Flash Block contents. The following steps show the mass erase register access sequence.

- Unlock the FMC\_CTL register if necessary.
- Check the BUSY bit in FMC\_STAT register to confirm that no flash memory operation is in progress (BUSY equal to 0). Otherwise, wait until the operation has been finished.
- Write the mass erase command into MER bit in FMC\_CTL register.
- Send the mass erase command to the FMC by setting the START bit in FMC\_CTL register.
- Wait until all the operations have been completed by checking the value of the BUSY bit in FMC\_STAT register.
- Read and verify the flash memory if required using a DBUS access.

When the operation is executed successfully, an interrupt will be triggered by FMC if the ENDIE bit in the FMC\_CTL register is set, and the ENDF in FMC\_STAT register is set. Since all flash data will be reset to a value of 0xFFFF FFFF, the mass erase operation can be implemented using a program that runs in SRAM or by using the debugging tool to access the FMC registers directly. The end of this operation is indicated by the ENDF bit in the FMC\_STAT register. (The starting address of programming operation should be 0x0800 0000) The following figure indicates the mass erase operation flow.

**Figure 2-2. Process of the mass erase operation**



### 2.3.6. Main flash programming

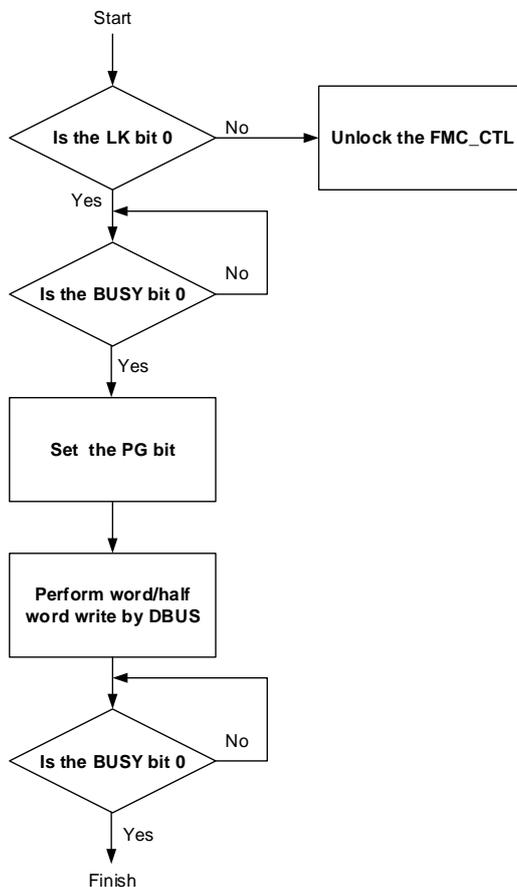
The FMC provides a 32-bit word/16-bit half word programming function which is used to modify the main flash memory contents. The following steps show the word programming operation register access sequence.

- Unlock the FMC\_CTL register if necessary.

- Check the BUSY bit in FMC\_STAT register to confirm that no flash memory operation is in progress (BUSY equal to 0). Otherwise, wait until the operation has been finished.
- Write the word program command into the PG bit in FMC\_CTL register.
- A 32-bit word/16-bit half word write at desired address by DBUS.
- Wait until all the operations have been completed by checking the value of the BUSY bit in FMC\_STAT register.
- Read and verify the flash memory if required using a DBUS access.

When the operation is executed successfully, an interrupt will be triggered by FMC if the ENDIE bit in the FMC\_CTL register is set, and the ENDF in FMC\_STAT register is set. Note that before the word/half word programming operation you should check the address that it has been erased. If the address has not been erased, PGERR bit will set when programming the address except programming 0x0. Additionally, the program operation will be ignored on protected pages. A flash operation error interrupt will be triggered by the FMC if the ERRIE bit in the FMC\_CTL register is set. The software can check the PGERR bit in the FMC\_STAT register to detect this condition in the interrupt handler. The end of this operation is indicated by the ENDF bit in the FMC\_STAT register. The following figure displays the word programming operation flow.

**Figure 2-3. Process of the word programming operation**



### 2.3.7. Option bytes erase

The FMC provides an erase function which is used for initializing the option bytes block in flash. The following steps show the erase sequence.

- Unlock the FMC\_CTL register if necessary.
- Unlock the OBWEN bit in FMC\_CTL register if necessary.
- Check the BUSY bit in FMC\_STAT register to confirm that no flash memory operation is in progress (BUSY equal to 0). Otherwise, wait until the operation has been finished.
- Write the option bytes erase command into OBER bit in FMC\_CTL register.
- Send the option bytes erase command to the FMC by setting the START bit in FMC\_CTL register.
- Wait until all the operations have been completed by checking the value of the BUSY bit in FMC\_STAT register.
- Read and verify the flash memory if required using a DBUS access.

When the operation is executed successfully, an interrupt will be triggered by FMC if the ENDIE bit in the FMC\_CTL register is set, and the ENDF in FMC\_STAT register is set. The end of this operation is indicated by the ENDF bit in the FMC\_STAT register.

### 2.3.8. Option bytes programming

The FMC provides a 32-bit word/16-bit half word programming function which is used for modifying the option bytes block contents. The following steps show the programming operation sequence.

- Unlock the FMC\_CTL register if necessary.
- Unlock the OBWEN bit in FMC\_CTL register if necessary.
- Check the BUSY bit in FMC\_STAT register to confirm that no flash memory operation is in progress (BUSY equal to 0). Otherwise, wait until the operation has been finished.
- Write the program command into the OBPG bit in FMC\_CTL register.
- A 32-bit word/16-bit half word write at desired address by DBUS.
- Wait until all the operations have been completed by checking the value of the BUSY bit in FMC\_STAT register.
- Read and verify the flash memory if required using a DBUS access.

When the operation is executed successfully, an interrupt will be triggered by FMC if the ENDIE bit in the FMC\_CTL register is set, and the ENDF in FMC\_STAT register is set. Note that before the word/half word programming operation you should check the address that it has been erased. If the address has not been erased, PGERR bit will set when programming

the address except programming 0x0. The end of this operation is indicated by the ENDF bit in the FMC\_STAT register.

## 2.3.9. Option bytes description

The option bytes block of flash memory reloaded to FMC\_OBSTAT and FMC\_WP registers after each system reset or OBRLD bit set in FMC\_CTL register, and the option bytes work. The option complement bytes are the opposite of option bytes. When option bytes reload, if the option complement bytes and option bytes does not match, the OBERR bit in FMC\_OBSTAT register is set, and the option bytes is set to 0xFF. The following table is the detail of option bytes.

**Table 2-2. Option bytes**

Address	Name	Description
0x1fff f800	OB_SPC	option bytes Security Protection Code 0xA5: No protection any value except 0xA5 or 0xCC: Protection level low 0xCC: Protection level high
0x1fff f801	OB_SPC_N	OB_SPC complement value
0x1fff f802	OB_USER	option bytes which user defined [7]: Reserved [6]: SRAM_PARITY_CHECK 0: Enable sram parity check 1: Disable sram parity check [5]: VDDA_VISOR 0: Disable V <sub>DDA</sub> monitor 1: Enable V <sub>DDA</sub> monitor [4]: BOOT1_n 0: BOOT1 bit is 1 1: BOOT1 bit is 0 [3]: Reserved [2]: nRST_STDBY 0: Generate a reset instead of entering standby mode 1: No reset when entering standby mode [1]: nRST_DPSLP 0: Generate a reset instead of entering Deep-sleep mode 1: No reset when entering Deep-sleep mode [0]: nWDG_SW 0: Hardware free watchdog timer 1: Software free watchdog timer
0x1fff f803	OB_USER_N	OB_USER complement value
0x1fff f804	OB_DATA[7:0]	user defined data bit 7 to 0
0x1fff f805	OB_DATA_N[7:0]	OB_DATA complement value bit 7 to 0

Address	Name	Description
0x1fff f806	OB_DATA[15:8]	user defined data bit 15 to 8
0x1fff f807	OB_DATA_N[15:8]	OB_DATA complement value bit 15 to 8
0x1fff f808	OB_WP[7:0]	Page Erase/Program Protection bit 7 to 0
0x1fff f809	OB_WP_N[7:0]	OB_WP complement value bit 7 to 0
0x1fff f80a	OB_WP[15:8]	Page Erase/Program Protection bit 15 to 8
0x1fff f80b	OB_WP_N[15:8]	OB_WP complement value bit 15 to 8

### 2.3.10. Page erase/Program protection

The FMC provides page erase/program protection functions to prevent inadvertent operations on the flash memory. The page erase or program will not be accepted by the FMC on protected pages. If the page erase or program command is sent to the FMC on a protected page, the WPERR bit in the FMC\_STAT register will then be set by the FMC. If the WPERR bit is set and the ERRIE bit is also set to 1 to enable the corresponding interrupt, then the flash operation error interrupt will be triggered by the FMC to get the attention of the CPU. The page protection function can be individually enabled by configuring the OB\_WP [15:0] bit field to 0 in the option bytes. If a page erase operation is executed on the Option Bytes region, all the flash memory page protection functions will be disabled. When setting or resetting OB\_WP in the option bytes, the software need to set OBRLD in FMC\_CTL register or a system reset to reload the OB\_WP bits. The following table shows which pages are protected by set OB\_WP [15:0].

**Table 2-3. OB\_WP bit for pages protected**

OB_WP bit	pages protected
OB_WP[0]	page 0 ~ page 3
OB_WP[1]	page 4 ~ page 7
OB_WP[2]	page 8 ~ page 11
.	.
.	.
.	.
OB_WP[14]	page 56 ~ page 59
OB_WP[15]	page 60 ~ page 63

### 2.3.11. Security protection

The FMC provides a security protection function to prevent illegal code/data access on the flash memory. This function is useful for protecting the software/firmware from illegal users. There are 3 levels for protecting:

No protection: when setting OB\_SPC byte and its complement value to 0xA55A, no protection performed. The main flash and option bytes block are accessible by all operations.

Protection level low: when setting OB\_SPC byte and its complement value to any value

except 0xA55A or 0xCC33, protection level low performed. The main flash can only be accessed by user code. In debug mode, boot from SRAM or boot from boot loader mode, all operations to main flash is forbidden. If a read operation is executed to main flash in debug mode, boot from SRAM or boot from boot loader mode, a bus error will be generated. If a program/erase operation is executed to main flash in debug mode, boot from SRAM or boot from boot loader mode, the PGERR bit in FMC\_STAT register will be set. At protection level low, option bytes block are accessible by all operations. If program back to no protection level by setting OB\_SPC byte and its complement value to 0xA55A, a mass erase for main flash will be performed.

Protection level high: when set OB\_SPC byte and its complement value to 0xCC33, protection level high performed. When this level is programmed in debug mode, boot from SRAM or boot from boot loader mode is disabled. The main flash block is accessible by all operations from user code. The option bytes cannot be erased, and the OB\_SPC byte and its complement value cannot be reprogrammed. So, if protection level high is programmed, it cannot move back to protection level low or no protection level.

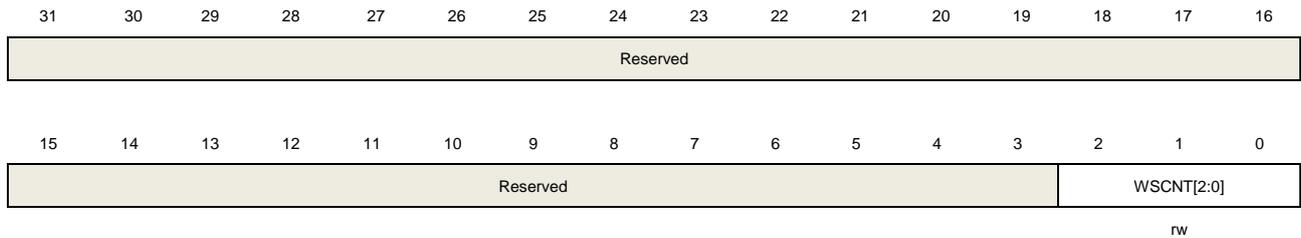
## 2.4. Register definition

### 2.4.1. Wait state register (FMC\_WS)

Address offset: 0x00

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)



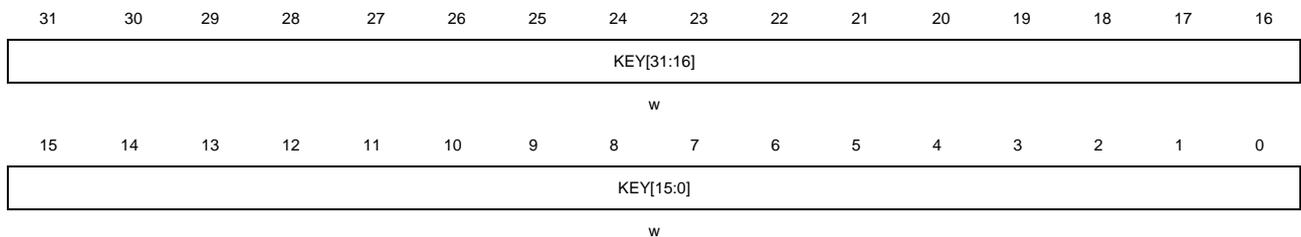
Bits	Fields	Descriptions
31:3	Reserved	Must be kept at reset value
2:0	WSCNT[2:0]	Wait state counter register These bits set and reset by software. The WSCNT valid when WSEN bit is set 000: 0 wait state added 001: 1 wait state added 010: 2 wait state added 011 ~ 111: Reserved

### 2.4.2. Unlock key register (FMC\_KEY)

Address offset: 0x04

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)



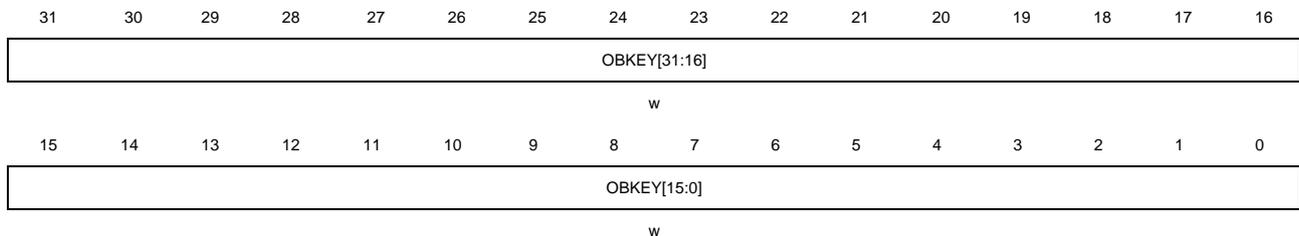
Bits	Fields	Descriptions
31:0	KEY[31:0]	FMC_CTL unlock registers These bits are only be written by software Write KEY[31:0] with key to unlock FMC_CTL register.

### 2.4.3. Option bytes unlock key register (FMC\_OBKEY)

Address offset: 0x08

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)



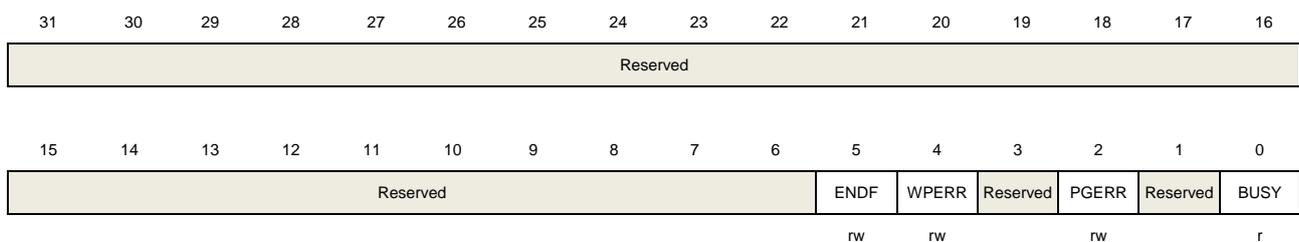
Bits	Fields	Descriptions
31:0	OBKEY[31:0]	FMC_CTL option bytes operation unlock registers These bits are only be written by software Write OBKEY[31:0] with key to unlock option bytes command in FMC_CTL register.

### 2.4.4. Status register (FMC\_STAT)

Address offset: 0x0C

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)



Bits	Fields	Descriptions
31:6	Reserved	Must be kept at reset value
5	ENDF	End of operation flag bit When the operation executed successfully, this bit is set by hardware. The software can clear it by writing 1.
4	WPERR	Erase/Program protection error flag bit When erasing/programming on protected pages, this bit is set by hardware. The software can clear it by writing 1.
3	Reserved	Must be kept at reset value

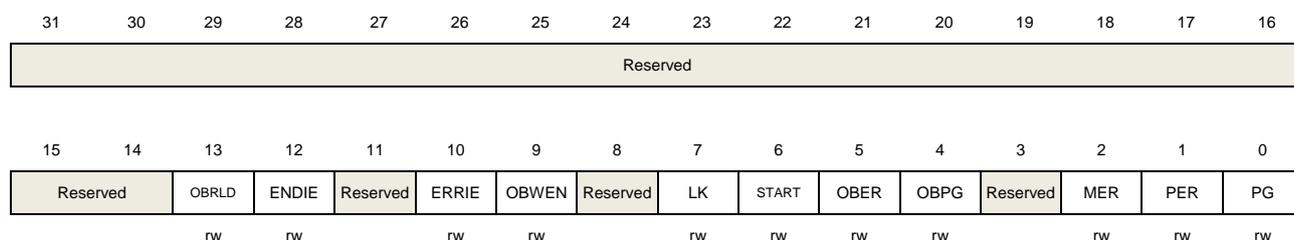
2	PGERR	Program error flag bit When programming to the flash while it is not 0xFFFF, this bit is set by hardware. The software can clear it by writing 1.
1	Reserved	Must be kept at reset value
0	BUSY	The flash busy bit When the operation is in progress, this bit is set to 1. When the operation is end or an error generated, this bit is clear to 0.

### 2.4.5. Control register (FMC\_CTL)

Address offset: 0x10

Reset value: 0x0000 0080

This register has to be accessed by word(32-bit)



Bits	Fields	Descriptions
31:14	Reserved	Must be kept at reset value
13	OBRLD	Option bytes reload bit This bit is set by software. 0: No effect 1: Force option bytes reload, and generate a system reset
12	ENDIE	End of operation interrupt enable bit This bit is set or cleared by software. 0: No interrupt generated by hardware 1: End of operation interrupt enable
11	Reserved	Must be kept at reset value
10	ERRIE	Error interrupt enable bit This bit is set or cleared by software. 0: No interrupt generated by hardware 1: Error interrupt enable
9	OBWEN	Option bytes erase/program enable bit This bit is set by hardware when right sequence written to FMC_OBKEY register. This bit can be cleared by software.

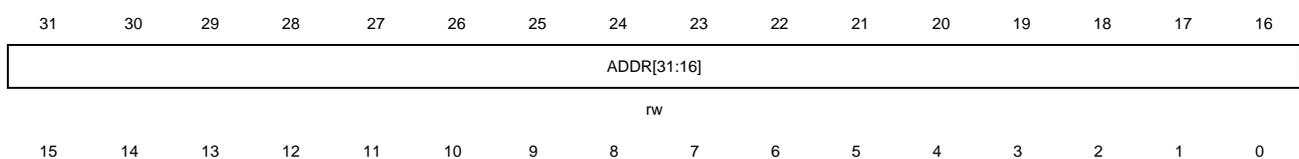
8	Reserved	Must be kept at reset value
7	LK	FMC_CTL lock bit This bit is cleared by hardware when right sequent written to FMC_KEY register. This bit can be set by software.
6	START	Send erase command to FMC bit This bit is set by software to send erase command to FMC. This bit is cleared by hardware when the BUSY bit is cleared.
5	OBER	Option bytes erase command bit This bit is set or cleared by software. 0: No effect 1: Option bytes erase command
4	OBPG	Option bytes program command bit This bit is set or cleared by software. 0: No effect 1: Option bytes program command
3	Reserved	Must be kept at reset value
2	MER	Main flash mass erase command bit This bit is set or cleared by software. 0: No effect 1: Main flash mass erase command
1	PER	Main flash page erase command bit This bit is set or cleared by software. 0: No effect 1: Main flash page erase command
0	PG	Main flash page program command bit This bit is set or cleared by software. 0: No effect 1: Main flash page program command

## 2.4.6. Address register (FMC\_ADDR)

Address offset: 0x14

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)





rw

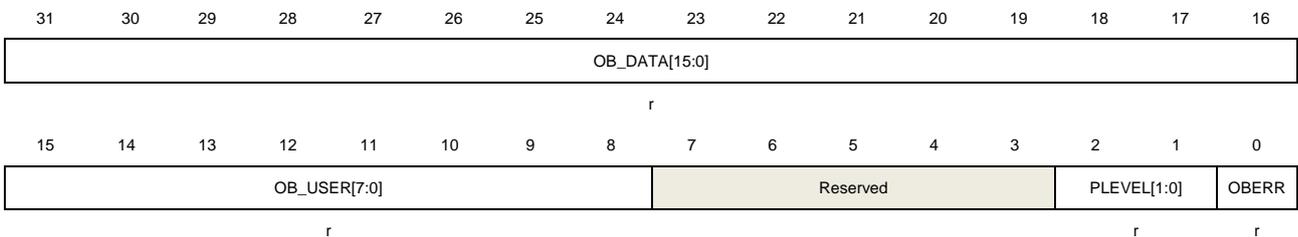
Bits	Fields	Descriptions
31:0	ADDR[31:0]	Flash command address bits These bits are set by software. ADDR bits are the address of flash erase command

### 2.4.7. Option bytes status register (FMC\_OBSTAT)

Address offset: 0x1C

Reset value: 0XXXXX XX0X

This register has to be accessed by word(32-bit)



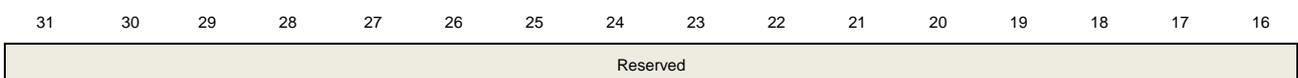
Bits	Fields	Descriptions
31:16	OB_DATA[15:0]	Store OB_DATA[15:0] of option bytes block after system reset
15:8	OB_USER[7:0]	Store OB_USER byte of option bytes block after system reset
7:3	Reserved	Must be kept at reset value
2:1	PLEVEL[1:0]	Security Protection level 00: No protection level 01: Protect level low 11: Protect level high
0	OBERR	Option bytes read error bit. This bit is set by hardware when the option bytes and its complement bytes do not match, and the option bytes set 0xFF.

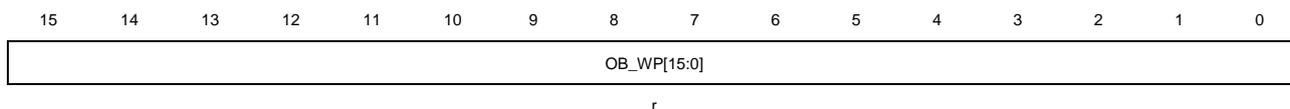
### 2.4.8. Write protection register (FMC\_WP)

Address offset: 0x20

Reset value: 0x0000 XXXX

This register has to be accessed by word(32-bit)





Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	OB_WP[15:0]	Store OB_WP[15:0] of option bytes block after system reset 0: Protection active 1: Unprotected

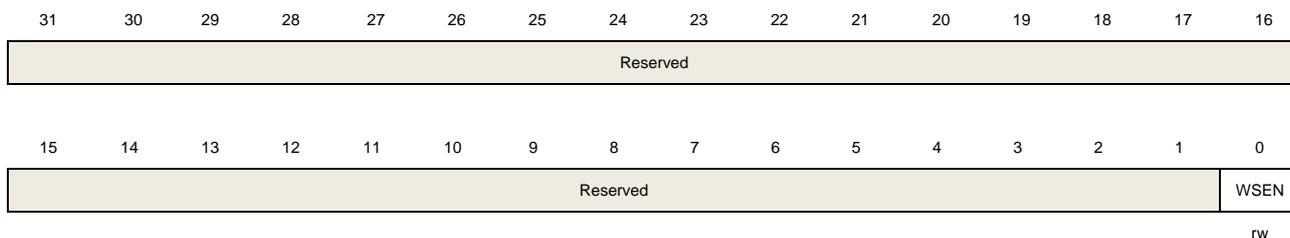
### 2.4.9. Wait state enable register (FMC\_WSEN)

#### For GD32F130xx and GD32F150xx devices

Address offset: 0xFC

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)



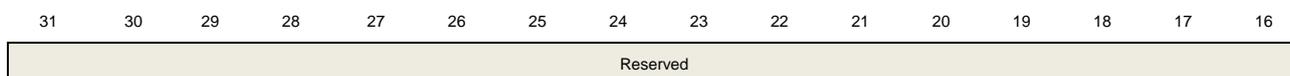
Bits	Fields	Descriptions
31:1	Reserved	Must be kept at reset value
0	WSEN	FMC wait state enable register This bit set and reset by software. This bit is also protected by the FMC_KEY register. The software need writing 0x45670123 and 0xCDEF89AB to the FMC_KEY register. 0: No wait state added when fetching flash 1: Wait state added when fetching flash

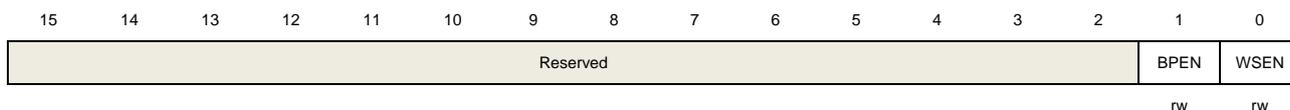
#### For GD32F170xx and GD32F190xx devices

Address offset: 0xFC

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)





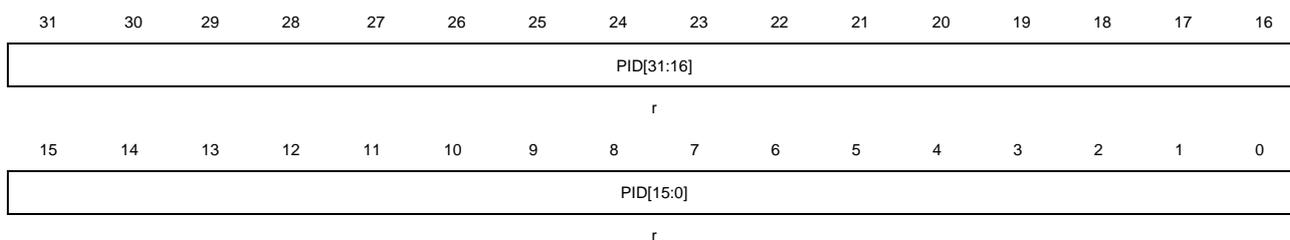
Bits	Fields	Descriptions
31:2	Reserved	Must be kept at reset value
1	BPEN	<p>FMC bit program enable register</p> <p>This bit set and reset by software.</p> <p>0: No effect, write page must check if the flash is “FF”</p> <p>1: Write page do not check if the flash is FF. The FMC can program each bit.</p>
0	WSEN	<p>FMC wait state enable register</p> <p>This bit set and reset by software. This bit is also protected by the FMC_KEY register. The software need writing 0x45670123 and 0xCDEF89AB to the FMC_KEY register.</p> <p>0: No wait state added when fetching flash</p> <p>1: Wait state added when fetching flash</p>

## 2.4.10. Product ID register (FMC\_PID)

Address offset: 0x100

Reset value: 0XXXXX XXXX

This register has to be accessed by word(32-bit)



Bits	Fields	Descriptions
31:0	PID[31:0]	<p>Product reserved ID code register</p> <p>These bits are read only by software.</p> <p>These bits are unchanged constantly after power on. These bits are one time programmed when the chip product.</p>

## 3. Power management unit (PMU)

### 3.1. Overview

The power consumption is regarded as one of the most important issues for the devices of GD32F1x0 series. The Power management unit (PMU), provides three types of power saving modes, including Sleep, Deep-sleep and Standby mode. These modes reduce the power consumption and allow the application to achieve the best tradeoff among the conflicting demands of CPU operating time, speed and power consumption. For GD32F130xx and GD32F150xx devices, there are three power domains, including  $V_{DD}/V_{DDA}$  domain, 1.2V domain, and Backup domain, as is shown in the following figure. For GD32F170xx and GD32F190xx devices, there are three power domains, including  $V_{DD}/V_{DDA}$  domain, 1.8V domain, and Backup domain, as is shown in the following figure. The power of the  $V_{DD}$  domain is supplied directly by  $V_{DD}$ . An embedded LDO in the  $V_{DD}/V_{DDA}$  domain is used to supply the 1.2V/1.8V domain power. A power switch is implemented for the Backup domain. It can be powered from the  $V_{BAT}$  voltage when the main  $V_{DD}$  supply is shut down.

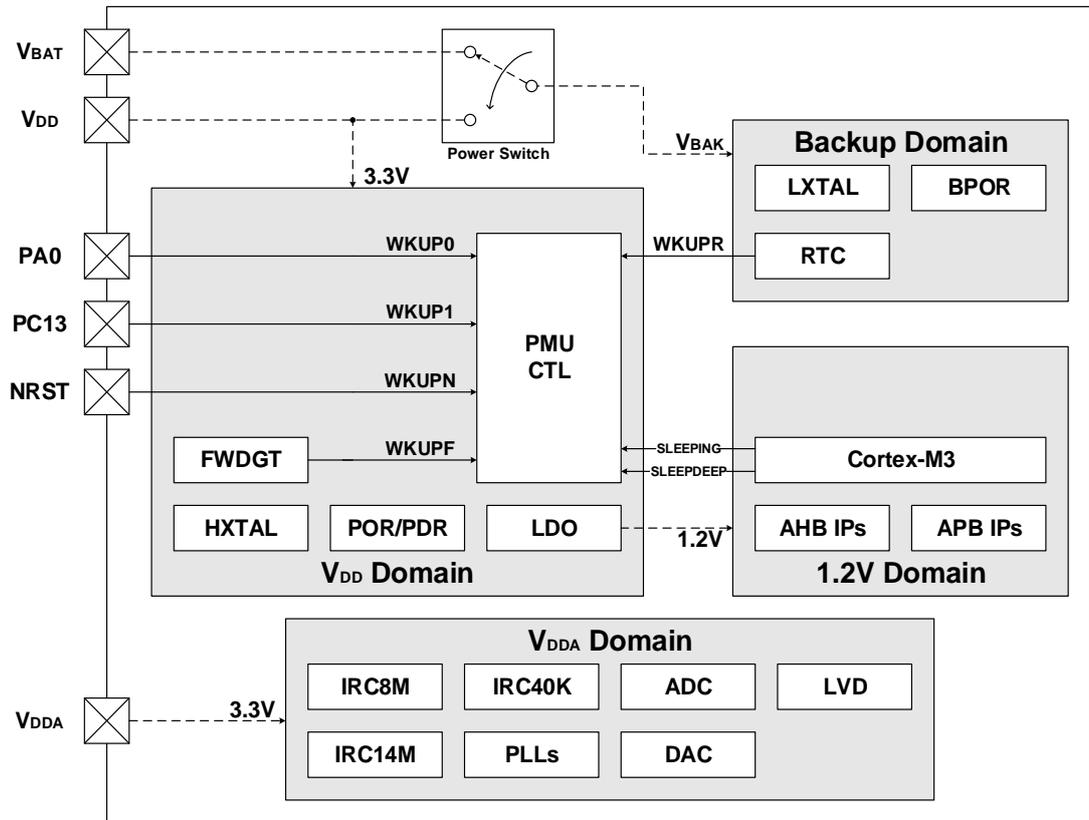
### 3.2. Characteristics

- Three power domains:  $V_{BAK}$ ,  $V_{DD}/V_{DDA}$  and 1.2V power domains for GD32F130xx and GD32F150xx devices or 1.8V power domains for GD32F170xx and GD32F190xx devices
- Three power saving modes: Sleep, Deep-sleep and Standby modes
- Internal Voltage regulator(LDO) supplies 1.2V voltage source for GD32F130xx and GD32F150xx devices or 1.8 V voltage source for GD32F170xx and GD32F190xx devices
- Low Voltage Detector can issue an interrupt or event when the power is lower than a programmed threshold
- Battery power ( $V_{BAT}$ ) for Backup domain when  $V_{DD}$  is shut down

### 3.3. Function overview

[Figure 3-1. Power supply overview of GD32F130xx and GD32F150xx devices](#) and [Figure 3-2. Power supply overview of GD32F170xx and GD32F190xx devices](#) provide details on the internal configuration of the PMU and the relevant power domains.

Figure 3-1. Power supply overview of GD32F130xx and GD32F150xx devices

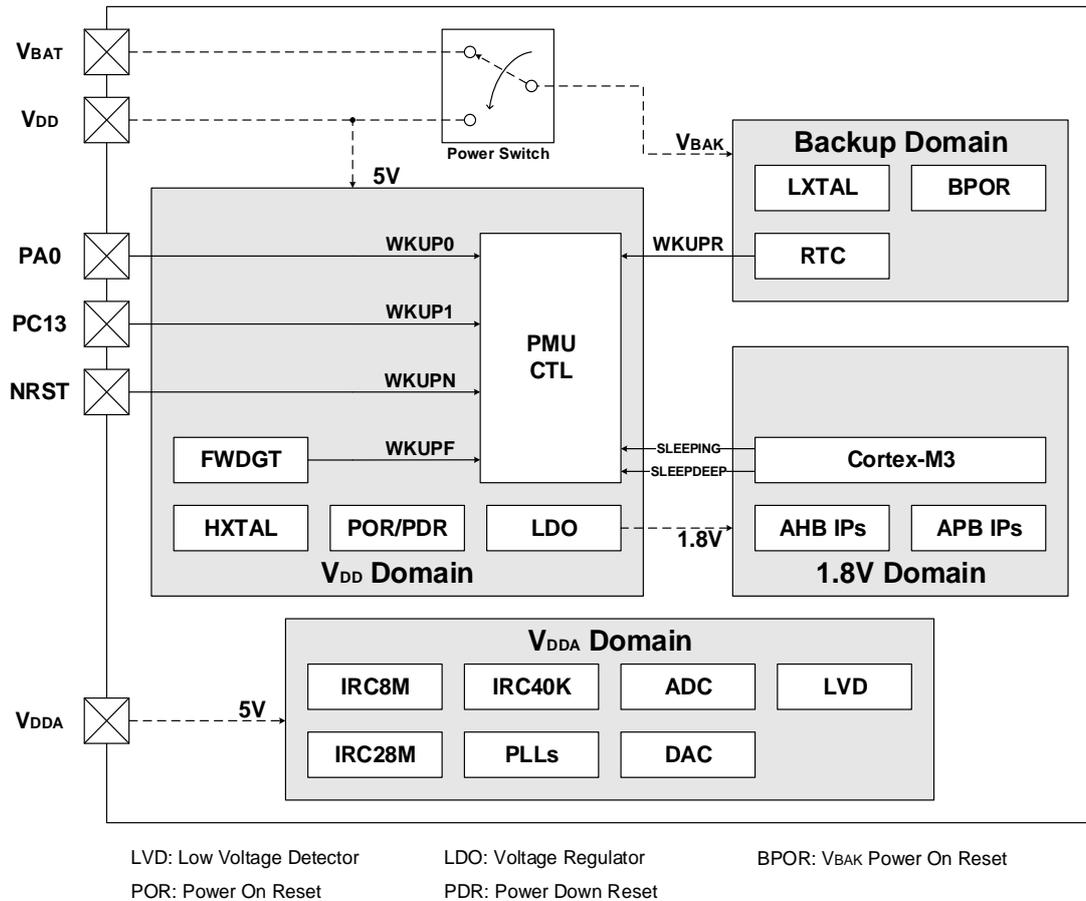


LVD: Low Voltage Detector  
 POR: Power On Reset

LDO: Voltage Regulator  
 PDR: Power Down Reset

BPOR: V<sub>BAK</sub> Power On Reset

Figure 3-2. Power supply overview of GD32F170xx and GD32F190xx devices



### 3.3.1. Battery backup domain

The Backup domain is powered by the  $V_{DD}$  or the battery power source ( $V_{BAT}$ ) selected by the internal power switch, and the  $V_{BAK}$  pin which drives Backup Domain, power supply for RTC unit, LXTAL oscillator, BPOR, and three pads, including PC13 to PC15. In order to keeping the content of the Backup domain registers and the RTC supply, when  $V_{DD}$  supply is shut down,  $V_{BAT}$  pin can be connected to an optional standby voltage supplied by a battery or by another source. The power switch is controlled by the Power Down Reset circuit in the  $V_{DD}/V_{DDA}$  domain. If no external battery is used in the application, it is recommended to connect  $V_{BAT}$  pin externally to  $V_{DD}$  pin with a 100nF external ceramic decoupling capacitor.

The Backup Domain reset sources include the Backup domain power-on-reset (BPOR) and the Backup Domain software reset. The BPOR signal forces the device to stay in the reset mode until  $V_{BAK}$  is completely powered up. Also the application software can trigger the Backup domain software reset by setting the BKPRST bit in the RCU\_BDCTL register to reset the Backup domain.

The clock source of Real Time Clock (RTC) circuit can be derived from the Internal 40 KHz RC oscillator (IRC40K) or the Low Speed Crystal oscillator (LXTAL), or HXTAL clock divided by 32. When  $V_{DD}$  is shut down, only LXTAL is valid for RTC. Before entering the power saving

mode by executing the WFI/WFE instruction, the Cortex™-M3 can setup the RTC register with an expected alarm time and enable the alarm function and according EXTI lines to achieve the RTC timer wakeup event. After entering the power saving mode for a certain amount of time, the RTC alarm will wake up the device when the time match event occurs. The details of the RTC configuration and operation will be described in the RTC chapter.

When the Backup domain is supplied by  $V_{DD}$  ( $V_{BAK}$  pin is connected to  $V_{DD}$ ), the following functions are available:

- PC13 can be used as GPIO or RTC function pin described in the RTC chapter.
- PC14 and PC15 can be used as either GPIO or LXTAL Crystal oscillator pins.

When the Backup domain is supplied by  $V_{BAT}$  ( $V_{BAK}$  pin is connected to  $V_{BAT}$ ), the following functions are available:

- PC13 can be used as RTC function pin described in the RTC chapter.
- PC14 and PC15 can be used as LXTAL Crystal oscillator pins only.

**Note:** Since PC13, PC14 and PC15 are supplied through the Power Switch, which can only be obtained by a small current, the speed of GPIOs PC13 to PC15 should not exceed 2MHz when they are in output mode(maximum load: 30pF).

### 3.3.2. $V_{DD}/V_{DDA}$ power domain

$V_{DD}/V_{DDA}$  domain includes two parts:  $V_{DD}$  domain and  $V_{DDA}$  domain.  $V_{DD}$  domain includes HXTAL (High Speed Crystal oscillator), LDO (Voltage Regulator), POR/PDR (Power On/Down Reset), FWDGT (Free Watchdog Timer), all pads except PC13/PC14/PC15, etc.  $V_{DDA}$  domain includes ADC/DAC (AD/DA Converter), IRC8M (Internal 8MHz RC oscillator), IRC14M (Internal 14MHz RC oscillator at 14MHz frequency) for GD32F130xx and GD32F150xx devices, IRC28M (Internal 28MHz RC oscillator at 28MHz frequency) for GD32F170xx and GD32F190xx devices, IRC40K (Internal 40KHz RC oscillator), PLLs (Phase Locking Loop), LVD (Low Voltage Detector), etc.

#### $V_{DD}$ domain

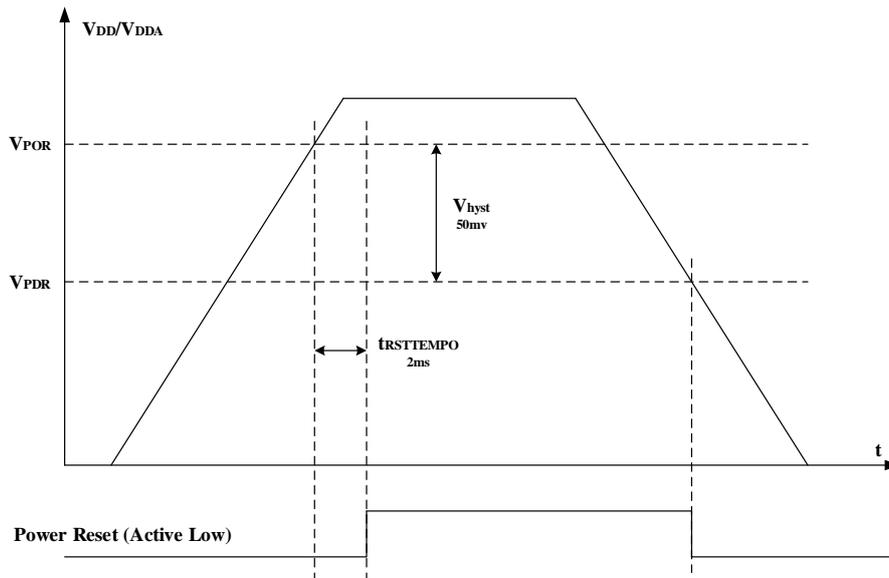
The LDO, which is implemented to supply power for the 1.2V domain in GD32F130xx and GD32F150xx devices or 1.8V domain in GD32F170xx and GD32F190xx devices, is always enabled after reset. It can be configured to operate in three different status, including in the Sleep mode (full power on), in the Deep-sleep mode (on or low power), and in the Standby mode (power off).

The POR/PDR circuit is implemented to detect  $V_{DD}/V_{DDA}$  and generate the power reset signal which resets the whole chip except the Backup domain when the supply voltage is lower than the specified threshold.

**Figure 3-3. Waveform of the POR/PDR** shows the relationship between the supply voltage and the power reset signal.  $V_{POR}$  indicates the threshold of power on reset, while  $V_{PDR}$  means

the threshold of power down reset. For GD32F130xx and GD32F150xx devices,  $V_{PDR}$  is configurable. There are two  $V_{PDR}$  values which can be selected by the PDVSEL bit in the RCU\_PDVSSEL registers (Refer to Chapter4 RCU registers). When the lower one is selected, it is strongly recommended that neither programming nor erasing is performed to the flash memory since it may fail when the voltage is low enough nearly the threshold. The hysteresis voltage ( $V_{hyst}$ ) is around 50mV.

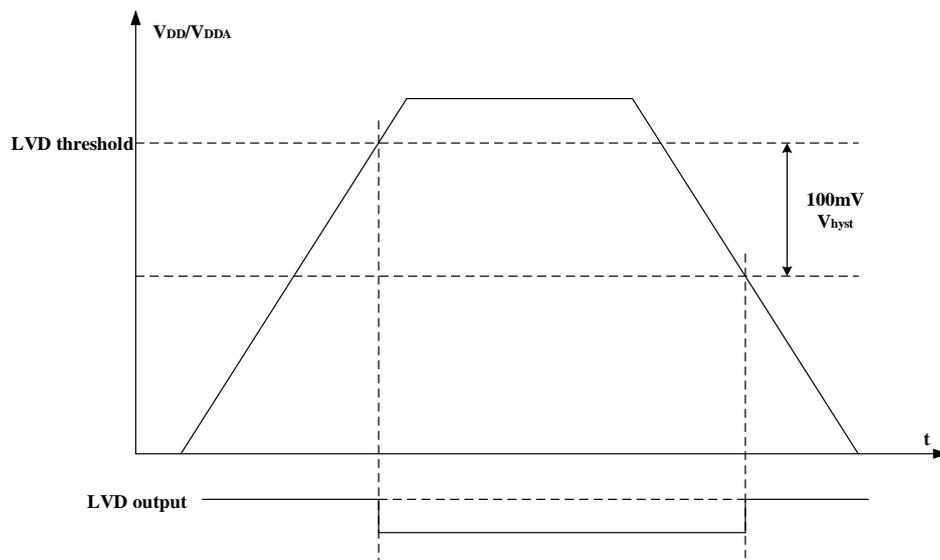
**Figure 3-3. Waveform of the POR/PDR**



## $V_{DDA}$ domain

The LVD is used to detect whether the  $V_{DD}/V_{DDA}$  supply voltage is lower than a programmed threshold selected by the LVDT[2:0] bits in the Power control register(PMU\_CTL). The LVD is enabled by setting the LVDEN bit, and LVDF bit, which in the Power status register (PMU\_CS), indicates if  $V_{DD}/V_{DDA}$  is higher or lower than the LVD threshold. This event is internally connected to the EXTI line 16 and can generate an interrupt if it is enabled through the EXTI registers. [Figure 3-4. Waveform of LVD threshold](#) shows the relationship between the LVD threshold and the LVD output (LVD interrupt signal depends on EXTI line 16 rising or falling edge configuration). The following figure shows the relationship between the supply voltage and the LVD signal. The hysteresis voltage ( $V_{hyst}$ ) is 100mV.

Figure 3-4. Waveform of LVD threshold



Generally, digital circuits are powered by  $V_{DD}$ , while most of analog circuits are powered by  $V_{DDA}$ . To improve the ADC and DAC conversion accuracy, the independent power supply  $V_{DDA}$  is implemented to achieve better performance of analog circuits.  $V_{DDA}$  can be externally connected to  $V_{DD}$  through the external filtering circuit that avoids noise on  $V_{DDA}$ , and  $V_{SSA}$  should be connected to  $V_{SS}$  through the specific circuit independently. Otherwise, if  $V_{DDA}$  is different from  $V_{DD}$ ,  $V_{DDA}$  must always be higher, but the voltage difference should not exceed 0.2V.

### 3.3.3. 1.2V power domain for GD32F130xx and GD32F150xx devices

The main functions that include Cortex™-M3 logic, AHB/APB peripherals, the APB interfaces for the Backup domain and the  $V_{DD}/V_{DDA}$  domain, etc., are located in the 1.2V power domain. Once the 1.2V is powered up, the POR will generate a reset sequence on the 1.2V power domain. To enter the expected power saving mode, the associated control bits must be configured. Then, once a WFI (Wait for Interrupt) or WFE (Wait for Event) instruction is executed, the device will enter an expected power saving mode which will be discussed in the following section.

### 3.3.4. 1.8V power domain for GD32F170xx and GD32F190xx devices

The main functions that include Cortex™-M3 logic, AHB/APB peripherals, the APB interfaces for the Backup domain and the  $V_{DD}$  domain, etc., are located in the 1.8V power domain. Once the 1.8V is powered up, the POR will generate a reset sequence on the 1.8V power domain. To enter the expected power saving mode, the associated control bits must be configured. Then, once a WFI (Wait for Interrupt) or WFE (Wait for Event) instruction is executed, the device will enter an expected power saving mode which will be discussed in the following section.

### 3.3.5. Power saving modes

After a system reset or a power reset, the GD32F1x0 MCU operates at full function and all power domains are active. Users can achieve lower power consumption through slowing down the system clocks (HCLK, PCLK1, PCLK2) or gating the clocks of the unused peripherals. Besides, three power saving modes are provided to achieve even lower power consumption, they are Sleep mode, Deep-sleep mode, and Standby mode.

#### Sleep Mode

The Sleep mode is corresponding to the SLEEPING mode of the Cortex™-M3. In Sleep mode, only clock of Cortex™-M3 is off. To enter the Sleep mode, it is only necessary to clear the SLEEPDEEP bit in the Cortex™-M3 System Control Register, and execute a WFI or WFE instruction. If the Sleep mode is entered by executing a WFI instruction, any interrupt can wake up the system. If it is entered by executing a WFE instruction, any wakeup event can wake up the system. The mode offers the lowest wakeup time as no time is wasted in interrupt entry or exit.

According to the SLEEPONEXIT bit in the Cortex™-M3 System Control Register, there are two options to select the Sleep mode entry mechanism.

- Sleep-now: if the SLEEPONEXIT bit is cleared, the MCU enters Sleep mode as soon as WFI or WFE instruction is executed.
- Sleep-on-exit: if the SLEEPONEXIT bit is set, the MCU enters Sleep mode as soon as it exits from the lowest priority ISR.

#### Deep-sleep Mode

The Deep-sleep mode is based on the SLEEPDEEP mode of the Cortex™-M3. In Deep-sleep mode, all clocks in the 1.2V domain for GD32F130/150xx devices or 1.8V domain for GD32F170/190xx devices are off, and all of IRC8M, IRC14M for GD32F130/150xx devices or IRC28M for GD32F170/190xx devices, HXTAL and PLLs are disabled. The contents of SRAM and registers are preserved. The LDO can operate normally or in low power mode depending on the LDOLP bit in the PMU\_CTL register. Before entering the Deep-sleep mode, it is necessary to set the SLEEPDEEP bit in the Cortex™-M3 System Control Register, and clear the STBMOD bit in the PMU\_CTL register. Then, the device enters the Deep-sleep mode after a WFI or WFE instruction is executed. If the Deep-sleep mode is entered by executing a WFI instruction, any interrupt from EXTI lines can wake up the system. If it is entered by executing a WFE instruction, any wakeup event from EXTI lines can wake up the system. When exiting the Deep-sleep mode, the IRC8M is selected as the system clock. Notice that an additional wakeup delay will be incurred if the LDO operates in low power mode.

**Note:** In order to enter Deep-sleep mode smoothly, all EXTI line pending status (in the EXTI\_PD register) and RTC alarm/timestamp/tamper flag must be reset. If not, the program will skip the entry process of Deep-sleep mode to continue to execute the following procedure.

## Standby Mode

The Standby mode is based on the SLEEPDEEP mode of the Cortex™-M3, too. In Standby mode, the whole 1.2V domain for GD32F130/150xx devices or 1.8V domain for GD32F170/190xx devices is power off, the LDO is shut down, and all of IRC8M, IRC14M for GD32F130/150xx devices or IRC28M for GD32F170/190xx devices, HXTAL and PLLs are disabled. Before entering the Standby mode, it is necessary to set the SLEEPDEEP bit in the Cortex™-M3 System Control Register, and set the STBMOD bit in the PMU\_CTL register, and clear the WUF bit in the PMU\_CS register. Then, the device enters the Standby mode after a WFI or WFE instruction is executed. There are five wakeup sources for the Standby mode, including the external reset from NRST pin, the RTC alarm/time stamp/tamper events, the FWDGT reset, and the rising edge on WKUP0 or WKUP1 pin. The Standby mode achieves the lowest power consumption, but spends longest time to wake up. Besides, the contents of SRAM and registers (except Backup Registers) are lost in Standby mode. When exiting from the Standby mode, a power-on reset occurs and the Cortex™-M3 will execute instruction code from the 0x0000 0000 address.

**Table 3-1. Power saving mode summary**

Mode	Sleep	Deep-sleep	Standby
<b>Description</b>	Only CPU clock is off	<ol style="list-style-type: none"> <li>All clocks in the 1.2V domain for GD32F130/150xx devices or 1.8V domain for GD32F170/190xx devices are off</li> <li>Disable IRC8M, IRC14M for GD32F130/150xx devices or IRC28M for GD32F170/190xx devices, HXTAL and PLL</li> </ol>	<ol style="list-style-type: none"> <li>1.2V domain for GD32F130/150xx devices or 1.8V domain for GD32F170/190xx devices is power off</li> <li>Disable IRC8M, IRC14M for GD32F130/150xx devices or IRC28M for GD32F170/190xx devices, HXTAL and PLL</li> </ol>
<b>LDO Status</b>	On	On or in low power mode	Off
<b>Configuration</b>	SLEEPDEEP = 0	SLEEPDEEP = 1 STBMOD = 0	SLEEPDEEP = 1 STBMOD = 1, WURST = 1
<b>Entry</b>	WFI or WFE	WFI or WFE	WFI or WFE
<b>Wakeup</b>	Any interrupt for WFI Any event for WFE	Any interrupt or event from EXTI lines	<ol style="list-style-type: none"> <li>NRST pin</li> <li>RTC</li> <li>FWDGT reset</li> <li>WKUP0 pin</li> <li>WKUP1 pin</li> </ol>
<b>Wakeup Latency</b>	None	IRC8M wakeup time LDO wakeup time added if LDO is in low power mode	Power on sequence

### 3.4. Register definition

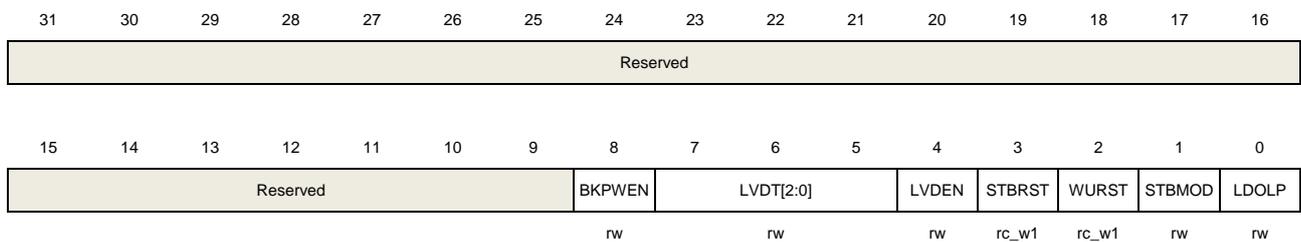
#### 3.4.1. Control register (PMU\_CTL)

For GD32F130xx and GD32F150xx devices

Address offset: 0x00

Reset value: 0x0000 0000 (reset by wakeup from Standby mode)

This register can be accessed by half-word(16-bit) or word(32-bit).



Bits	Fields	Descriptions
31:9	Reserved	Must be kept at reset value.
8	BKPWEN	Backup Domain Write Enable 0: Disable write access to the registers in Backup domain 1: Enable write access to the registers in Backup domain After reset, any write access to the registers in Backup domain is ignored. This bit has to be set to enable write access to these registers.
7:5	LVDT[2:0]	Low Voltage Detector Threshold 000: 2.2V 001: 2.3V 010: 2.4V 011: 2.5V 100: 2.6V 101: 2.7V 110: 2.8V 111: 2.9V
4	LVDEN	Low Voltage Detector Enable 0: Disable Low Voltage Detector 1: Enable Low Voltage Detector <b>Note:</b> When LVD_LOCK bit is set to 1 in the SYSCFG_CFG2 register, LVDEN and LVDT[2:0] are read only.
3	STBRST	Standby Flag Reset 0: No effect 1: Reset the standby flag

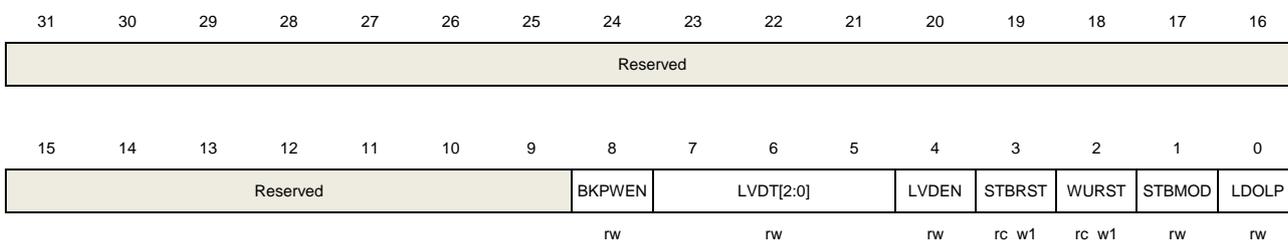
		This bit is always read as 0.
2	WURST	<p>Wakeup Flag Reset</p> <p>0: No effect</p> <p>1: Reset the wakeup flag</p> <p>This bit is always read as 0.</p>
1	STBMOD	<p>Standby Mode</p> <p>0: Enter the Deep-sleep mode when the Cortex™-M3 enters SLEEPDEEP mode</p> <p>1: Enter the Standby mode when the Cortex™-M3 enters SLEEPDEEP mode</p>
0	LDOLP	<p>LDO Low Power Mode</p> <p>0: The LDO operates normally during the Deep-sleep mode</p> <p>1: The LDO is in low power mode during the Deep-sleep mode</p> <p><b>Note:</b> Some peripherals may work with the IRC8M clock in the Deep-sleep mode. In this case, the LDO automatically switches from the low power mode to the normal mode and remains in this mode until the peripheral stop working.</p>

### For GD32F170xx and GD32F190xx devices

Address offset: 0x00

Reset value: 0x0000 0000 (reset by wakeup from Standby mode)

This register can be accessed by half-word(16-bit) or word(32-bit).



Bits	Fields	Descriptions
31:9	Reserved	Must be kept at reset value.
8	BKPWEN	<p>Backup Domain Write Enable</p> <p>0: Disable write access to the registers in Backup domain</p> <p>1: Enable write access to the registers in Backup domain</p> <p>After reset, any write access to the registers in Backup domain is ignored. This bit has to be set to enable write access to these registers.</p>
7:5	LVDT[2:0]	<p>Low Voltage Detector Threshold</p> <p>000: 2.4V</p> <p>001: 2.7V</p> <p>010: 3.0V</p> <p>011: 3.3V</p> <p>100: 3.6V</p>

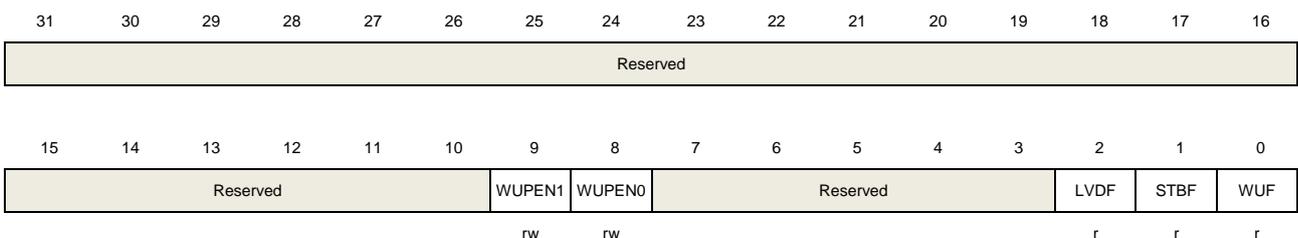
		101: 3.9V
		110: 4.2V
		111: 4.5V
4	LV DEN	Low Voltage Detector Enable 0: Disable Low Voltage Detector 1: Enable Low Voltage Detector <b>Note:</b> When LVD_LOCK bit is set to 1 in the SYSCFG_CFG2 register, LV DEN and LVDT[2:0] are read only.
3	STBRST	Standby Flag Reset 0: No effect 1: Reset the standby flag This bit is always read as 0.
2	WURST	Wakeup Flag Reset 0: No effect 1: Reset the wakeup flag This bit is always read as 0.
1	STBMOD	Standby Mode 0: Enter the Deep-sleep mode when the Cortex™-M3 enters SLEEPDEEP mode 1: Enter the Standby mode when the Cortex™-M3 enters SLEEPDEEP mode
0	LDOLP	LDO Low Power Mode 0: the LDO operates normally during the Deep-sleep mode 1: the LDO is in low power mode during the Deep-sleep mode <b>Note:</b> Some peripherals may work with the IRC8M clock in the Deep-sleep mode. In this case, the LDO automatically switches from the low power mode to the normal mode and remains in this mode until the peripheral stop working.

### 3.4.2. Power control/status register (PMU\_CS)

Address offset: 0x04

Reset value: 0x0000 0000 (not reset by wakeup from Standby mode)

This register can be accessed by half-word(16-bit) or word(32-bit).



Bits	Fields	Descriptions
------	--------	--------------

31:10	Reserved	Must be kept at reset value.
9	WUPEN1	<p>WKUP1 Pin (PC13) Enable</p> <p>0: Disable WKUP1 pin function</p> <p>1: Enable WKUP1 pin function</p> <p>If WUPEN1 is set before entering the power saving mode, a rising edge on the WKUP1 pin wakes up the system from the power saving mode. As the WKUP1 pin is active high, the WKUP1 pin is internally configured to input pull down mode. And set this bit will trigger a wakeup event when the input is already high.</p>
8	WUPEN0	<p>WKUP0 Pin (PA0) Enable</p> <p>0: Disable WKUP0 pin function</p> <p>1: Enable WKUP0 pin function</p> <p>If WUPEN0 is set before entering the power saving mode, a rising edge on the WKUP0 pin wakes up the system from the power saving mode. As the WKUP0 pin is active high, the WKUP0 pin is internally configured to input pull down mode. And set this bit will trigger a wakeup event when the input is already high.</p>
7:3	Reserved	Must be kept at reset value.
2	LVDF	<p>Low Voltage Detector Status Flag</p> <p>0: Low Voltage event has not occurred (<math>V_{DD}</math> is higher than the specified LVD threshold)</p> <p>1: Low Voltage event occurred (<math>V_{DD}</math> is equal to or lower than the specified LVD threshold)</p> <p><b>Note:</b> The LVD function is stopped in Standby mode.</p>
1	STBF	<p>Standby Flag</p> <p>0: The device has not entered the Standby mode</p> <p>1: The device has been in Standby mode</p> <p>This bit is cleared only by a POR/PDR or by setting the STBRST bit in the PMU_CTL register.</p>
0	WUF	<p>Wakeup Flag</p> <p>0: No wakeup event has been received</p> <p>1: Wakeup event occurred from the WKUP pin or the RTC wakeup event including RTC Tamper event, RTC alarm event, RTC Time Stamp event</p> <p>This bit is cleared only by a POR/PDR or by setting the WURST bit in the PMU_CTL register.</p>

## 4. Reset and clock unit (RCU)

### 4.1. Reset control unit (RCTL)

#### 4.1.1. Overview

GD32F1x0 Reset Control includes the control of three kinds of reset: power reset, system reset and backup domain reset. The power on reset, known as a cold reset, resets the full system except the Backup domain during a power up. A system reset resets the processor core and peripheral IP components with the exception of the SW-DP controller and the Backup domain. A backup domain reset resets the Backup domain. The resets can be triggered by an external signal, internal events and the reset generators. More information about these resets will be described in the following sections.

#### 4.1.2. Function overview

##### Power Reset

The Power reset is generated by either an external reset as Power On and Power Down reset (POR/PDR reset), or by the internal reset generator when exiting Standby mode. The power reset sets all registers to their reset values except the Backup domain. The Power reset which active signal is low will be de-asserted when the internal LDO voltage regulator is ready to provide 1.2V power for GD32F130/150xx devices or 1.8V power for GD32F170/190xx devices power. The RESET service routine vector is fixed at address 0x0000\_0004 in the memory map.

##### System Reset

A system reset is generated by the following events:

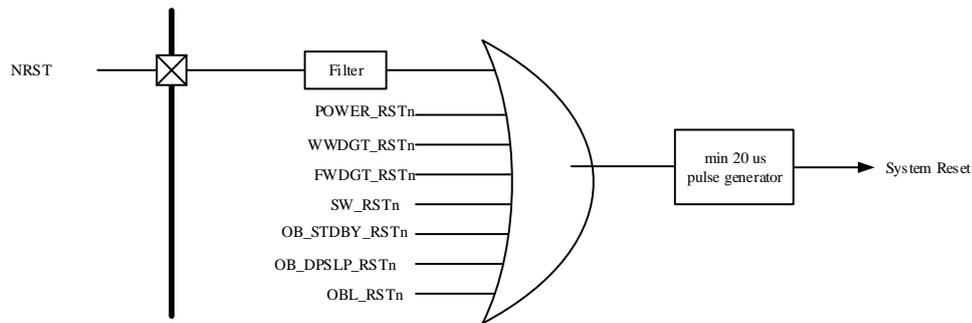
- A power on reset (POWER\_RSTn)
- A external pin reset (NRST)
- A window watchdog timer reset (WWDGT\_RSTn)
- A free watchdog timer reset (FWDGT\_RSTn)
- The SYSRESETREQ bit in Cortex™-M3 Application Interrupt and Reset Control Register is set (SW\_RSTn)
- Option byte loader reset (OBL\_RSTn)
- Reset generated when entering Standby mode when resetting nRST\_STDBY bit in User Option Bytes (OB\_STDBY\_RSTn)
- Reset generated when entering Deep-sleep mode when resetting nRST\_DSLP bit in

## User Option Bytes (OB\_DPSLP\_RSTn)

A system reset resets the processor core and peripheral IP components except for the SW-DP controller and the Backup domain.

A system reset pulse generator guarantees low level pulse duration of 20  $\mu$ s for each reset source (external or internal reset).

**Figure 4-1. The system reset circuit**



### Backup domain reset

A backup domain reset is generated by setting the BKPRST bit in the Backup domain control register or Backup domain power on reset ( $V_{DD}$  or  $V_{BAT}$  power on, if both supplies have previously been powered off).

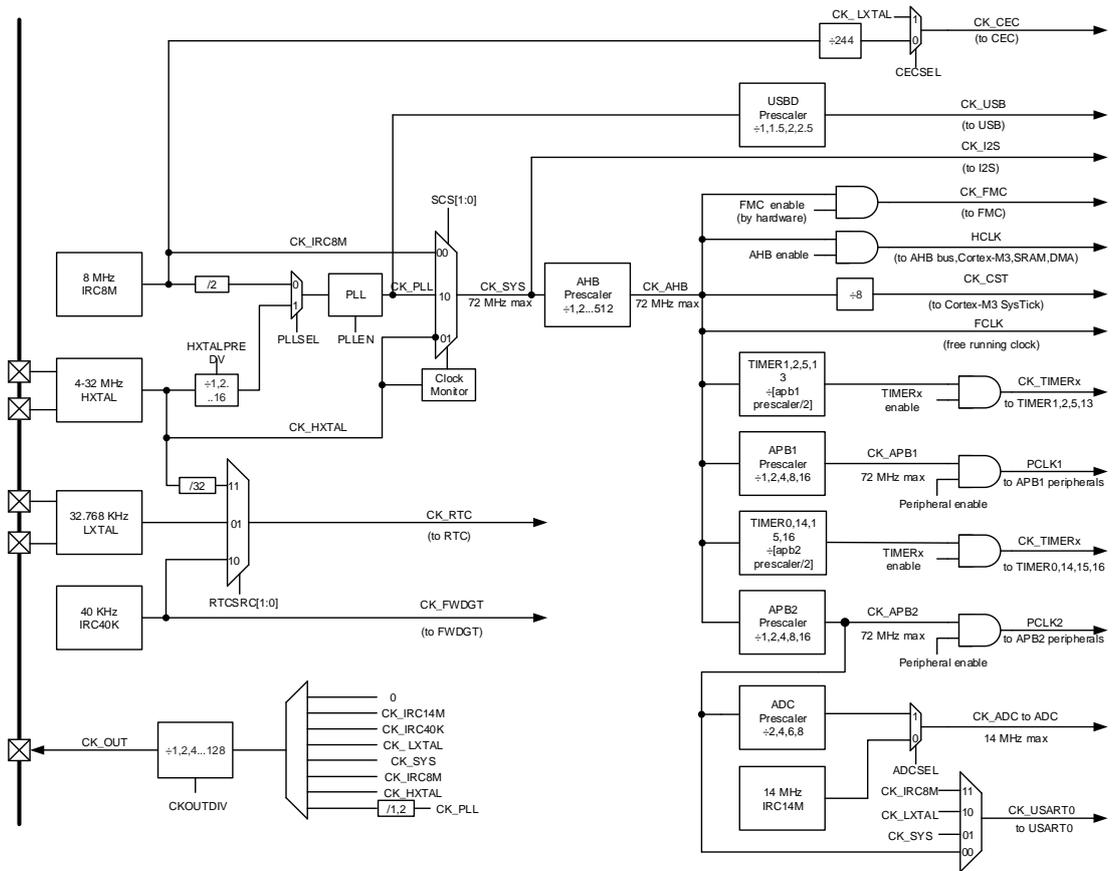
## 4.2. Clock control unit (CCTL)

### 4.2.1. Overview

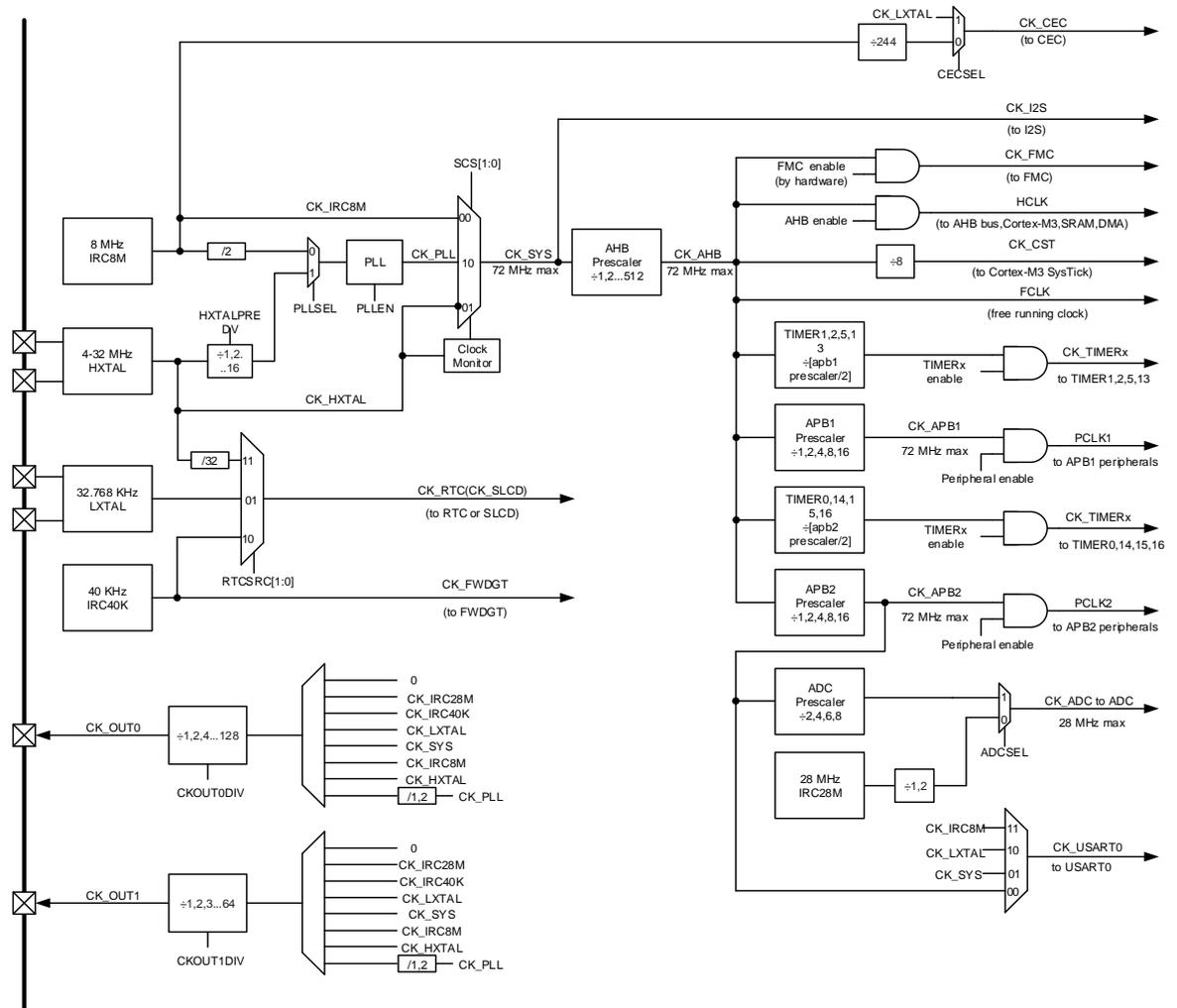
The Clock Control unit provides a range of frequencies and clock functions. These include a Internal 8 MHz RC oscillator (IRC8M), a Internal 14 MHz RC oscillator (IRC14M) for GD32F130xx and GD32F150xx devices or a Internal 28 MHz RC oscillator (IRC28M) for GD32F170xx and GD32F190xx devices, a High speed crystal oscillator (HXTAL), Internal 40KHz RC oscillator (IRC40K), a Low speed crystal oscillator (LXTAL), a Phase Lock Loop (PLL), a HXTAL clock monitor, clock prescalers, clock multiplexers and clock gating circuitry.

The clocks of the AHB, APB and Cortex™-M3 are derived from the system clock (CK\_SYS) which can source from the IRC8M, HXTAL or PLL. The maximum operating frequency of the system clock (CK\_SYS) can be up to 72 MHz. The Free Watchdog Timer has independent clock source (IRC40K), and Real Time Clock (RTC) use the IRC40K, LXTAL or HXTAL/32 as its clock source.

**Figure 4-2. Clock tree of GD32F130xx and GD32F150xx devices**



**Figure 4-3. Clock tree of GD32F170xx and GD32F190xx devices**



The frequency of AHB, APB2 and the APB1 domains can be configured by each prescaler. The maximum frequency of the AHB and the APB2/APB1 domains is 72 MHz. When using I2Cx peripherals, APB1 clock need to ensure that no more than 36MHz. The Cortex System Timer (SysTick) external clock is clocked with the AHB clock (HCLK) divided by 8. The SysTick can work either with this clock or with the Cortex clock (HCLK), configurable in the SysTick Control and Status Register.

The ADC are clocked by the clock of APB2 divided by 2, 4, 6, 8 or IRC14M clock selected for GD32F130xx and GD32F150xx devices or 2, 4, 6, 8 or IRC28M or IRC28M/2 clock for GD32F170xx and GD32F190xx devices selected by ADCSEL bit in Configuration register 2 (RCU\_CFG2). The USART0 is clocked by IRC8M clock or LXTAL clock or system clock or APB2 clock, which selected by USART0SEL bits in Configuration register 2 (RCU\_CFG2). The CEC clock is clocked by IRC8M divided 244 or LXTAL clock which selected by CECSEL bit in Configuration register 2 (RCU\_CFG2).

The RTC or SLCD (for GD32F170xx and GD32F190xx devices only) is clocked by LXTAL clock or IRC40K clock or HXTAL clock divided by 32 which select by RTCSRC bit in Backup

Domain Control Register (RCU\_BDCTL).

The FWDGT is clocked by IRC40K clock, which is forced on when FWDGT started.

If the APB prescaler is 1, the timer clock frequencies are set to AHB frequency divide by 1. Otherwise, they are set to the AHB frequency divide by half of APB prescaler.

## 4.2.2. Characteristics

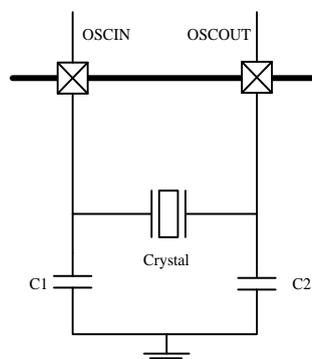
- 4 to 32 MHz High speed crystal oscillator (HXTAL)
- Internal 8 MHz RC oscillator (IRC8M)
- Internal 14 MHz RC oscillator (IRC14M) for GD32F130xx and GD32F150xx devices or Internal 28 MHz RC oscillator (IRC28M) for GD32F170xx and GD32F190xx devices
- 32,768 Hz Low speed crystal oscillator (LXTAL)
- Internal 40KHz RC oscillator (IRC40K)
- PLL clock source can be HXTAL or IRC8M
- HXTAL clock monitor

## 4.2.3. Function overview

### High Speed Crystal Oscillator (HXTAL)

The high speed crystal oscillator (HXTAL), which has a frequency from 4 to 32 MHz, produces a highly accurate clock source for use as the system clock. A crystal with a specific frequency must be connected and located close to the two HXTAL pins. The external resistor and capacitor components connected to the crystal are necessary for proper oscillation.

**Figure 4-4. HXTAL clock source**



The HXTAL crystal oscillator can be switched on or off using the HXTALEN bit in the Control register 0, RCU\_CTL0. The HXTALSTB flag in Control register 0, RCU\_CTL0 indicates if the high-speed external crystal oscillator is stable. When the HXTAL is powered up, it will not be

released for use until this HXTALSTB bit is set by the hardware. This specific delay period is known as the oscillator “Start-up time”. As the HXTAL becomes stable, an interrupt will be generated if the related interrupt enable bit HXTALSTBIE in the Interrupt register RCU\_INT is set. At this point the HXTAL clock can be used directly as the system clock source or the PLL input clock.

Select external clock bypass mode by setting the HXTALBPS and HXTALEN bits in the Control register 0, RCU\_CTL0. The CK\_HXTAL is equal to the external clock which drives the OSCIN pin.

#### **Internal 8 MHz RC Oscillator (IRC8M)**

The Internal 8 MHz RC oscillator, IRC8M, has a fixed frequency of 8 MHz and is the default clock source selection for the CPU when the device is powered up. The IRC8M oscillator provides a lower cost type clock source as no external components are required. The IRC8M RC oscillator can be switched on or off using the IRC8MEN bit in the Control register 0, RCU\_CTL0. The IRC8MSTB flag in the Control register 0, RCU\_CTL0 is used to indicate if the internal RC oscillator is stable. The start-up time of the IRC8M oscillator is shorter than the HXTAL crystal oscillator. An interrupt can be generated if the related interrupt enable bit, IRC8MSTBIE, in the Interrupt register, RCU\_INT, is set when the IRC8M becomes stable. The IRC8M clock can also be used as the PLL input clock.

The frequency accuracy of the IRC8M can be calibrated by the manufacturer, but its operating frequency is still less accurate than HXTAL. The application requirements, environment and cost will determine which oscillator type is selected.

If the HXTAL or PLL is the system clock source, to minimize the time required for the system to recover from the Deep-sleep Mode, the hardware forces the IRC8M clock to be the system clock when the system initially wakes-up.

#### **Phase Locked Loop (PLL)**

The internal Phase Locked Loop, PLL, can provide 16~72 MHz clock output which is 2 ~32 multiples of a fundamental reference frequency of 4 ~ 32 MHz.

The PLL can be switched on or off by using the PLEN bit in the Control register 0, RCU\_CTL0. The PLLSTB flag in the Control register 0, RCU\_CTL0 will indicate if the PLL clock is stable. An interrupt can be generated if the related interrupt enable bit, PLLSTBIE, in the Interrupt register, RCU\_INT, is set as the PLL becomes stable.

#### **Internal 14 MHz RC Oscillator (IRC14M) for GD32F130xx and GD32F150xx devices**

The Internal 14 MHz RC Oscillator, IRC14M, has a fixed frequency of 14 MHz and dedicated as ADC clock. The IRC14M RC oscillator can be switched on or off using the IRC14MEN bit in the Control register 1 (RCU\_CTL1). The IRC14MSTB flag in the Control register 1 (RCU\_CTL1) is used to indicate if the internal 14M RC oscillator is stable. An interrupt can be generated if the related interrupt enable bit, IRC14MSTBIE, in the Interrupt register, RCU\_INT, is set when the IRC14M becomes stable.

**Internal 28 MHz RC Oscillator (IRC28M) for GD32F170xx and GD32F190xx devices**

The Internal 28 MHz RC Oscillator, IRC28M, has a fixed frequency of 28 MHz and dedicated as ADC clock. The IRC28M RC oscillator can be switched on or off using the IRC28MEN bit in the Control register 1 (RCU\_CTL1). The IRC28MSTB flag in the Control register 1 (RCU\_CTL1) is used to indicate if the internal 28M RC oscillator is stable. An interrupt can be generated if the related interrupt enable bit, IRC28MSTBIE, in the Interrupt register, RCU\_INT, is set when the IRC28M becomes stable.

**Low Speed Crystal Oscillator (LXTAL)**

The low speed crystal or ceramic resonator oscillator, which has a frequency of 32,768 Hz, produces a low power but highly accurate clock source for the Real Time Clock circuit. The LXTAL oscillator can be switched on or off using the LXTALEN bit in the Backup Domain Control Register (RCU\_BDCTL). The LXTALSTB flag in the Backup Domain Control Register (RCU\_BDCTL) will indicate if the LXTAL clock is stable. An interrupt can be generated if the related interrupt enable bit, LXTALSTBIE, in the Interrupt register RCU\_INT is set when the LXTAL becomes stable.

Select external clock bypass mode by setting the LXTALBPS and LXTALEN bits in the Backup Domain Control Register (RCU\_BDCTL). The CK\_LXTAL is equal to the external clock which drives the OSC32IN pin.

**Internal 40KHz RC Oscillator (IRC40K)**

The Internal 40KHz RC Oscillator has a frequency of about 40 kHz and is a low power clock source for the Real Time Clock circuit or the Free Watchdog Timer. The IRC40K offers a low cost clock source as no external components are required. The IRC40K RC oscillator can be switched on or off by using the IRC40KEN bit in the Reset Source/Clock Register, RCU\_RSTSCK. The IRC40KSTB flag in the Reset Source/Clock Register RCU\_RSTSCK will indicate if the IRC40K clock is stable. An interrupt can be generated if the related interrupt enable bit IRC40KSTBIE in the Interrupt register RCU\_INT is set when the IRC40K becomes stable.

**System Clock (CK\_SYS) Selection**

After the system reset, the default CK\_SYS source will be IRC8M and can be switched to HXTAL or PLL by changing the System Clock Switch bits, SCS, in the Configuration register 0, RCU\_CFG0. When the SCS value is changed, the CK\_SYS will continue to operate using the original clock source until the target clock source is stable. When a clock source is used directly by the CK\_SYS or the PLL, it is not possible to stop it.

**HXTAL Clock Monitor (CKM)**

The HXTAL clock monitor function is enabled by the HXTAL Clock Monitor Enable bit, CKMEN, in the Control register 0, RCU\_CTL0. This function should be enabled after the HXTAL start-up delay and disabled when the HXTAL is stopped. Once the HXTAL failure is detected, the HXTAL will be automatically disabled. The HXTAL Clock Stuck Flag, CKMIF, in the Interrupt register, RCU\_INT, will be set and the HXTAL failure event will be generated.

This failure interrupt is connected to the Non-Maskable Interrupt, NMI, of the Cortex-M3. If the HXTAL is selected as the clock source of CK\_SYS or PLL, the HXTAL failure will force the CK\_SYS source to IRC8M and the PLL will be disabled automatically

## Clock Output Capability

### For GD32F130xx and GD32F150xx devices

The clock output capability is ranging from 32 kHz to 72 MHz. There are several clock signals can be selected via the CK\_OUT Clock Source Selection bits, CKOUTSEL, in the Configuration register 0 (RCU\_CFG0). The corresponding GPIO pin should be configured in the properly Alternate Function I/O (AFIO) mode to output the selected clock signal.

**Table 4-1. Clock source select**

Clock Source Selection bits	Clock Source
000	No Clock
001	CK_IRC14M
010	CK_IRC40K
011	CK_LXTAL
100	CK_SYS
101	CK_IRC8M
110	CK_HXTAL
111	CK_PLL or CK_PLL/2

The CK\_OUT frequency can be reduced by a configurable binary divider, controlled by the CKOUTDIV[2:0] bits , in the Configuration register 0 (RCU\_CFG0).

### For GD32F170xx and GD32F190xx devices

The clock output capability is ranging from 32 kHz to 72 MHz. There are several clock signals can be selected via the CK\_OUT0 Clock Source Selection bits, CKOUT0SEL, in the Configuration register 0 (RCU\_CFG0) and CKOUT1SRC in the Configuration register 3 (RCU\_CFG3). The corresponding GPIO pin should be configured in the properly Alternate Function I/O (AFIO) mode to output the selected clock signal.

**Table 4-2. Clock source select**

Clock Source Selection bits	Clock Source
000	No Clock
001	CK_IRC28M
010	CK_IRC40K
011	CK_LXTAL
100	CK_SYS
101	CK_IRC8M
110	CK_HXTAL
111	CK_PLL or CK_PLL/2

The CK\_OUT0 frequency can be reduced by a configurable binary divider, controlled by the CKOUT0DIV[2:0] bits , in the Configuration register 0 (RCU\_CFG0).

The CK\_OUT1 is selected by CKOUT1SRC, in the Configuration register 3 (RCU\_CFG3). The CK\_OUT1 frequency can be reduced by a configurable binary divider, controlled by the CKOUT1DIV[2:0] bits, in the Configuration register 3 (RCU\_CFG3).

### Deep-sleep mode clock control

When the MCU is in Deep-sleep mode, the HDMI CEC or USART0 can wake up the MCU, when their clock is provided by LXTAL clock and LXTAL clock is enable.

If the HDMI CEC or USART0 clock is selected IRC8M clock in Deep-sleep mode, they have capable of open IRC8M clock or close IRC8M clock, which used to the HDMI CEC or USART0 to wake up the Deep-sleep mode.

### Voltage control

#### For GD32F130xx and GD32F150xx devices

The power down reset can be controlled by PDRVS bit in the Power down voltage select register (RCU\_PDVSEL). If the PDRVS bit is 0, the power down reset assert when the  $V_{DD}$  is below 2.6V. If the PDRVS bit is 1, the power down reset assert when the  $V_{DD}$  is below 1.8V. When the PDRVS bit is 1 and  $V_{DD}$  is below 2.6V, the flash program/erase cannot be used.

The core domain voltage in Deep-sleep mode can be controlled by DSLPVS[2:0] bit in the Deep-sleep mode voltage register (RCU\_DSV).

**Table 4-3. Core domain voltage selected in Deep-sleep mode -**

DSLPVS[2:0]	Deep-sleep mode voltage(V)
000	1.2
001	1.1
010	1.0
011	0.9
100 ~ 111	reserved

The RCU\_PDVSEL and RCU\_DSV register are protected by Voltage Key register (RCU\_VKEY). Only after write 0x1A2B3C4D to the RCU\_VKEY register, the RCU\_PDVSEL and RCU\_DSV register can be write.

#### For GD32F170xx and GD32F190xx devices

The core domain voltage in Deep-sleep mode can be controlled by DSLPVS[2:0] bit in the Deep-sleep mode voltage register (RCU\_DSV).

**Table 4-4. Core domain voltage selected in Deep-sleep mode -**

DSLPVS[2:0]	Deep-sleep mode voltage(V)
000	1.8
001	1.6
010	1.4
011	1.2
100 ~ 111	reserved

The RCU\_DSV register are protected by Voltage Key register (RCU\_VKEY). Only after write 0x1A2B3C4D to the RCU\_VKEY register, the RCU\_DSV register can be write.

## 4.3. Register definition

### 4.3.1. Control register 0 (RCU\_CTL0)

Address offset: 0x00

Reset value: 0x0000 XX83 where X is undefined.

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved						PLLSTB	PLLEN	Reserved				CKMEN	HXTALB PS	HXTALS TB	HXTALE N
						r	rw					rw	rw	r	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IRC8MCALIB[7:0]								IRC8MADJ[4:0]				Reserve d.	IRC8MS TB	IRC8ME N	
r								rw					r	rw	

Bits	Fields	Descriptions
31:26	Reserved	Must be kept at reset value.
25	PLLSTB	PLL Clock Stabilization Flag Set by hardware to indicate if the PLL output clock is stable and ready for use. 0: PLL is not stable 1: PLL is stable
24	PLLEN	PLL enable Set and reset by software. This bit cannot be reset if the PLL clock is used as the system clock. Reset by hardware when entering Deep-sleep or Standby mode. 0: PLL is switched off 1: PLL is switched on
23:20	Reserved	Must be kept at reset value.
19	CKMEN	HXTAL Clock Monitor Enable 0: Disable the External 4 ~ 32 MHz crystal oscillator (HXTAL) clock monitor 1: Enable the External 4 ~ 32 MHz crystal oscillator (HXTAL) clock monitor When the hardware detects that the HXTAL clock is stuck at a low or high state, the internal hardware will switch the system clock to be the internal high speed IRC8M RC clock. The way to recover the original system clock is by either an external reset, power on reset or clearing CKMIF by software. <b>Note:</b> When the HXTAL clock monitor is enabled, the hardware will automatically enable the IRC8M internal RC oscillator regardless of the control bit, IRC8MEN, state.
18	HXTALBPS	External crystal oscillator (HXTAL) clock bypass mode enable The HXTALBPS bit can be written only if the HXTALEN is 0.

		0: Disable the HXTAL Bypass mode 1: Enable the HXTAL Bypass mode in which the HXTAL output clock is equal to the input clock.
17	HXTALSTB	External crystal oscillator (HXTAL) clock stabilization flag Set by hardware to indicate if the HXTAL oscillator is stable and ready for use. 0: HXTAL oscillator is not stable 1: HXTAL oscillator is stable
16	HXTALEN	External High Speed oscillator Enable Set and reset by software. This bit cannot be reset if the HXTAL clock is used as the system clock or the PLL input clock. Reset by hardware when entering Deep-sleep or Standby mode. 0: External 4 ~ 32 MHz crystal oscillator disabled 1: External 4 ~ 32 MHz crystal oscillator enabled
15:8	IRC8MCALIB[7:0]	High Speed Internal Oscillator calibration value register These bits are load automatically at power on.
7:3	IRC8MADJ[4:0]	High Speed Internal Oscillator clock trim adjust value These bits are set by software. The trimming value is there bits (IRC8MADJ) added to the IRC8MCALIB[7:0] bits. The trimming value should trim the IRC8M to 8 MHz $\pm$ 1%.
2	Reserved	Must be kept at reset value.
1	IRC8MSTB	IRC8M High Speed Internal Oscillator stabilization Flag Set by hardware to indicate if the IRC8M oscillator is stable and ready for use. 0: IRC8M oscillator is not stable 1: IRC8M oscillator is stable
0	IRC8MEN	Internal High Speed oscillator Enable Set and reset by software. This bit cannot be reset if the IRC8M clock is used as the system clock. Set by hardware when leaving Deep-sleep or Standby mode or the HXTAL clock is stuck at a low or high state when HXTALCKM is set. 0: Internal 8 MHz RC oscillator disabled 1: Internal 8 MHz RC oscillator enabled

## 4.3.2. Configuration register 0 (RCU\_CFG0)

### For GD32F130xx and GD32F150xx devices

Address offset: 0x04

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

PLLDV	CKOUTDIV[2:0]	PLLMF[4]	CKOUTSEL[2:0]	USBDPSC[1:0]	PLLMF[3:0]	PLLPRE DV	PLLSEL
rw	rw	rw	rw	rw	rw	rw	rw
15	14 13 12	11	10 9 8	7 6	5 4 3 2	1	0
ADCPSC[1:0]	APB2PSC[2:0]	APB1PSC[2:0]	AHBPSC[3:0]	SCSS[1:0]	SCS[1:0]		
rw	rw	rw	rw	r	rw		

Bits	Fields	Descriptions
31	PLLDV	The CK_PLL divide by 1 or 2 for CK_OUT 0: CK_PLL divide by 2 for CK_OUT 1: CK_PLL divide by 1 for CK_OUT
30:28	CKOUTDIV[2:0]	The CK_OUT divider which the CK_OUT frequency can be reduced see bits 26:24 of RCU_CFG0 for CK_OUT. 000: The CK_OUT is divided by 1 001: The CK_OUT is divided by 2 010: The CK_OUT is divided by 4 011: The CK_OUT is divided by 8 100: The CK_OUT is divided by 16 101: The CK_OUT is divided by 32 110: The CK_OUT is divided by 64 111: The CK_OUT is divided by 128
27	PLLMF[4]	Bit 4 of PLLMF register see bits 21:18 of RCU_CFG0.
26:24	CKOUTSEL[2:0]	CK_OUT Clock Source Selection Set and reset by software. 000: No clock selected 001: Internal 14M RC oscillator clock selected 010: Internal 40K RC oscillator clock selected 011: External Low Speed oscillator clock selected 100: System clock selected 101: Internal 8MHz RC Oscillator clock selected 110: External High Speed oscillator clock selected 111: (CK_PLL / 2) or CK_PLL selected depend on PLLDV
23:22	USBDPSC[1:0]	USB clock prescaler selection Set and reset by software to control the USB clock prescaler value. The USB clock must be 48MHz. These bits can't be reset if the USB clock is enabled. 00: (CK_PLL / 1.5) selected 01: CK_PLL selected 10: (CK_PLL / 2.5) selected 11: (CK_PLL / 2) selected

21:18	PLLMF[3:0]	<p>PLL multiply factor</p> <p>These bits and bit 27 of RCU_CFG0 are written by software to define the PLL multiplication factor.</p> <p>00000: (PLL source clock x 2)</p> <p>00001: (PLL source clock x 3)</p> <p>00010: (PLL source clock x 4)</p> <p>00011: (PLL source clock x 5)</p> <p>00100: (PLL source clock x 6)</p> <p>00101: (PLL source clock x 7)</p> <p>00110: (PLL source clock x 8)</p> <p>00111: (PLL source clock x 9)</p> <p>01000: (PLL source clock x 10)</p> <p>01001: (PLL source clock x 11)</p> <p>01010: (PLL source clock x 12)</p> <p>01011: (PLL source clock x 13)</p> <p>01100: (PLL source clock x 14)</p> <p>01101: (PLL source clock x 15)</p> <p>01110: (PLL source clock x 16)</p> <p>01111: (PLL source clock x 16)</p> <p>10000: (PLL source clock x 17)</p> <p>10001: (PLL source clock x 18)</p> <p>10010: (PLL source clock x 19)</p> <p>10011: (PLL source clock x 20)</p> <p>10100: (PLL source clock x 21)</p> <p>10101: (PLL source clock x 22)</p> <p>10110: (PLL source clock x 23)</p> <p>10111: (PLL source clock x 24)</p> <p>11000: (PLL source clock x 25)</p> <p>11001: (PLL source clock x 26)</p> <p>11010: (PLL source clock x 27)</p> <p>11011: (PLL source clock x 28)</p> <p>11100: (PLL source clock x 29)</p> <p>11101: (PLL source clock x 30)</p> <p>11110: (PLL source clock x 31)</p> <p>11111: (PLL source clock x 32)</p> <p><b>Note:</b> The PLL output frequency must not exceed 72 MHz.</p>
17	PLLPREDV	<p>HXTAL divider for PLL source clock selection. This bit is the same bit as bit HXTALPREDV[0] from RCU_CFG1. Refer to RCU_CFG1 HXTALPREDV bits description.</p> <p>Set and cleared by software to divide HXTAL or not which is selected to PLL.</p> <p>0: HXTAL clock selected</p>

		1: (CK_HXTAL / 2) clock selected
16	PLLSEL	<p>PLL Clock Source Selection</p> <p>Set and reset by software to control the PLL clock source.</p> <p>0: (CK_IRC8M / 2) selected as PLL source clock</p> <p>1: HXTAL selected as PLL source clock</p>
15:14	ADCPSC[1:0]	<p>ADC clock prescaler selection</p> <p>Set and cleared by software.</p> <p>00: (CK_APB2 / 2) selected</p> <p>01: (CK_APB2 / 4) selected</p> <p>10: (CK_APB2 / 6) selected</p> <p>11: (CK_APB2 / 8) selected</p>
13:11	APB2PSC[2:0]	<p>APB2 prescaler selection</p> <p>Set and reset by software to control the APB2 clock division ratio.</p> <p>0xx: CK_AHB selected</p> <p>100: (CK_AHB / 2) selected</p> <p>101: (CK_AHB / 4) selected</p> <p>110: (CK_AHB / 8) selected</p> <p>111: (CK_AHB / 16) selected</p>
10:8	APB1PSC[2:0]	<p>APB1 prescaler selection</p> <p>Set and reset by software to control the APB1 clock division ratio.</p> <p>0xx: CK_AHB selected</p> <p>100: (CK_AHB / 2) selected</p> <p>101: (CK_AHB / 4) selected</p> <p>110: (CK_AHB / 8) selected</p> <p>111: (CK_AHB / 16) selected</p>
7:4	AHBPSC[3:0]	<p>AHB prescaler selection</p> <p>Set and reset by software to control the AHB clock division ratio</p> <p>0xxx: CK_SYS selected</p> <p>1000: (CK_SYS / 2) selected</p> <p>1001: (CK_SYS / 4) selected</p> <p>1010: (CK_SYS / 8) selected</p> <p>1011: (CK_SYS / 16) selected</p> <p>1100: (CK_SYS / 64) selected</p> <p>1101: (CK_SYS / 128) selected</p> <p>1110: (CK_SYS / 256) selected</p> <p>1111: (CK_SYS / 512) selected</p>
3:2	SCSS[1:0]	<p>System clock switch status</p> <p>Set and reset by hardware to indicate the clock source of system clock.</p> <p>00: select CK_IRC8M as the CK_SYS source</p> <p>01: select CK_HXTAL as the CK_SYS source</p> <p>10: select CK_PLL as the CK_SYS source</p>

		11: reserved
1:0	SCSS[1:0]	<p>System clock switch</p> <p>Set by software to select the CK_SYS source. Because the change of CK_SYS has inherent latency, software should read SCSS to confirm whether the switching is complete or not. The switch will be forced to IRC8M when leaving Deep-sleep and Standby mode or by HXTAL clock monitor when the HXTAL failure is detected and the HXTAL is selected as the clock source of CK_SYS or PLL.</p> <p>00: select CK_IRC8M as the CK_SYS source            01: select CK_HXTAL as the CK_SYS source            10: select CK_PLL as the CK_SYS source            11: reserved</p>

### For GD32F170xx and GD32F190xx devices

Address offset: 0x04

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PLLDV		CKOUT0DIV[2:0]		PLLMF[4]	CKOUT0SEL [2:0]		Reserved		PLLMF[3:0]			PLLPRE DV	PLLSEL		
rw		rw		rw	rw				rw			rw	rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADCPSC[1:0]		APB2PSC[2:0]		APB1PSC[2:0]		AHBPSC[3:0]			SCSS[1:0]		SCS[1:0]				
rw		rw		rw		rw			r		rw				

Bits	Fields	Descriptions
31	PLLDV	The CK_PLL divide by 1 or 2 for CK_OUT0 0: CK_PLL divide by 2 for CK_OUT0 1: CK_PLL divide by 1 for CK_OUT0
30:28	CKOUT0DIV[2:0]	The CK_OUT0 divider which the CK_OUT0 frequency can be reduced see bits 26:24 of RCU_CFG0 for CK_OUT0 000: The CK_OUT0 is divided by 1 001: The CK_OUT0 is divided by 2 010: The CK_OUT0 is divided by 4 011: The CK_OUT0 is divided by 8 100: The CK_OUT0 is divided by 16 101: The CK_OUT0 is divided by 32 110: The CK_OUT0 is divided by 64 111: The CK_OUT0 is divided by 128
27	PLLMF[4]	Bit 4 of PLLMF register

see bits 21:18 of RCU\_CFG0.

26:24	CKOUT0SEL[2:0]	<p>CKOUT0 Clock Source Selection</p> <p>Set and reset by software.</p> <p>000: No clock selected</p> <p>001: Internal 28MHz RC oscillator clock selected</p> <p>010: Internal 40K RC oscillator clock selected</p> <p>011: External Low Speed oscillator clock selected</p> <p>100: System clock selected</p> <p>101: Internal 8MHz RC Oscillator clock selected</p> <p>110: External High Speed oscillator clock selected</p> <p>111: (CK_PLL / 2) or CK_PLL selected depend on PLLDV</p>
23:22	Reserved	Must be kept at reset value.
21:18	PLLMF[3:0]	<p>PLL multiply factor</p> <p>These bits and bit 27 of RCU_CFG0 are written by software to define the PLL multiplication factor.</p> <p>00000: (PLL source clock x 2)</p> <p>00001: (PLL source clock x 3)</p> <p>00010: (PLL source clock x 4)</p> <p>00011: (PLL source clock x 5)</p> <p>00100: (PLL source clock x 6)</p> <p>00101: (PLL source clock x 7)</p> <p>00110: (PLL source clock x 8)</p> <p>00111: (PLL source clock x 9)</p> <p>01000: (PLL source clock x 10)</p> <p>01001: (PLL source clock x 11)</p> <p>01010: (PLL source clock x 12)</p> <p>01011: (PLL source clock x 13)</p> <p>01100: (PLL source clock x 14)</p> <p>01101: (PLL source clock x 15)</p> <p>01110: (PLL source clock x 16)</p> <p>01111: (PLL source clock x 16)</p> <p>10000: (PLL source clock x 17)</p> <p>10001: (PLL source clock x 18)</p> <p>10010: (PLL source clock x 19)</p> <p>10011: (PLL source clock x 20)</p> <p>10100: (PLL source clock x 21)</p> <p>10101: (PLL source clock x 22)</p> <p>10110: (PLL source clock x 23)</p> <p>10111: (PLL source clock x 24)</p> <p>11000: (PLL source clock x 25)</p> <p>11001: (PLL source clock x 26)</p> <p>11010: (PLL source clock x 27)</p>

		11011: (PLL source clock x 28)
		11100: (PLL source clock x 29)
		11101: (PLL source clock x 30)
		11110: (PLL source clock x 31)
		11111: (PLL source clock x 32)
		<b>Note:</b> The PLL output frequency must not exceed 72 MHz.
17	PLLPREDV	<p>HXTAL divider for PLL source clock selection. This bit is the same bit as bit HXTALPREDV[0] from RCU_CFG1. Refer to RCU_CFG1 HXTALPREDV bits description.</p> <p>Set and cleared by software to divide HXTAL or not which is selected to PLL.</p> <p>0: HXTAL clock selected</p> <p>1: (CK_HXTAL / 2) clock selected</p>
16	PLLSEL	<p>PLL Clock Source Selection</p> <p>Set and reset by software to control the PLL clock source.</p> <p>0: (CK_IRC8M / 2) selected as PLL source clock</p> <p>1: HXTAL selected as PLL source clock</p>
15:14	ADCPSC[1:0]	<p>ADC clock prescaler selection</p> <p>Set and cleared by software.</p> <p>00: (CK_APB2 / 2) selected</p> <p>01: (CK_APB2 / 4) selected</p> <p>10: (CK_APB2 / 6) selected</p> <p>11: (CK_APB2 / 8) selected</p>
13:11	APB2PSC[2:0]	<p>APB2 prescaler selection</p> <p>Set and reset by software to control the APB2 clock division ratio.</p> <p>0xx: CK_AHB selected</p> <p>100: (CK_AHB / 2) selected</p> <p>101: (CK_AHB / 4) selected</p> <p>110: (CK_AHB / 8) selected</p> <p>111: (CK_AHB / 16) selected</p>
10:8	APB1PSC[2:0]	<p>APB1 prescaler selection</p> <p>Set and reset by software to control the APB1 clock division ratio.</p> <p>0xx: CK_AHB selected</p> <p>100: (CK_AHB / 2) selected</p> <p>101: (CK_AHB / 4) selected</p> <p>110: (CK_AHB / 8) selected</p> <p>111: (CK_AHB / 16) selected</p>
7:4	AHBPSC[3:0]	<p>AHB prescaler selection</p> <p>Set and reset by software to control the AHB clock division ratio</p> <p>0xxx: CK_SYS selected</p> <p>1000: (CK_SYS / 2) selected</p> <p>1001: (CK_SYS / 4) selected</p>

- 1010: (CK\_SYS / 8) selected
  - 1011: (CK\_SYS / 16) selected
  - 1100: (CK\_SYS / 64) selected
  - 1101: (CK\_SYS / 128) selected
  - 1110: (CK\_SYS / 256) selected
  - 1111: (CK\_SYS / 512) selected
- 3:2 SCSS[1:0] System clock switch status  
Set and reset by hardware to indicate the clock source of system clock.  
00: select CK\_IRC8M as the CK\_SYS source  
01: select CK\_HXTAL as the CK\_SYS source  
10: select CK\_PLL as the CK\_SYS source  
11: reserved
- 1:0 SCS[1:0] System clock switch  
Set by software to select the CK\_SYS source. Because the change of CK\_SYS has inherent latency, software should read SCSS to confirm whether the switching is complete or not. The switch will be forced to IRC8M when leaving Deep-sleep and Standby mode or by HXTAL clock monitor when the HXTAL failure is detected and the HXTAL is selected as the clock source of CK\_SYS or PLL.  
00: select CK\_IRC8M as the CK\_SYS source  
01: select CK\_HXTAL as the CK\_SYS source  
10: select CK\_PLL as the CK\_SYS source  
11: reserved

### 4.3.3. Interrupt register (RCU\_INT)

#### For GD32F130xx and GD32F150xx devices

Address offset: 0x08

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								CKMIC	Reserved	IRC14MS TBIC	PLL STBIC	HXTAL STBIC	IRC8M STBIC	LXTAL STBIC	IRC40K STBIC
								w		w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		IRC14 MSTBIE	PLL STBIE	HXTAL STBIE	IRC8M STBIE	LXTAL STBIE	IRC40K STBIE	CKMIF	Reserved	IRC14M STBIF	PLL STBIF	HXTAL STBIF	IRC8M STBIF	LXTAL STBIF	IRC40K STBIF
		r	r	r	r	r	r	r		r	r	r	r	r	r

**Bits      Fields      Descriptions**

31:24	Reserved	Must be kept at reset value
23	CKMIC	HXTAL Clock Stuck Interrupt Clear Write 1 by software to reset the CKMIF flag. 0: Not reset CKMIF flag 1: Reset CKMIF flag
22	Reserved	Must be kept at reset value
21	IRC14MSTBIC	IRC14M stabilization Interrupt Clear Write 1 by software to reset the IRC14MSTBIF flag. 0: Not reset IRC14MSTBIF flag 1: Reset IRC14MSTBIF flag
20	PLLSTBIC	PLL stabilization Interrupt Clear Write 1 by software to reset the PLLSTBIF flag. 0: Not reset PLLSTBIF flag 1: Reset PLLSTBIF flag
19	HXTALSTBIC	HXTAL Stabilization Interrupt Clear Write 1 by software to reset the HXTALSTBIF flag. 0: Not reset HXTALSTBIF flag 1: Reset HXTALSTBIF flag
18	IRC8MSTBIC	IRC8M Stabilization Interrupt Clear Write 1 by software to reset the IRC8MSTBIF flag. 0: Not reset IRC8MSTBIF flag 1: Reset IRC8MSTBIF flag
17	LXTALSTBIC	LXTAL Stabilization Interrupt Clear Write 1 by software to reset the LXTALSTBIF flag. 0: Not reset LXTALSTBIF flag 1: Reset LXTALRDYF flag
16	IRC40KSTBIC	IRC40K Stabilization Interrupt Clear Write 1 by software to reset the IRC40KSTBIF flag. 0: Not reset IRC40KSTBIF flag 1: Reset IRC40KSTBIF flag
15:14	Reserved	Must be kept at reset value
13	IRC14MSTBIE	IRC14M Stabilization Interrupt Enable Set and reset by software to enable/disable the IRC14M stabilization interrupt. 0: Disable the IRC14M stabilization interrupt 1: Enable the IRC14M stabilization interrupt
12	PLLSTBIE	PLL Stabilization Interrupt Enable Set and reset by software to enable/disable the PLL stabilization interrupt. 0: Disable the PLL stabilization interrupt

		1: Enable the PLL stabilization interrupt
11	HXTALSTBIE	<p>HXTAL Stabilization Interrupt Enable</p> <p>Set and reset by software to enable/disable the HXTAL stabilization interrupt</p> <p>0: Disable the HXTAL stabilization interrupt</p> <p>1: Enable the HXTAL stabilization interrupt</p>
10	IRC8MSTBIE	<p>IRC8M Stabilization Interrupt Enable</p> <p>Set and reset by software to enable/disable the IRC8M stabilization interrupt</p> <p>0: Disable the IRC8M stabilization interrupt</p> <p>1: Enable the IRC8M stabilization interrupt</p>
9	LXTALSTBIE	<p>LXTAL Stabilization Interrupt Enable</p> <p>LXTAL stabilization interrupt enable/disable control</p> <p>0: Disable the LXTAL stabilization interrupt</p> <p>1: Enable the LXTAL stabilization interrupt</p>
8	IRC40KSTBIE	<p>IRC40K Stabilization interrupt enable</p> <p>IRC40K stabilization interrupt enable/disable control</p> <p>0: Disable the IRC40K stabilization interrupt</p> <p>1: Enable the IRC40K stabilization interrupt</p>
7	CKMIF	<p>HXTAL Clock Stuck Interrupt Flag</p> <p>Set by hardware when the HXTAL clock is stuck.</p> <p>Reset by software when setting the CKMIC bit.</p> <p>0: Clock operating normally</p> <p>1: HXTAL clock stuck</p>
6	Reserved	Must be kept at reset value
5	IRC14MSTBIF	<p>IRC14M stabilization interrupt flag</p> <p>Set by hardware when the IRC14M is stable and the IRC14MSTBIE bit is set.</p> <p>Reset by software when setting the IRC14MSTBIC bit.</p> <p>0: No IRC14M stabilization interrupt generated</p> <p>1: IRC14M stabilization interrupt generated</p>
4	PLLSTBIF	<p>PLL stabilization interrupt flag</p> <p>Set by hardware when the PLL is stable and the PLLSTBIE bit is set.</p> <p>Reset by software when setting the PLLSTBIC bit.</p> <p>0: No PLL stabilization interrupt generated</p> <p>1: PLL stabilization interrupt generated</p>
3	HXTALSTBIF	<p>HXTAL stabilization interrupt flag</p> <p>Set by hardware when the External 4 ~ 32 MHz crystal oscillator clock is stable and the HXTALSTBIE bit is set.</p> <p>Reset by software when setting the HXTALSTBIC bit.</p> <p>0: No HXTAL stabilization interrupt generated</p> <p>1: HXTAL stabilization interrupt generated</p>

2	IRC8MSTBIF	<p>IRC8M stabilization interrupt flag</p> <p>Set by hardware when the Internal 8 MHz RC oscillator clock is stable and the IRC8MSTBIE bit is set.</p> <p>Reset by software when setting the IRC8MSTBIC bit.</p> <p>0: No IRC8M stabilization interrupt generated</p> <p>1: IRC8M stabilization interrupt generated</p>
1	LXTALSTBIF	<p>LXTAL stabilization interrupt flag</p> <p>Set by hardware when the External 32,768 Hz crystal oscillator clock is stable and the LXTALSTBIE bit is set.</p> <p>Reset by software when setting the LXTALSTBIC bit.</p> <p>0: No LXTAL stabilization interrupt generated</p> <p>1: LXTAL stabilization interrupt generated</p>
0	IRC40KSTBIF	<p>IRC40K stabilization interrupt flag</p> <p>Set by hardware when the Internal 32kHz RC oscillator clock is stable and the IRC40KSTBIE bit is set.</p> <p>Reset by software when setting the IRC40KSTBIC bit.</p> <p>0: No IRC40K stabilization clock ready interrupt generated</p> <p>1: IRC40K stabilization interrupt generated</p>

### For GD32F170xx and GD32F190xx devices

Address offset: 0x08

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								CKMIC	Reserve d	IRC28MS TBIC	PLL STBIC	HXTAL STBIC	IRC8M STBIC	LXTAL STBIC	IRC40K STBIC
								w	w	w	w	w	w	w	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		IRC28 MSTBI E	PLL STBIE	HXTAL STBIE	IRC8M STBIE	LXTAL STBIE	IRC40K STBIE	CKMIF	Reserve d	IRC28M STBIF	PLL STBIF	HXTAL STBIF	IRC8M STBIF	LXTAL STBIF	IRC40K STBIF
		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

Bits	Fields	Descriptions
31:24	Reserved	Must be kept at reset value
23	CKMIC	<p>HXTAL Clock Stuck Interrupt Clear</p> <p>Write 1 by software to reset the CKMIF flag.</p> <p>0: Not reset CKMIF flag</p> <p>1: Reset CKMIF flag</p>

22	Reserved	Must be kept at reset value
21	IRC28MSTBIC	IRC28M stabilization Interrupt Clear Write 1 by software to reset the IRC28MSTBIF flag. 0: Not reset IRC28MSTBIF flag 1: Reset IRC28MSTBIF flag
20	PLLSTBIC	PLL stabilization Interrupt Clear Write 1 by software to reset the PLLSTBIF flag. 0: Not reset PLLSTBIF flag 1: Reset PLLSTBIF flag
19	HXTALSTBIC	HXTAL Stabilization Interrupt Clear Write 1 by software to reset the HXTALSTBIF flag. 0: Not reset HXTALSTBIF flag 1: Reset HXTALSTBIF flag
18	IRC8MSTBIC	IRC8M Stabilization Interrupt Clear Write 1 by software to reset the IRC8MSTBIF flag. 0: Not reset IRC8MSTBIF flag 1: Reset IRC8MSTBIF flag
17	LXTALSTBIC	LXTAL Stabilization Interrupt Clear Write 1 by software to reset the LXTALSTBIF flag. 0: Not reset LXTALSTBIF flag 1: Reset LXTALSTBIF flag
16	IRC40KSTBIC	IRC40K Stabilization Interrupt Clear Write 1 by software to reset the IRC40KSTBIF flag. 0: Not reset IRC40KSTBIF flag 1: Reset IRC40KSTBIF flag
15:14	Reserved	Must be kept at reset value
13	IRC28MSTBIE	IRC28M Stabilization Interrupt Enable Set and reset by software to enable/disable the IRC28M stabilization interrupt. 0: Disable the IRC28M stabilization interrupt 1: Enable the IRC28M stabilization interrupt
12	PLLSTBIE	PLL Stabilization Interrupt Enable Set and reset by software to enable/disable the PLL stabilization interrupt. 0: Disable the PLL stabilization interrupt 1: Enable the PLL stabilization interrupt
11	HXTALSTBIE	HXTAL Stabilization Interrupt Enable Set and reset by software to enable/disable the HXTAL stabilization interrupt 0: Disable the HXTAL stabilization interrupt 1: Enable the HXTAL stabilization interrupt

10	IRC8MSTBIE	<p>IRC8M Stabilization Interrupt Enable</p> <p>Set and reset by software to enable/disable the IRC8M stabilization interrupt</p> <p>0: Disable the IRC8M stabilization interrupt</p> <p>1: Enable the IRC8M stabilization interrupt</p>
9	LXTALSTBIE	<p>LXTAL Stabilization Interrupt Enable</p> <p>LXTAL stabilization interrupt enable/disable control</p> <p>0: Disable the LXTAL stabilization interrupt</p> <p>1: Enable the LXTAL stabilization interrupt</p>
8	IRC40KSTBIE	<p>IRC40K Stabilization interrupt enable</p> <p>IRC40K stabilization interrupt enable/disable control</p> <p>0: Disable the IRC40K stabilization interrupt</p> <p>1: Enable the IRC40K stabilization interrupt</p>
7	CKMIF	<p>HXTAL Clock Stuck Interrupt Flag</p> <p>Set by hardware when the HXTAL clock is stuck.</p> <p>Reset by software when setting the CKMIC bit.</p> <p>0: Clock operating normally</p> <p>1: HXTAL clock stuck</p>
6	Reserved	Must be kept at reset value
5	IRC28MSTBIF	<p>IRC28M stabilization interrupt flag</p> <p>Set by hardware when the IRC28M is stable and the IRC28MSTBIE bit is set.</p> <p>Reset by software when setting the IRC28MSTBIC bit.</p> <p>0: No IRC28M stabilization interrupt generated</p> <p>1: IRC28M stabilization interrupt generated</p>
4	PLLSTBIF	<p>PLL stabilization interrupt flag</p> <p>Set by hardware when the PLL is stable and the PLLSTBIE bit is set.</p> <p>Reset by software when setting the PLLSTBIC bit.</p> <p>0: No PLL stabilization interrupt generated</p> <p>1: PLL stabilization interrupt generated</p>
3	HXTALSTBIF	<p>HXTAL stabilization interrupt flag</p> <p>Set by hardware when the External 4 ~ 32 MHz crystal oscillator clock is stable and the HXTALSTBIE bit is set.</p> <p>Reset by software when setting the HXTALSTBIC bit.</p> <p>0: No HXTAL stabilization interrupt generated</p> <p>1: HXTAL stabilization interrupt generated</p>
2	IRC8MSTBIF	<p>IRC8M stabilization interrupt flag</p> <p>Set by hardware when the Internal 8 MHz RC oscillator clock is stable and the IRC8MSTBIE bit is set.</p> <p>Reset by software when setting the IRC8MSTBIC bit.</p> <p>0: No IRC8M stabilization interrupt generated</p>

		1: IRC8M stabilization interrupt generated
1	LXTALSTBIF	<p>LXTAL stabilization interrupt flag</p> <p>Set by hardware when the External 32.768KHz crystal oscillator clock is stable and the LXTALSTBIE bit is set.</p> <p>Reset by software when setting the LXTALSTBIC bit.</p> <p>0: No LXTAL stabilization interrupt generated</p> <p>1: LXTAL stabilization interrupt generated</p>
0	IRC40KSTBIF	<p>IRC40K stabilization interrupt flag</p> <p>Set by hardware when the Internal 40kHz RC oscillator clock is stable and the IRC40KSTBIE bit is set.</p> <p>Reset by software when setting the IRC40KSTBIC bit.</p> <p>0: No IRC40K stabilization clock ready interrupt generated</p> <p>1: IRC40K stabilization interrupt generated</p>

#### 4.3.4. APB2 reset register (RCU\_APB2RST)

Address offset: 0x0C

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													TIMER16	TIMER15	TIMER14
													RST	RST	RST
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	USART0	Reserved.	SPI0	TIMER0	Reserved.	ADC	Reserved							CFGCMP	
	RST		RST	RST		RST								RST	
	rw		rw	rw		rw								rw	

Bits	Fields	Descriptions
31:19	Reserved	Must be kept at reset value
18	TIMER16RST	TIMER16 reset This bit is set and reset by software. 0: No reset 1: Reset the TIMER16
17	TIMER15RST	TIMER15 reset This bit is set and reset by software. 0: No reset 1: Reset the TIMER15
16	TIMER14RST	TIMER14 reset

		This bit is set and reset by software. 0: No reset 1: Reset the TIMER14
14	USART0RST	USART0 Reset This bit is set and reset by software. 0: No reset 1: Reset the USART0
13	Reserved	Must be kept at reset value
12	SPI0RST	SPI0 Reset This bit is set and reset by software. 0: No reset 1: Reset the SPI0
11	TIMER0RST	TIMER0 reset This bit is set and reset by software. 0: No reset 1: Reset the TIMER0
10	Reserved	Must be kept at reset value
9	ADCRST	ADC reset This bit is set and reset by software. 0: No reset 1: Reset the ADC
8:1	Reserved	Must be kept at reset value
0	CFGCMRST	System configuration and comparator reset This bit is set and reset by software. 0: No reset 1: Reset system configuration and comparator

#### 4.3.5. APB1 reset register (RCU\_APB1RST)

##### For GD32F130xx and GD32F150xx devices

Address offset: 0x10

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	CEC	DAC	PMU	Reserved				USB	I2C1	I2C0	Reserved			USART	Reserved
d	RST	RST	RST					RST	RST	RST				1RST	d



16	Reserved	Must be kept at reset value
15	SPI2RST	SPI2 reset This bit is set and reset by software. 0: No reset 1: Reset SPI2
14	SPI1RST	SPI1 reset This bit is set and reset by software. 0: No reset 1: Reset SPI1
13:12	Reserved	Must be kept at reset value
11	WWDGTRST	Window watchdog timer reset This bit is set and reset by software. 0: No reset 1: Reset window watchdog timer
10:9	Reserved	Must be kept at reset value
8	TIMER13RST	TIMER13 timer reset This bit is set and reset by software. 0: No reset 1: Reset TIMER13 Timer
7:5	Reserved	Must be kept at reset value
4	TIMER5RST	TIMER5 timer reset This bit is set and reset by software. 0: No reset 1: Reset TIMER5 Timer
3:2	Reserved	Must be kept at reset value
1	TIMER2RST	TIMER2 timer reset This bit is set and reset by software. 0: No reset 1: Reset TIMER2 timer
0	TIMER1RST	TIMER1 timer reset This bit is set and reset by software. 0: No reset 1: Reset TIMER1 timer

### For GD32F170xx and GD32F190xx devices

Address offset: 0x10

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OPAIVR EFRST	CEC RST	DAC RST	PMU RST	Reserved d	CAN1 RST	CAN0 RST	Reserved		I2C1 RST	I2C0 RST	Reserved			USART 1RST	Reserved d
rw		rw		rw		rw		rw		rw		rw		rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI2 RST	SPI1 RST	Reserved		WWDG TRST	Reserved d	SLCDR ST	TIMER1 3RST	Reserved			TIMER5 RST	Reserved		TIMER2 RST	TIMER1 RST
rw		rw		rw		rw		rw		rw		rw		rw	

Bits	Fields	Descriptions
31	OPAIVREFRST	OPA and IVREF reset This bit is set and reset by software. 0: No reset 1: Reset OPA unit
30	CECRST	HDMI CEC reset This bit is set and reset by software. 0: No reset 1: Reset hdmi cec unit
29	DACRST	DAC reset This bit is set and reset by software. 0: No reset 1: Reset DAC unit
28	PMURST	Power control reset This bit is set and reset by software. 0: No reset 1: Reset power control unit
27	Reserved	Must be kept at reset value
26	CAN1RST	CAN1 reset This bit is set and reset by software. 0: No reset 1: Reset CAN1
25	CAN0RST	CAN0 reset This bit is set and reset by software. 0: No reset 1: Reset CAN0
24:23	Reserved	Must be kept at reset value
22	I2C1RST	I2C1 reset This bit is set and reset by software.

		0: No reset 1: Reset I2C1
21	I2C0RST	I2C0 reset This bit is set and reset by software. 0: No reset 1: Reset I2C0
20:18	Reserved	Must be kept at reset value
17	USART1RST	USART1 reset This bit is set and reset by software. 0: No reset 1: Reset USART1
16	Reserved	Must be kept at reset value
15	SPI2RST	SPI2 reset This bit is set and reset by software. 0: No reset 1: Reset SPI2
14	SPI1RST	SPI1 reset This bit is set and reset by software. 0: No reset 1: Reset SPI1
13:12	Reserved	Must be kept at reset value
11	WWDGTRST	Window watchdog timer reset This bit is set and reset by software. 0: No reset 1: Reset window watchdog timer
10	Reserved	Must be kept at reset value
9	SLCDRST	SLCD reset This bit is set and reset by software. 0: No reset 1: Reset SLCD Timer
8	TIMER13RST	TIMER13 timer reset This bit is set and reset by software. 0: No reset 1: Reset TIMER13 timer
7:5	Reserved	Must be kept at reset value
4	TIMER5RST	TIMER5 timer reset This bit is set and reset by software.

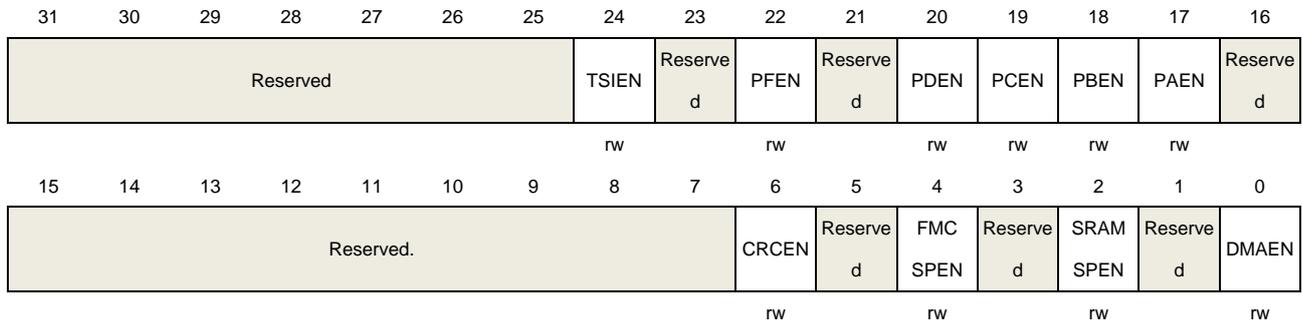
		0: No reset 1: Reset TIMER5 Timer
3:2	Reserved	Must be kept at reset value
1	TIMER2RST	TIMER2 timer reset This bit is set and reset by software. 0: No reset 1: Reset TIMER2 timer
0	TIMER1RST	TIMER1 timer reset This bit is set and reset by software. 0: No reset 1: Reset TIMER1 timer

### 4.3.6. AHB enable register (RCU\_AHBEN)

Address offset: 0x14

Reset value: 0x0000 0014

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



Bits	Fields	Descriptions
31:25	Reserved	Must be kept at reset value
24	TSIEN	TSI clock enable This bit is set and reset by software. 0: Disabled TSI clock 1: Enabled TSI clock
23	Reserved	Must be kept at reset value
22	PFEN	GPIO port F clock enable This bit is set and reset by software. 0: Disabled GPIO port F clock 1: Enabled GPIO port F clock

21	Reserved	Must be kept at reset value
20	PDEN	GPIO port D clock enable This bit is set and reset by software. 0: Disabled GPIO port D clock 1: Enabled GPIO port D clock
19	PCEN	GPIO port C clock enable This bit is set and reset by software. 0: Disabled GPIO port C clock 1: Enabled GPIO port C clock
18	PBEN	GPIO port B clock enable This bit is set and reset by software. 0: Disabled GPIO port B clock 1: Enabled GPIO port B clock
17	PAEN	GPIO port A clock enable This bit is set and reset by software. 0: Disabled GPIO port A clock 1: Enabled GPIO port A clock
16:7	Reserved	Must be kept at reset value
6	CRCEN	CRC clock enable This bit is set and reset by software. 0: Disabled CRC clock 1: Enabled CRC clock
5	Reserved	Must be kept at reset value
4	FMCSPEEN	FMC clock enable when sleep mode This bit is set and reset by software to enable/disable FMC clock during Sleep mode. 0: Disabled FMC clock during Sleep mode 1: Enabled FMC clock during Sleep mode
3	Reserved	Must be kept at reset value
2	SRAMSPEN	SRAM interface clock enable when sleep mode This bit is set and reset by software to enable/disable SRAM interface clock during Sleep mode. 0: Disabled SRAM interface clock during Sleep mode. 1: Enabled SRAM interface clock during Sleep mode
1	Reserved	Must be kept at reset value
0	DMAEN	DMA clock enable This bit is set and reset by software. 0: Disabled DMA clock

1: Enabled DMA clock

### 4.3.7. APB2 enable register (RCU\_APB2EN)

Address offset: 0x18

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												TIMER1 6EN	TIMER1 5EN	TIMER1 4EN	
												rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve d	USART 0EN	Reserve d	SPI0EN	TIMER0 EN	Reserve d	ADCEN	Reserved						CFGCM PEN		
	rw		rw	rw		rw							rw		

Bits	Fields	Descriptions
31:19	Reserved	Must be kept at reset value
18	TIMER16EN	TIMER16 timer clock enable This bit is set and reset by software. 0: Disabled TIMER16 timer clock 1: Enabled TIMER16 timer clock
17	TIMER15EN	TIMER15 timer clock enable This bit is set and reset by software. 0: Disabled TIMER15 timer clock 1: Enabled TIMER15 timer clock
16	TIMER14EN	TIMER14 timer clock enable This bit is set and reset by software. 0: Disabled TIMER14 timer clock 1: Enabled TIMER14 timer clock
15	Reserved	Must be kept at reset value
14	USART0EN	USART0 clock enable This bit is set and reset by software. 0: Disabled USART0 clock 1: Enabled USART0 clock
13	Reserved	Must be kept at reset value
12	SPI0EN	SPI0 clock enable This bit is set and reset by software.

		0: Disabled SPI0 clock 1: Enabled SPI0 clock
11	TIMER0EN	TIMER0 timer clock enable This bit is set and reset by software. 0: Disabled TIMER0 timer clock 1: Enabled TIMER0 timer clock
10	Reserved	Must be kept at reset value
9	ADCEN	ADC interface clock enable This bit is set and reset by software. 0: Disabled ADC interface clock 1: Enabled ADC interface clock
8:1	Reserved	Must be kept at reset value
0	CFGCOMPEN	System configuration and comparator clock enable This bit is set and reset by software. 0: Disabled System configuration and comparator clock 1: Enabled System configuration and comparator clock

### 4.3.8. APB1 enable register (RCU\_APB1EN)

#### For GD32F130xx and GD32F150xx devices

Address offset: 0x1C

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	CECEN	DACEN	PMUEN	Reserved				USBDE N	I2C1EN	I2C0EN	Reserved			USART 1EN	Reserved
	rw	rw	rw					rw	rw	rw				rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI2EN	SPI1EN	Reserved		WWDG TEN	Reserved		TIMER1 3EN	Reserved			TIMER5 EN	Reserved		TIMER2 EN	TIMER1 EN
rw	rw			rw			rw				rw			rw	rw

Bits	Fields	Descriptions
31	Reserved	Must be kept at reset value
30	CECEN	HDMI CEC interface clock enable This bit is set and reset by software. 0: Disabled HDMI CEC interface clock

		1: Enabled HDMI CEC interface clock
29	DACEN	DAC interface clock enable This bit is set and reset by software. 0: Disabled DAC interface clock 1: Enabled DAC interface clock
28	PMUEN	Power interface clock enable This bit is set and reset by software. 0: Disabled Power interface clock 1: Enabled Power interface clock
27:24	Reserved	Must be kept at reset value
23	USBDEN	USB D clock enable This bit is set and reset by software. 0: Disabled USB D clock 1: Enabled USB D clock
22	I2C1EN	I2C1 clock enable This bit is set and reset by software. 0: Disabled I2C1 clock 1: Enabled I2C1 clock
21	I2C0EN	I2C0 clock enable This bit is set and reset by software. 0: Disabled I2C0 clock 1: Enabled I2C0 clock
20:18	Reserved	Must be kept at reset value
17	USART1EN	USART1 clock enable This bit is set and reset by software. 0: Disabled USART1 clock 1: Enabled USART1 clock
16	Reserved	Must be kept at reset value
15	SPI2EN	SPI2 clock enable This bit is set and reset by software. 0: Disabled SPI2 clock 1: Enabled SPI2 clock
14	SPI1EN	SPI1 clock enable This bit is set and reset by software. 0: Disabled SPI1 clock 1: Enabled SPI1 clock
13:12	Reserved	Must be kept at reset value

11	WWDGTEN	Window watchdog timer clock enable This bit is set and reset by software. 0: Disabled Window watchdog timer clock 1: Enabled Window watchdog timer clock
10:9	Reserved	Must be kept at reset value
8	TIMER13EN	TIMER13 timer clock enable This bit is set and reset by software. 0: Disabled TIMER13 timer clock 1: Enabled TIMER13 timer clock
7:5	Reserved	Must be kept at reset value
4	TIMER5EN	TIMER5 timer clock enable This bit is set and reset by software. 0: Disabled TIMER5 timer clock 1: Enabled TIMER5 timer clock
3:2	Reserved	Must be kept at reset value
1	TIMER2EN	TIMER2 timer clock enable This bit is set and reset by software. 0: Disabled TIMER2 timer clock 1: Enabled TIMER2 timer clock
0	TIMER1EN	TIMER1 timer clock enable This bit is set and reset by software. 0: Disabled TIMER1 timer clock 1: Enabled TIMER1 timer clock

### For GD32F170xx and GD32F190xx devices

Address offset: 0x1C

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OPAIVR EFEN	CECEN	DACEN	PMUEN	Reserve d	CAN1E N	CAN0E N	Reserved	I2C1EN	I2C0EN	Reserved	Reserved	Reserved	Reserved	USART 1EN	Reserve d
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
SPI2EN	SPI1EN	Reserved	Reserved	WWDG TEN	Reserve d	SLCDE N	TIMER1 3EN	Reserved	Reserved	TIMER5 EN	Reserved	Reserved	Reserved	TIMER2 EN	TIMER1 EN
rw	rw			rw		rw	rw			rw				rw	rw

**Bits      Fields      Descriptions**

31	OPAIVREFEN	OPA and IVREF clock enable This bit is set and reset by software. 0: Disabled OPA and IVREF interface clock 1: Enabled OPA and IVREF interface clock
30	CECEN	HDMI CEC interface clock enable This bit is set and reset by software. 0: Disabled HDMI CEC interface clock 1: Enabled HDMI CEC interface clock
29	DACEN	DAC interface clock enable This bit is set and reset by software. 0: Disabled DAC interface clock 1: Enabled DAC interface clock
28	PMUEN	Power interface clock enable This bit is set and reset by software. 0: Disabled Power interface clock 1: Enabled Power interface clock
27	Reserved	Must be kept at reset value
26	CAN1EN	CAN1 clock enable This bit is set and reset by software. 0: Disabled CAN1 clock 1: Enabled CAN1 clock
25	CAN0EN	CAN0 clock enable This bit is set and reset by software. 0: Disabled CAN0 clock 1: Enabled CAN0 clock
24:23	Reserved	Must be kept at reset value
22	I2C1EN	I2C1 clock enable This bit is set and reset by software. 0: Disabled I2C1 clock 1: Enabled I2C1 clock
21	I2C0EN	I2C0 clock enable This bit is set and reset by software. 0: Disabled I2C0 clock 1: Enabled I2C0 clock
20:18	Reserved	Must be kept at reset value
17	USART1EN	USART1 clock enable This bit is set and reset by software. 0: Disabled USART1 clock

		1: Enabled USART1 clock
16	Reserved	Must be kept at reset value
15	SPI2EN	SPI2 clock enable This bit is set and reset by software. 0: Disabled SPI2 clock 1: Enabled SPI2 clock
14	SPI1EN	SPI1 clock enable This bit is set and reset by software. 0: Disabled SPI1 clock 1: Enabled SPI1 clock
13:12	Reserved	Must be kept at reset value
11	WWDGTEN	Window watchdog timer clock enable This bit is set and reset by software. 0: Disabled Window watchdog timer clock 1: Enabled Window watchdog timer clock
10	Reserved	Must be kept at reset value
9	SLCDEN	SLCD clock enable This bit is set and reset by software. 0: Disabled SLCD clock 1: Enabled SLCD clock
8	TIMER13EN	TIMER13 timer clock enable This bit is set and reset by software. 0: Disabled TIMER13 timer clock 1: Enabled TIMER13 timer clock
7:5	Reserved	Must be kept at reset value
4	TIMER5EN	TIMER5 timer clock enable This bit is set and reset by software. 0: Disabled TIMER5 timer clock 1: Enabled TIMER5 timer clock
3:2	Reserved	Must be kept at reset value
1	TIMER2EN	TIMER2 timer clock enable This bit is set and reset by software. 0: Disabled TIMER2 timer clock 1: Enabled TIMER2 timer clock
0	TIMER1EN	TIMER1 timer clock enable This bit is set and reset by software. 0: Disabled TIMER1 timer clock

1: Enabled TIMER1 timer clock

## 4.3.9. Backup domain control register (RCU\_BDCTL)

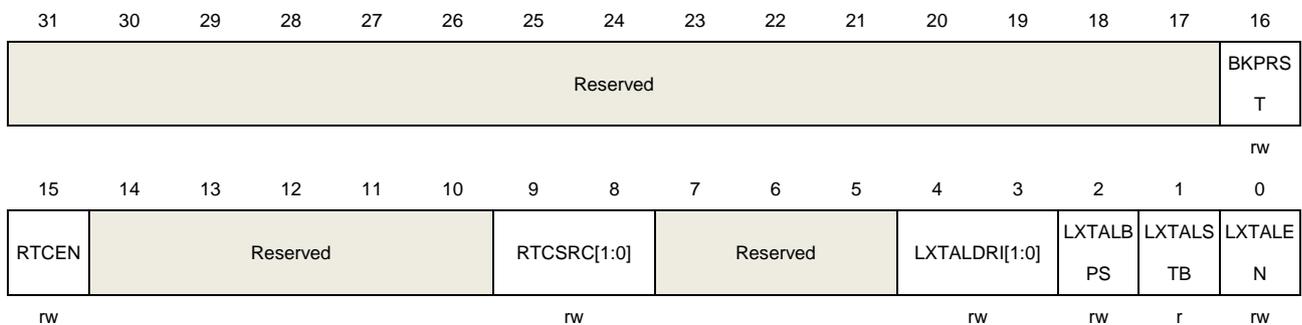
### For GD32F130xx and GD32F150xx devices

Address offset: 0x20

Reset value: 0x0000 0018, reset by Backup domain Reset.

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

**Note:** The LXTALEN, LXTALBPS, RTCSRC and RTCEN bits of the Backup domain control register (BDCTL) are only reset after a Backup domain Reset. These bits can be modified only when the BKPWEN bit in the Power control register (PMU\_CTL) has to be set.



Bits	Fields	Descriptions
31:17	Reserved	Must be kept at reset value
16	BKPRST	Backup domain reset This bit is set and reset by software. 0: No reset 1: Resets Backup domain
15	RTCEN	RTC clock enable This bit is set and reset by software. 0: Disabled RTC clock 1: Enabled RTC clock
14:10	Reserved	Must be kept at reset value
9:8	RTCSRC[1:0]	RTC clock entry selection Set and reset by software to control the RTC clock source. 00: No clock selected 01: CK_LXTAL selected as RTC source clock 10: CK_IRC40K selected as RTC source clock 11: (CK_HXTAL / 32) selected as RTC source clock

7:5	Reserved	Must be kept at reset value
4:3	LXTALDRI[1:0]	<p>LXTAL drive capability</p> <p>Set and reset by software. Backup domain reset reset this value.</p> <p>00: lower driving capability</p> <p>01: medium low driving capability</p> <p>10: medium high driving capability</p> <p>11: higher driving capability (reset value)</p> <p><b>Note:</b> The LXTALDRI is not in bypass mode.</p>
2	LXTALBPS	<p>LXTAL bypass mode enable</p> <p>Set and reset by software.</p> <p>0: Disable the LXTAL Bypass mode</p> <p>1: Enable the LXTAL Bypass mode</p>
1	LXTALSTB	<p>External low-speed oscillator stabilization</p> <p>Set by hardware to indicate if the LXTAL output clock is stable and ready for use.</p> <p>0: LXTAL is not stable</p> <p>1: LXTAL is stable</p>
0	LXTALEN	<p>LXTAL enable</p> <p>Set and reset by software.</p> <p>0: Disable LXTAL</p> <p>1: Enable LXTAL</p>

### For GD32F170xx and GD32F190xx devices

Address offset: 0x20

Reset value: 0x0000 0018, reset by Backup domain Reset.

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

**Note:** The LXTALEN, LXTALBPS, RTCSRC and RTCEN bits of the Backup domain control register (BDCTL) are only reset after a Backup domain Reset. These bits can be modified only when the BKPWEN bit in the Power control register (PMU\_CTL) has to be set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															BKPRS
															T
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTCEN	Reserved					RTCSRC[1:0]		Reserved				LXTALDRI[1:0]	LXTALB	LXTALS	LXTALE
												PS	TB	N	
rw						rw						rw	r	rw	

Bits	Fields	Descriptions
------	--------	--------------

31:17	Reserved	Must be kept at reset value
16	BKPRST	Backup domain reset This bit is set and reset by software. 0: No reset 1: Resets Backup domain
15	RTCEN	RTC clock enable This bit is set and reset by software. 0: Disabled RTC clock 1: Enabled RTC clock
14:10	Reserved	Must be kept at reset value
9:8	RTCSRC[1:0]	RTC and SLCD clock entry selection Set and reset by software to control the RTC and SLCD clock source. Once the RTC and SLCD clock source has been selected, it cannot be changed anymore unless the Backup domain is reset. 00: No clock selected 01: CK_LXTAL selected as RTC/SLCD source clock 10: CK_IRC40K selected as RTC/SLCD source clock 11: (CK_HXTAL / 32) selected as RTC/SLCD source clock
7:5	Reserved	Must be kept at reset value
4:3	LXTALDRI[1:0]	LXTAL drive capability Set and reset by software. Backup domain reset reset this value. 00: lower driving capability 01: medium low driving capability 10: medium high driving capability 11: higher driving capability (reset value) <b>Note:</b> The LXTALDRI is not in bypass mode.
2	LXTALBPS	LXTAL bypass mode enable Set and reset by software. 0: Disable the LXTAL Bypass mode 1: Enable the LXTAL Bypass mode
1	LXTALSTB	External low-speed oscillator stabilization Set by hardware to indicate if the LXTAL output clock is stable and ready for use. 0: LXTAL is not stable 1: LXTAL is stable
0	LXTALEN	LXTAL enable Set and reset by software. 0: Disable LXTAL 1: Enable LXTAL

#### 4.3.10. Reset source /clock register (RCU\_RSTSCK)

Address offset: 0x24

Reset value: 0x0C00 0000, reset flags reset by power Reset only, other reset by system reset.

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LP RSTF	WWDG TRSTF	FWDGT RSTF	SW RSTF	POR RSTF	EP RSTF	OBL RSTF	RSTFC	V12 RSTF	Reserved						
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													IRC40K STB	IRC40K EN	
													r	rw	

Bits	Fields	Descriptions
31	LPRSTF	Low-power reset flag Set by hardware when Deep-sleep /standby reset generated. Reset by writing 1 to the RSTFC bit. 0: No Low-power management reset generated 1: Low-power management reset generated
30	WWDGTRSTF	Window watchdog timer reset flag Set by hardware when a window watchdog timer reset generated. Reset by writing 1 to the RSTFC bit. 0: No window watchdog reset generated 1: Window watchdog reset generated
29	FWDGTRSTF	Free Watchdog timer reset flag Set by hardware when a Free Watchdog timer generated. Reset by writing 1 to the RSTFC bit. 0: No Free Watchdog timer reset generated 1: Free Watchdog timer reset generated
28	SWRSTF	Software reset flag Set by hardware when a software reset generated. Reset by writing 1 to the RSTFC bit. 0: No software reset generated 1: Software reset generated
27	PORRSTF	Power reset flag Set by hardware when a Power reset generated. Reset by writing 1 to the RSTFC bit. 0: No Power reset generated

		1: Power reset generated
26	EPRSTF	External PIN reset flag Set by hardware when an External PIN generated. Reset by writing 1 to the RSTFC bit. 0: No External PIN reset generated 1: External PIN reset generated
25	OBLRSTF	Option byte loader reset flag Set by hardware when an option byte loader generated. Reset by writing 1 to the RSTFC bit. 0: No Option byte loader reset generated 1: Option byte loader reset generated
24	RSTFC	Reset flag clear This bit is set by software to clear all reset flags. 0: Not clear reset flags 1: Clear reset flags
23	V12RSTF	V12 domain Power reset flag Set by hardware when a V12 domain Power reset generated. Reset by writing 1 to the RSTFC bit. 0: No V12 domain Power reset generated 1: V12 domain Power reset generated
22:2	Reserved	Must be kept at reset value
1	IRC40KSTB	IRC40K stabilization Set by hardware to indicate if the IRC40K output clock is stable and ready for use. 0: IRC40K is not stable 1: IRC40K is stable
0	IRC40KEN	IRC40K enable Set and reset by software. 0: Disable IRC40K 1: Enable IRC40K

### 4.3.11. AHB reset register (RCU\_AHBRST)

Address offset: 0x28

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved							TSIRST	Reserved	PFRST	Reserved	PDRST	PCRST	PBRST	PARST	Reserved
							rw		rw		rw	rw	rw	rw	

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

Reserved.

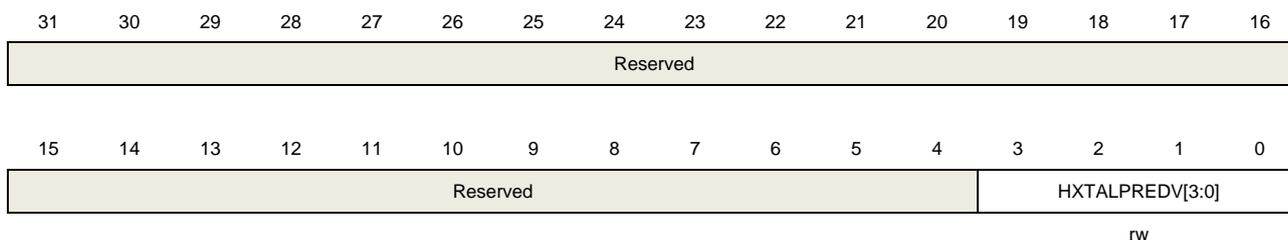
Bits	Fields	Descriptions
31:25	Reserved	Must be kept at reset value
24	TSIRST	TSI unit reset This bit is set and reset by software. 0: No reset TSI unit 1: Reset TSI unit
23	Reserved	Must be kept at reset value
22	PFRST	GPIO port F reset This bit is set and reset by software. 0: No reset GPIO port F 1: Reset GPIO port F
21	Reserved	Must be kept at reset value
20	PDRST	GPIO port D reset This bit is set and reset by software. 0: No reset GPIO port D 1: Reset GPIO port D
19	PCRST	GPIO port C reset This bit is set and reset by software. 0: No reset GPIO port C 1: Reset GPIO port C
18	PBRST	GPIO port B reset This bit is set and reset by software. 0: No reset GPIO port B 1: Reset GPIO port B
17	PARST	GPIO port A reset This bit is set and reset by software. 0: No reset GPIO port A 1: Reset GPIO port A
16:0	Reserved	Must be kept at reset value

#### 4.3.12. Configuration register 1 (RCU\_CFG1)

Address offset: 0x2c

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



Bits	Fields	Descriptions
31:4	Reserved	Must be kept at reset value
3:0	HXTALPREDV[3:0]	CK_HXTAL divider previous PLL This bit is set and reset by software. These bits can be written when PLL is disable <b>Note:</b> The bit 0 of HXTALPREDV is same as bit 17 of RCU_CFG0, so modifying bit 17 of RCU_CFG0 aslo modifies bit 0 of RCU_CFG2. The CK_HXTAL is divided by (HXTALPREDV + 1). 0000: HXTAL input to PLL not divided 0001: HXTAL input to PLL divided by 2 0010: HXTAL input to PLL divided by 3 0011: HXTAL input to PLL divided by 4 0100: HXTAL input to PLL divided by 5 0101: HXTAL input to PLL divided by 6 0110: HXTAL input to PLL divided by 7 0111: HXTAL input to PLL divided by 8 1000: HXTAL input to PLL divided by 9 1001: HXTAL input to PLL divided by 10 1010: HXTAL input to PLL divided by 11 1011: HXTAL input to PLL divided by 12 1100: HXTAL input to PLL divided by 13 1101: HXTAL input to PLL divided by 14 1110: HXTAL input to PLL divided by 15 1111: HXTAL input to PLL divided by 16

### 4.3.13. Configuration register 2 (RCU\_CFG2)

#### For GD32F130xx and GD32F150xx devices

Address offset: 0x30

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							ADCSE	Reserve	CECSE	Reserved				USART0SEL[1:0]	
							L	d	L						
							rw		rw					rw	

Bits	Fields	Descriptions
31:9	Reserved	Must be kept at reset value
8	ADCSEL	CK_ADC clock source selection This bit is set and reset by software. 0: CK_ADC select CK_IRC14M 1: CK_ADC select CK_APB2 which is divided by 2/4/6/8.
7	Reserved	Must be kept at reset value
6	CECSEL	CK_CEC clock source selection This bit is set and reset by software. 0: CK_CEC select CK_IRC8M divided by 244 1: CK_CEC select CK_LXTAL
5:2	Reserved	Must be kept at reset value
1:0	USART0SEL[1:0]	CK_USART0 clock source selection This bit is set and reset by software. 00: CK_USART0 select CK_APB2 01: CK_USART0 select CK_SYS 10: CK_USART0 select CK_LXTAL 11: CK_USART0 select CK_IRC8M

### For GD32F170xx and GD32F190xx devices

Address offset: 0x30

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															IRC28M
															DIV
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							ADCSE	Reserve	CECSE	Reserved				USART0SEL[1:0]	
							L	d	L						
							rw		rw					rw	

Bits	Fields	Descriptions
------	--------	--------------

31:17	Reserved	Must be kept at reset value
16	IRC28MDIV	CK_IRC28M divider 2 or not This bit is set and reset by software. 0: IRC28M/2 select to ADC clock 1: IRC28M select to ADC clock.
15:9	Reserved	Must be kept at reset value
8	ADCSEL	CK_ADC clock source selection This bit is set and reset by software. 0: CK_ADC select CK_IRC28M or CK_IRC28M/2 set by IRC28MDIV 1: CK_ADC select CK_APB2 which is divided by 2/4/6/8.
7	Reserved	Must be kept at reset value
6	CECSEL	CK_CEC clock source selection This bit is set and reset by software. 0: CK_CEC select CK_IRC8M divided by 244 1: CK_CEC select CK_LXTAL
5:2	Reserved	Must be kept at reset value
1:0	USART0SEL[1:0]	CK_USART0 clock source selection This bit is set and reset by software. 00: CK_USART0 select CK_APB2 01: CK_USART0 select CK_SYS 10: CK_USART0 select CK_LXTAL 11: CK_USART0 select CK_IRC8M

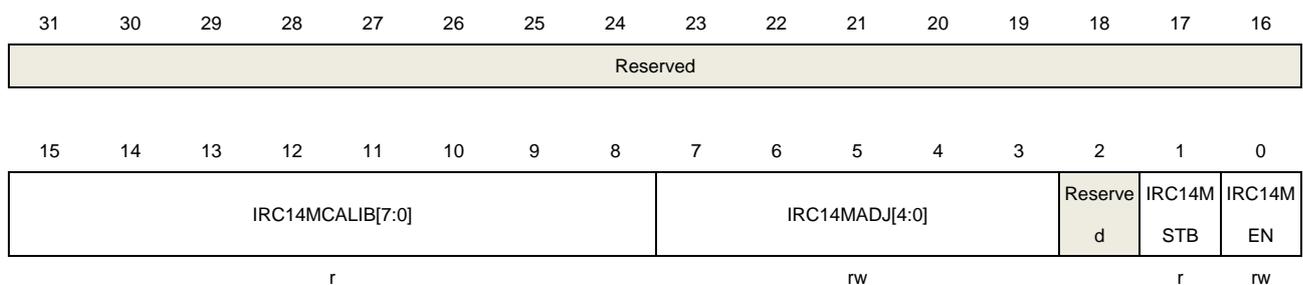
#### 4.3.14. Control register 1 (RCU\_CTL1)

##### For GD32F130xx and GD32F150xx devices

Address offset: 0x34

Reset value: 0x0000 XX80 where X is undefined.

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



Bits	Fields	Descriptions
------	--------	--------------

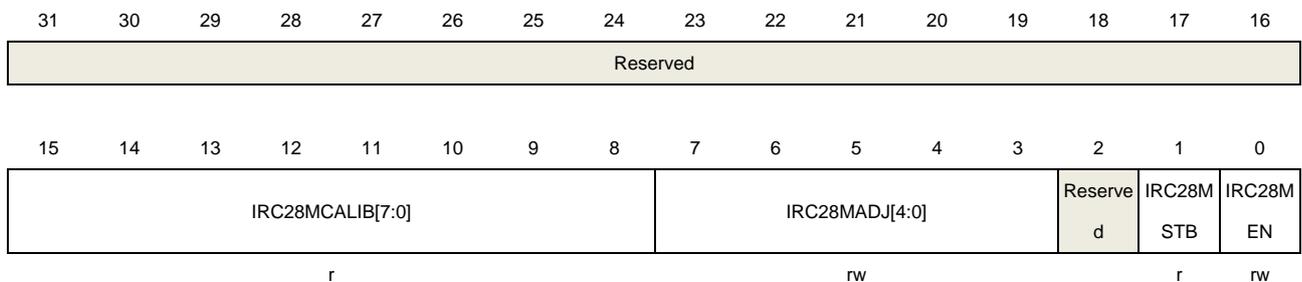
31:16	Reserved	Must be kept at reset value
15:8	IRC14MCALIB[7:0]	Internal 14M RC Oscillator calibration value register These bits are load automatically at power on.
7:3	IRC14MADJ[4:0]	Internal 14M RC Oscillator clock trim adjust value These bits are set by software. The trimming value is there bits (IRC14MADJ) added to the IRC14MCALIB[7:0] bits. The trimming value should trim the IRC14M to 14 MHz $\pm$ 1%.
2	Reserved	Must be kept at reset value
1	IRC14MSTB	IRC14M Internal 14M RC Oscillator stabilization Flag Set by hardware to indicate if the IRC14M oscillator is stable and ready for use. 0: IRC14M oscillator is not stable 1: IRC14M oscillator is stable
0	IRC14MEN	IRC14M Internal 14M RC oscillator Enable Set and reset by software. 0: Internal 14 MHz RC oscillator disabled 1: Internal 14 MHz RC oscillator enabled

### For GD32F170xx and GD32F190xx devices

Address offset: 0x34

Reset value: 0x0000 XX80 where X is undefined.

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:8	IRC28MCALIB[7:0]	Internal 28MHz RC Oscillator calibration value register These bits are load automatically at power on.
7:3	IRC28MADJ[4:0]	Internal 28MHz RC Oscillator clock trim adjust value These bits are set by software. The trimming value is there bits (IRC28MADJ) added to the IRC28MCALIB[7:0] bits. The trimming value should trim the IRC28M to 28 MHz $\pm$ 1%.
2	Reserved	Must be kept at reset value.
1	IRC28MSTB	IRC28M Internal 28M RC Oscillator stabilization Flag

Set by hardware to indicate if the IRC28M oscillator is stable and ready for use.

0: IRC28M oscillator is not stable

1: IRC28M oscillator is stable

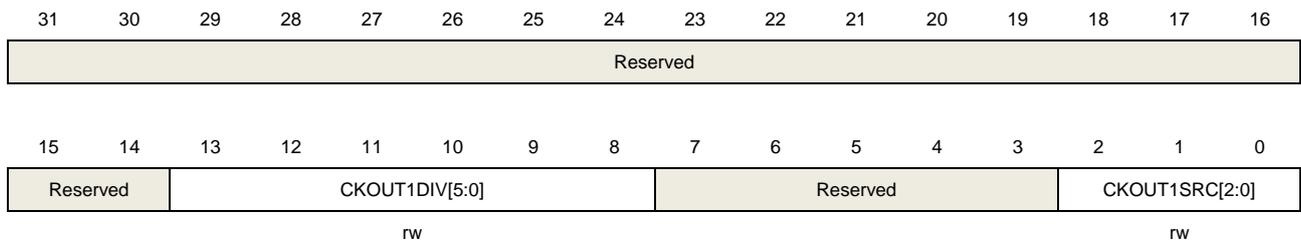
0	IRC28MEN	IRC28M Internal 28M RC oscillator Enable Set and reset by software. 0: Internal 28MHz RC oscillator disabled 1: Internal 28 MHz RC oscillator enabled
---	----------	--

### 4.3.15. Configuration register 3 (RCU\_CFG3) of GD32F170xx and GD32F190xx devices

Address offset: 0x80

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



Bits	Fields	Descriptions
31:14	Reserved	Must be kept at reset value
13:8	CKOUT1DIV[5:0]	The CK_OUT1 divider which the CK_OUT1 frequency can be reduced see bits 2:0 of RCU_CFG3 for CK_OUT1 0: The CK_OUT1 is divided by 1 1: The CK_OUT1 is divided by 2 2: The CK_OUT1 is divided by 3 ... 63: The CK_OUT1 is divided by 64
7:3	Reserved	Must be kept at reset value
2:0	CKOUT1SRC[2:0]	CKOUT1 Clock Source Selection Set and reset by software. 000: No clock selected 001: Internal 28 MHz RC Oscillator clock selected 010: Internal 40KHz RC Oscillator clock selected 011: Low Speed Crystal Oscillator clock selected 100: System clock selected 101: Internal 8 MHz RC Oscillator clock selected 110: High Speed Crystal Oscillator clock selected

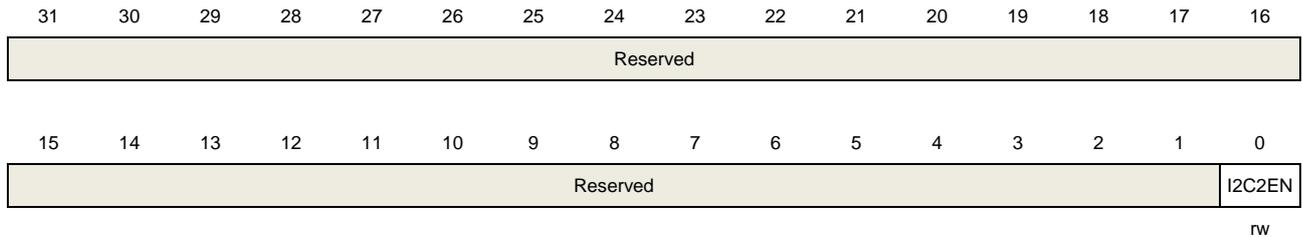
111: (CK\_PLL / 2) or CK\_PLL selected depend on PLLDV

### 4.3.16. APB1 additional enable register (RCU\_ADDAPB1EN)

Address offset: 0xF8

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



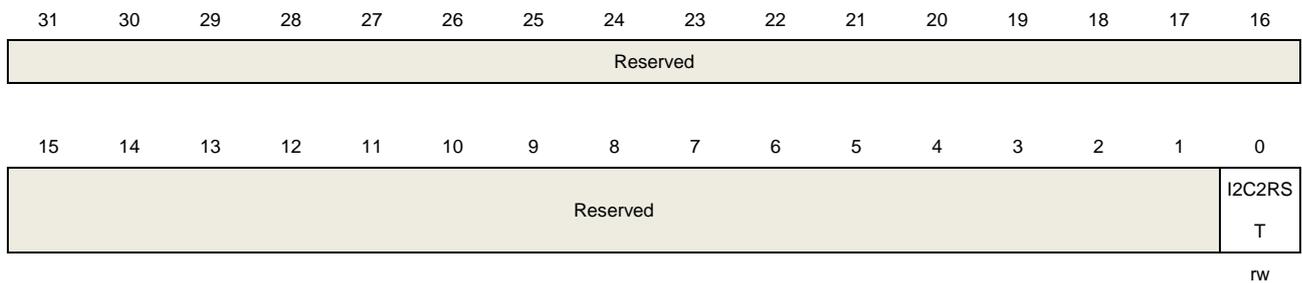
Bits	Fields	Descriptions
31:1	Reserved	Must be kept at reset value
0	I2C2EN	I2C2 unit clock enable This bit is set and reset by software 0: Disable I2C2 unit clock 1: Enable I2C2 unit clock

### 4.3.17. APB1 additional reset register (RCU\_ADDAPB1RST)

Address offset: 0xFC

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



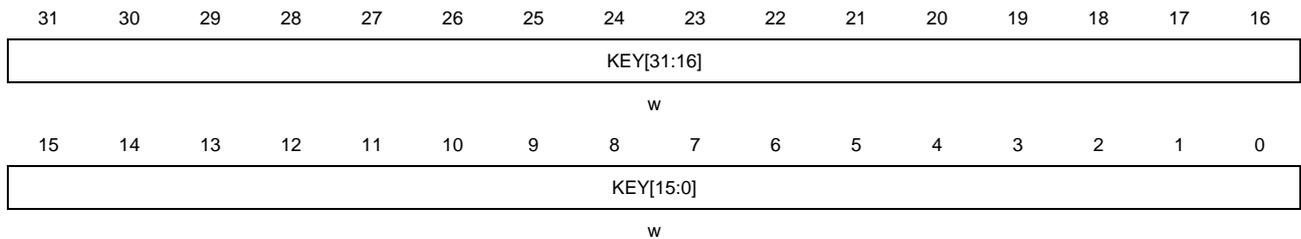
Bits	Fields	Descriptions
31:1	Reserved	Must be kept at reset value
0	I2C2RST	I2C2 unit reset This bit is set and reset by software 0: Not reset I2C2 unit

### 4.3.18. Voltage key register (RCU\_VKEY)

Address offset: 0x100

Reset value: 0x0000 0000.

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



Bits	Fields	Descriptions
31:0	KEY[31:0]	<p>The key of RCU_PDVSSEL and RCU_DSV register</p> <p>These bits are written only by software and read as 0. Only after write 0x1A2B3C4D to the RCU_VKEY, the RCU_PDVSSEL and RCU_DSV register can be written.</p>

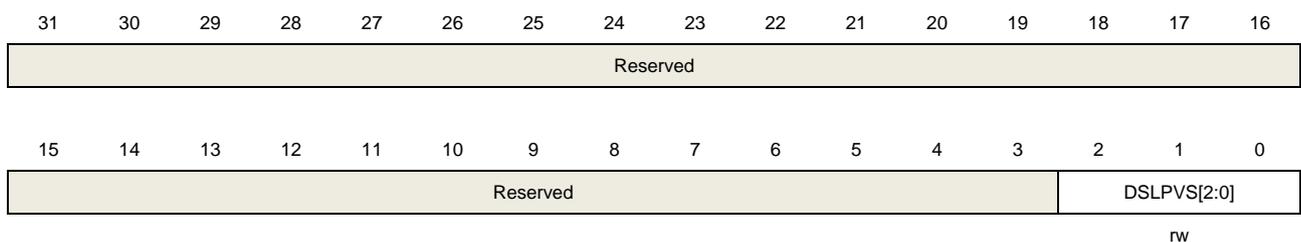
### 4.3.19. Deep-sleep mode voltage register (RCU\_DSV)

**For GD32F130xx and GD32F150xx devices**

Offset: 0x134

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



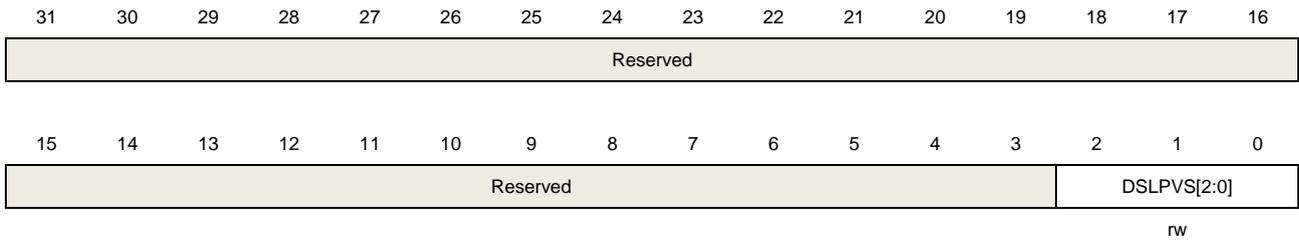
Bits	Fields	Descriptions
31:3	Reserved	Must be kept at reset value
2:0	DSL PVS[2:0]	<p>Deep-sleep mode voltage select</p> <p>These bits is set and reset by software</p> <p>000 : The core voltage is 1.2V in Deep-sleep mode</p> <p>001 : The core voltage is 1.1V in Deep-sleep mode</p> <p>010 : The core voltage is 1.0V in Deep-sleep mode</p>

011 : The core voltage is 0.9V in Deep-sleep mode  
 100~111 : Reserved

### For GD32F170xx and GD32F190xx devices

Address offset: 0x134  
 Reset value: 0x0000 0000.

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

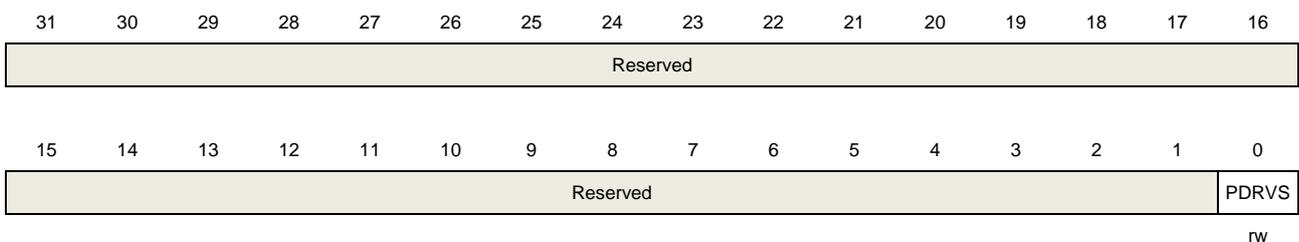


Bits	Fields	Descriptions
31:3	Reserved	Must be kept at reset value
2:0	DSL PVS[2:0]	Deep-sleep mode voltage select These bits is set and reset by software 000 : The core voltage is 1.8V in Deep-sleep mode 001 : The core voltage is 1.6V in Deep-sleep mode 010 : The core voltage is 1.4V in Deep-sleep mode 011 : The core voltage is 1.2V in Deep-sleep mode 100~111 : Reserved

### 4.3.20. Power down voltage select register (RCU\_PDVSEL) of GD32F130xx and GD32F150xx devices

Address offset: 0x138  
 Reset value: 0x0000 0000.

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



Bits	Fields	Descriptions
31:1	Reserved	Must be kept at reset value

0	PDRVS	Power down voltage select This bit is set and reset by software 0: The Power down voltage is 2.6V 1: The Power down voltage is 1.8V
---	-------	--

## 5. Interrupt/event controller (EXTI)

### 5.1. Overview

Cortex-M3 integrates the Nested Vectored Interrupt Controller (NVIC) for efficient exception and interrupts processing. NVIC facilitates low-latency exception and interrupt handling and controls power management. It's tightly coupled to the processor core. You can read the Technical Reference Manual of Cortex-M3 for more details about NVIC.

EXTI (interrupt/event controller) contains up to 23 independent edge detectors and generates interrupt requests or events to the processor. The EXTI has three trigger types: rising edge, falling edge and both edges. Each edge detector in the EXTI can be configured and masked independently.

### 5.2. Characteristics

- Cortex-M3 system exception.
- Up to 52 maskable peripheral interrupts for GD32F130xx and GD32F150xx devices or 74 maskable peripheral interrupts for GD32F170xx and GD32F190xx devices.
- 4 bits interrupt priority configuration-16 priority levels.
- Efficient interrupt processing.
- Support exception pre-emption and tail-chaining.
- Wake up system from power saving mode.
- Up to 23 independent edge detectors in EXTI.
- 3 trigger types: rising, falling and both edges.
- Software interrupt or event trigger.
- Trigger sources configurable.

### 5.3. Interrupts function overview

The ARM Cortex-M3 processor and the Nested Vectored Interrupt Controller (NVIC) prioritize and handle all exceptions in Handler Mode. The processor state is automatically stored to the stack on an exception and automatically restored from the stack at the end of the Interrupt Service Routine (ISR).

The vector is fetched in parallel to the state saving, enabling efficient interrupt entry. The processor supports tail-chaining, which enables back-to-back interrupts to be performed

without the overhead of state saving and restoration.

**Table 5-1. NVIC exception types in Cotrex-M3**

Exception Type	Vector Number	Priority (a)	Vector Address	Description
-	0	-	0x0000_0000	Reserved
Reset	1	-3	0x0000_0004	Reset
NMI	2	-2	0x0000_0008	Non maskable interrupt.
HardFault	3	-1	0x0000_000C	All class of fault
MemManage	4	Programmable	0x0000_0010	Memory management
BusFault	5	Programmable	0x0000_0014	Prefetch fault, memory access fault
UsageFault	6	Programmable	0x0000_0018	Undefined instruction or illegal state
-	7-10	-	0x0000_001C - 0x0000_002B	Reserved
SVCall	11	Programmable	0x0000_002C	System service call via SWI instruction
Debug Monitor	12	Programmable	0x0000_0030	Debug Monitor
-	13	-	0x0000_0034	Reserved
PendSV	14	Programmable	0x0000_0038	Pendable request for system service
SysTick	15	Programmable	0x0000_003C	System tick timer(b)
Interrupts	16-89	Programmable	0x0000_0040 - 0x0000_0164	Peripheral Interrupts (See below table for detail)

**Table 5-2. Interrupt vector table of GD32F130xx and GD32F150xx devices**

Interrupt Number	Vector Number	Peripheral Interrupt Description	Vector Address
IRQ 0	16	Window watchdog interrupt	0x0000_0040
IRQ 1	17	LVD through EXTI Line detection interrupt	0x0000_0044
IRQ 2	18	RTC global interrupt	0x0000_0048
IRQ 3	19	Flash global interrupt	0x0000_004C
IRQ 4	20	RCU global interrupt	0x0000_0050
IRQ 5	21	EXTI Line0-1 interrupt	0x0000_0054
IRQ 6	22	EXTI Line2-3 interrupt	0x0000_0058
IRQ 7	23	EXTI Line4-15 interrupt	0x0000_005C
IRQ 8	24	TSI global interrupt	0x0000_0060
IRQ 9	25	DMA Channel0 global interrupt	0x0000_0064
IRQ 10	26	DMA Channel1-2 global interrupt	0x0000_0068
IRQ 11	27	DMA Channel3-4 global interrupt	0x0000_006C
IRQ 12	28	ADC and CMP0-1 interrupt	0x0000_0070
IRQ 13	29	TIMER0 Break, update, trigger and	0x0000_0074

Interrupt Number	Vector Number	Peripheral Interrupt Description	Vector Address
		commutation interrupt	
IRQ 14	30	TIMER0 Capture Compare interrupt	0x0000_0078
IRQ 15	31	TIMER1 global interrupt	0x0000_007C
IRQ 16	32	TIMER2 global interrupt	0x0000_0080
IRQ 17	33	TIMER5 and DAC global interrupt	0x0000_0084
IRQ 18	34	Reserved	0x0000_0088
IRQ 19	35	TIMER13 global interrupt	0x0000_008C
IRQ 20	36	TIMER14 global interrupt	0x0000_0090
IRQ 21	37	TIMER15 global interrupt	0x0000_0094
IRQ 22	38	TIMER16 global interrupt	0x0000_0098
IRQ 23	39	I2C0 event interrupt	0x0000_009C
IRQ 24	40	I2C1 event interrupt	0x0000_00A0
IRQ 25	41	SPI0 global interrupt	0x0000_00A4
IRQ 26	42	SPI1 global interrupt	0x0000_00A8
IRQ 27	43	USART0 global interrupt	0x0000_00AC
IRQ 28	44	USART1 global interrupt	0x0000_00B0
IRQ 29	45	Reserved	0x0000_00B4
IRQ 30	46	CEC global interrupt	0x0000_00B8
IRQ 31	47	Reserved	0x0000_00BC
IRQ 32	48	I2C0 error interrupt	0x0000_00C0
IRQ 33	49	Reserved	0x0000_00C4
IRQ 34	50	I2C1 error interrupt	0x0000_00C8
IRQ 35	51	I2C2 event interrupt	0x0000_00CC
IRQ 36	52	I2C2 error interrupt	0x0000_00D0
IRQ 37	53	USB D Low Priority interrupts	0x0000_00D4
IRQ 38	54	USB D High Priority interrupts	0x0000_00D8
IRQ 39-41	55-57	Reserved	0x0000_00DC- 0x0000_00E4
IRQ 42	58	USB D Wake Up through EXTI Line18 interrupt	0x0000_00E8
IRQ 43-47	59-63	Reserved	0x0000_00EC- 0x0000_00FC
IRQ 48	64	DMA Channel5-6 global interrupt	0x0000_0100
IRQ 49-50	65-66	Reserved	0x0000_0104- 0x0000_0108
IRQ 51	67	SPI2 global interrupt	0x0000_010C

Table 5-3. Interrupt vector table of GD32F170xx and GD32F190xx devices

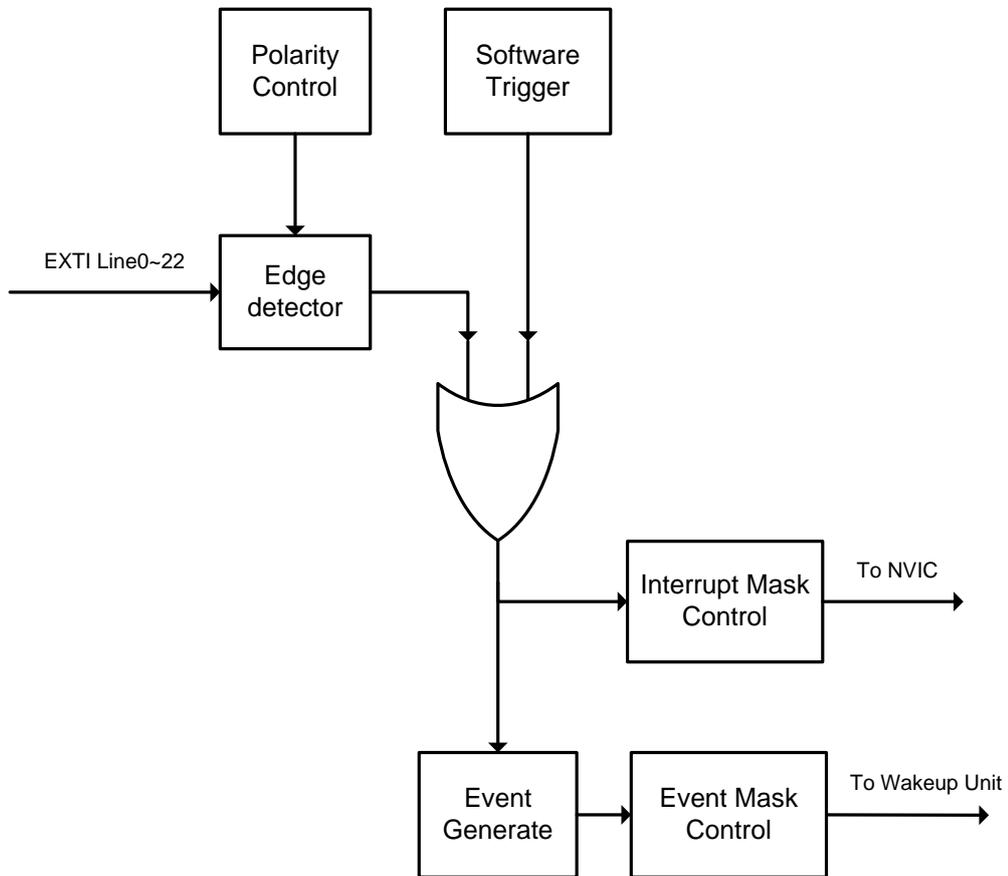
Interrupt Number	Vector Number	Peripheral Interrupt Description	Vector Address
IRQ 0	16	Window watchdog timer interrupt	0x0000_0040
IRQ 1	17	LVD through EXTI Line detection interrupt	0x0000_0044

Interrupt Number	Vector Number	Peripheral Interrupt Description	Vector Address
IRQ 2	18	RTC global interrupt	0x0000_0048
IRQ 3	19	Flash global interrupt	0x0000_004C
IRQ 4	20	RCU global interrupt	0x0000_0050
IRQ 5	21	EXTI Line0-1 interrupt	0x0000_0054
IRQ 6	22	EXTI Line2-3 interrupt	0x0000_0058
IRQ 7	23	EXTI Line4-15 interrupt	0x0000_005C
IRQ 8	24	TSI global interrupt	0x0000_0060
IRQ 9	25	DMA Channel0 global interrupt	0x0000_0064
IRQ 10	26	DMA Channel1-2 global interrupt	0x0000_0068
IRQ 11	27	DMA Channel3-4 global interrupt	0x0000_006C
IRQ 12	28	ADC and CMP0-1 interrupt	0x0000_0070
IRQ 13	29	TIMER0 Break, update, trigger and commutation interrupt	0x0000_0074
IRQ 14	30	TIMER0 Capture Compare interrupt	0x0000_0078
IRQ 15	31	TIMER1 global interrupt	0x0000_007C
IRQ 16	32	TIMER2 global interrupt	0x0000_0080
IRQ 17	33	TIMER5 and DAC global interrupt	0x0000_0084
IRQ 18	34	Reserved	0x0000_0088
IRQ 19	35	TIMER13 global interrupt	0x0000_008C
IRQ 20	36	TIMER14 global interrupt	0x0000_0090
IRQ 21	37	TIMER15 global interrupt	0x0000_0094
IRQ 22	38	TIMER16 global interrupt	0x0000_0098
IRQ 23	39	I2C0 event interrupt	0x0000_009C
IRQ 24	40	I2C1 event interrupt	0x0000_00A0
IRQ 25	41	SPI0 global interrupt	0x0000_00A4
IRQ 26	42	SPI1 global interrupt	0x0000_00A8
IRQ 27	43	USART0 global interrupt	0x0000_00AC
IRQ 28	44	USART1 global interrupt	0x0000_00B0
IRQ 29	45	Reserved	0x0000_00B4
IRQ 30	46	CEC global interrupt	0x0000_00B8
IRQ 31	47	Reserved	0x0000_00BC
IRQ 32	48	I2C0 error interrupt	0x0000_00C0
IRQ 33	49	Reserved	0x0000_00C4
IRQ 34	50	I2C1 error interrupt	0x0000_00C8
IRQ 35	51	I2C2 event interrupt	0x0000_00CC
IRQ 36	52	I2C2 error interrupt	0x0000_00D0
IRQ 37-42	53-58	Reserved	0x0000_00D4- 0x0000_00E8
IRQ43	59	CAN0_TX	0x0000_00EC
IRQ44	60	CAN0_RX0	0x0000_00F0

Interrupt Number	Vector Number	Peripheral Interrupt Description	Vector Address
IRQ45	61	CAN0_RX1	0x0000_00F4
IRQ46	62	CAN0_SCE	0x0000_00F8
IRQ 47	63	SLCD	0x0000_00FC
IRQ 48	64	DMA Channe5-6 global interrupt	0x0000_0100
IRQ 49-50	65-66	Reserved	0x0000_0104- 0x0000_0108
IRQ 51	67	SPI2 global interrupt	0x0000_010C
IRQ52-69	68-85	Reserved	0x0000_0110- 0x0000_0154
IRQ70	86	CAN1_TX	0x0000_0158
IRQ71	87	CAN1_RX0	0x0000_015C
IRQ72	88	CAN1_RX1	0x0000_0160
IRQ73	89	CAN1_SCE	0x0000_0164

#### 5.4. External interrupt and event (EXTI) block diagram

Figure 5-1. Block diagram of EXTI.



## 5.5. External interrupt and Event function overview

The EXTI contains up to 23 independent edge detectors and generates 28 interrupt requests or events to the processor. The EXTI has three trigger types: rising edge, falling edge and both edges. Each edge detector in the EXTI can be configured and masked independently.

The EXTI trigger source includes 16 lines from I/O pins and 8 lines for GD32F130xx and GD32F150xx devices (including LVD, RTC, USB, USART, CEC and CMP, please refer to [Table 5-4. EXTI source of GD32F130xx and GD32F150xx devices](#) for detail) or 7 lines for GD32F170xx and GD32F190xx devices (including LVD, RTC, USART, CEC and CMP, please refer to [Table 5-5. EXTI source of GD32F170xx and GD32F190xx devices](#) for detail) from the internal modules. All GPIO pins can be selected as an EXTI trigger source by configuring SYSCFG\_EXTISSx registers in SYSCFG module (please refer to [System configuration registers \(SYSCFG\)](#) for detail).

EXTI can provide not only interrupts but also event signals to the processor. The Cortex-M3 processor fully implements the Wait-For-Interrupt (WFI), Wait-For-Event (WFE) and the Send Event (SEV) instructions. There is a Wake-up Interrupt Controller (WIC) in chip, which enables the processor and NVIC to be put into a very low-power sleep mode leaving the WIC to identify and prioritize interrupts and events. EXTI can be used to wake up the processor and the whole system when some expected events occur, such as a special I/O pin toggling or RTC alarm.

The internal trigger sources from CEC and USART are able to generate event for waking up the system. However, the two modules are also requested to generate a synchronous interrupt for the processor to wake up the CPU from Deep-sleep mode. Both internal trigger lines can be masked by setting corresponding INTEN and EVEN registers in EXTI module.

### Hardware Trigger

Hardware trigger may be used to detect the voltage change of external or internal signals. The software should follow these steps to use this function:

1. Configure EXTI sources in SYSCFG module based on application requirement.
2. Configure EXTI\_RTEN and EXTI\_FTEN to enable the rising or falling detection on related pins. (Software may set both RTENx and FTENx for a pin at the same time to detect both rising and falling changes on this pin).
3. Enable interrupts or events by setting related EXTI\_INTEN or EXTI\_EVEN bits.
4. EXTI starts to detect changes on the configured pins. The related PDx bits in EXTI\_PD will be set when desired change is detected on these pins and thus, trigger interrupt or event for software. The software should response to the interrupts or events and clear these PDx bits.

### Software Trigger

Software may also trigger EXTI interrupts or events following these steps:

1. Enable interrupts or events by setting related EXTI\_INTEN or EXTI\_EVENT bits.
2. Set SWIEVx bits in EXTI\_SWIEV register. The related PD bits will be set immediately and thus, trigger interrupts or events. Software should response to these interrupts, and clear related PDx bits.

**Table 5-4. EXTI source of GD32F130xx and GD32F150xx devices**

EXTI Line Number	Source	Attribute
0	PA0 / PB0 / PC0 / PF0	External
1	PA1 / PB1 / PC1 / PF1	External
2	PA2 / PB2 / PC2 / PD2	External
3	PA3 / PB3 / PC3	External
4	PA4 / PB4 / PC4 / PF4	External
5	PA5 / PB5 / PC5 / PF5	External
6	PA6 / PB6 / PC6 / PF6	External
7	PA7 / PB7 / PC7 / PF7	External
8	PA8 / PB8 / PC8	External
9	PA9 / PB9 / PC9	External
10	PA10 / PB10 / PC10	External
11	PA11 / PB11 / PC11	External
12	PA12 / PB12 / PC12	External
13	PA13 / PB13 / PC13	External
14	PA14 / PB14 / PC14	External
15	PA15 / PB15 / PC15	External
16	LVD	External
17	RTC Alarm	External
18	USB Wakeup	External
19	RTC tamper and Timestamp	External
20	Reserved	Reserved
21	CMP0 output	External
22	CMP1 output	External
23	Reserved	Reserved
24	Reserved	Reserved
25	USART0 Wakeup	Internal
26	Reserved	Reserved
27	CEC Wakeup	Internal

**Table 5-5. EXTI source of GD32F170xx and GD32F190xx devices**

EXTI Line Number	Source	Attribute
0	PA0 / PB0 / PC0 / PF0	External
1	PA1 / PB1 / PC1 / PF1	External
2	PA2 / PB2 / PC2 / PD2	External
3	PA3 / PB3 / PC3	External

EXTI Line Number	Source	Attribute
4	PA4 / PB4 / PC4 / PF4	External
5	PA5 / PB5 / PC5 / PF5	External
6	PA6 / PB6 / PC6 / PF6	External
7	PA7 / PB7 / PC7 / PF7	External
8	PA8 / PB8 / PC8	External
9	PA9 / PB9 / PC9	External
10	PA10 / PB10 / PC10	External
11	PA11 / PB11 / PC11	External
12	PA12 / PB12 / PC12	External
13	PA13 / PB13 / PC13	External
14	PA14 / PB14 / PC14	External
15	PA15 / PB15 / PC15	External
16	LVD	External
17	RTC Alarm	External
18	Reserved	Reserved
19	RTC tamper and Timestamp	External
20	Reserved	Reserved
21	CMP0 output	External
22	CMP1 output	External
23	Reserved	Reserved
24	Reserved	Reserved
25	USART0 Wakeup	Internal
26	Reserved	Reserved
27	CEC Wakeup	Internal

## 5.6. Register definition

### 5.6.1. Interrupt Enable register (EXTI\_INTEN)

Address offset: 0x00

Reset value: 0x0F90 0000

This register has to be accessed by word(32-bit).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				INTEN27	INTEN26	INTEN25	INTEN24	INTEN23	INTEN22	INTEN21	INTEN20	INTEN19	INTEN18	INTEN17	INTEN16
				rw											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INTEN15	INTEN14	INTEN13	INTEN12	INTEN11	INTEN10	INTEN9	INTEN8	INTEN7	INTEN6	INTEN5	INTEN4	INTEN3	INTEN2	INTEN1	INTEN0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:28	Reserved	Must be kept at reset value
27: 0	INTENx	Interrupt mask bit x(x=0..27) 0: Interrupt from Linex is disabled 1: Interrupt from Linex is enabled

### 5.6.2. Event enable register (EXTI\_EVEN)

Address offset: 0x04

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				EVEN27	EVEN26	EVEN25	EVEN24	EVEN23	EVEN22	EVEN21	EVEN20	EVEN19	EVEN18	EVEN17	EVEN16
				rw											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EVEN15	EVEN14	EVEN13	EVEN12	EVEN11	EVEN10	EVEN9	EVEN8	EVEN7	EVEN6	EVEN5	EVEN4	EVEN3	EVEN2	EVEN1	EVEN0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:28	Reserved	Must be kept at reset value
27: 0	EVENx	Event enable bit x(x=0..27) 0: Event from Linex is disabled 1: Event from Linex is enabled

## 5.6.3. Rising edge trigger enable register (EXTI\_RTEN)

Address offset: 0x08

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									RTEN22	RTEN21	Reserved	RTEN19	RTEN18	RTEN17	RTEN16
									rw	rw		rw		rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTEN15	RTEN14	RTEN13	RTEN12	RTEN11	RTEN10	RTEN9	RTEN8	RTEN7	RTEN6	RTEN5	RTEN4	RTEN3	RTEN2	RTEN1	RTEN0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:23	Reserved	Must be kept at reset value
22:21	RTENx	Rising edge trigger enable (x=21,22) 0: Rising edge of Linex is not valid 1: Rising edge of Linex is valid as an interrupt/event request
20	Reserved	Must be kept at reset value
19	RTEN19	Rising edge trigger enable 0: Rising edge of Line19 is not valid 1: Rising edge of Line19 is valid as an interrupt/event request
18	RTEN18	Rising edge trigger enable This bit is valid only for GD32F150xx device. 0: Rising edge of Line18 is invalid 1: Rising edge of Line18 is valid as an interrupt/event request
17: 0	RTENx	Rising edge trigger enable (x=0,17) 0: Rising edge of Linex is invalid 1: Rising edge of Linex is valid as an interrupt/event request

## 5.6.4. Falling edge trigger enable register (EXTI\_FTEN)

Address offset: 0x0C

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									FTEN22	FTEN21	Reserved	FTEN19	FTEN18	FTEN17	FTEN16
									rw	rw		rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

FTEN15	FTEN14	FTEN13	FTEN12	FTEN11	FTEN10	FTEN9	FTEN8	FTEN7	FTEN6	FTEN5	FTEN4	FTEN3	FTEN2	FTEN1	FTEN0
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

Bits	Fields	Descriptions
31: 23	Reserved	Must be kept at reset value
22: 21	FTENx	Falling edge trigger enable (x=21,22) 0: Falling edge of Linex is invalid 1: Falling edge of Linex is valid as an interrupt/event request
20	Reserved	Must be kept at reset value
19	FTEN19	Falling edge trigger enable 0: Falling edge of Line19 is invalid 1: Falling edge of Line19 is valid as an interrupt/event request
18	FTEN18	Falling edge trigger enable This bit is valid only for GD32F150xx device. 0: Falling edge of Line18 is invalid 1: Falling edge of Line18 is valid as an interrupt/event request
17: 0	FTENx	Falling edge trigger enable (x=0,17) 0: Falling edge of Linex is invalid 1: Falling edge of Linex is valid as an interrupt/event request

### 5.6.5. Software interrupt event register (EXTI\_SWIEV)

Address offset: 0x10

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									SWIEV22	SWIEV21	Reserved	SWIEV19	SWIEV18	SWIEV17	SWIEV16
									r/w	r/w	r/w		r/w	r/w	r/w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWIEV15	SWIEV14	SWIEV13	SWIEV12	SWIEV11	SWIEV10	SWIEV9	SWIEV8	SWIEV7	SWIEV6	SWIEV5	SWIEV4	SWIEV3	SWIEV2	SWIEV1	SWIEV0
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

Bits	Fields	Descriptions
31:23	Reserved	Must be kept at reset value
22: 21	SWIEVx	Interrupt/Event software trigger (x=21,22) 0: Deactivate the EXTIx software interrupt/event request 1: Activate the EXTIx software interrupt/event request

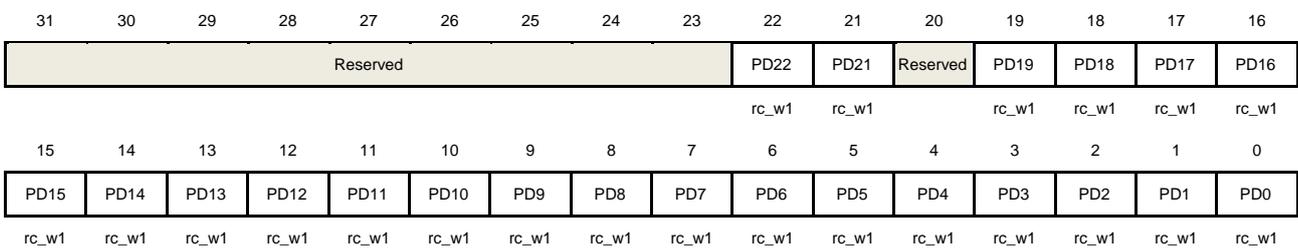
20	Reserved	Must be kept at reset value
19: 0	SWIEVx	Interrupt/Event software trigger (x=0,19) 0: Deactivate the EXTIx software interrupt/event request 1: Activate the EXTIx software interrupt/event request

### 5.6.6. Pending register (EXTI\_PD)

Address offset: 0x14

Reset value: undefined

This register has to be accessed by word(32-bit).



Bits	Fields	Descriptions
31: 23	Reserved	Must be kept at reset value
22: 21	PDx	Interrupt pending status (x=21,22) 0: EXIT Linex is not triggered 1: EXIT Linex is triggered This bit is cleared to 0 by writing 1 to it.
20	Reserved	Must be kept at reset value
19: 0	PDx	Interrupt pending status (x=0,19) 0: EXIT Linex is not triggered 1: EXIT Linex is triggered This bit is cleared to 0 by writing 1 to it.

## 6. General-purpose I/Os (GPIO)

### 6.1. Overview

There are up to 55 general purpose I/O pins, (GPIO), named PA0 ~ PA15 and PB0 ~ PB15, PC0 ~ PC15, PD2, PF0/PF1, PF4 ~ PF7 for the device to implement logic input/output functions. Each GPIO port has related control and configuration registers to satisfy the requirements of specific applications.

The GPIO ports are pin-shared with other alternative functions (AFs) to obtain maximum flexibility on the package pins. The GPIO pins can be used as alternative functional pins by configuring the corresponding registers regardless of the AF input or output pins.

Each of the GPIO pins can be configured by software as output (push-pull or open-drain), as input, as peripheral alternate function or as analog mode. Each GPIO pin can be configured as pull-up, pull-down or no pull-up/pull-down. All GPIOs are high-current capable except for analog mode.

### 6.2. Characteristics

- Input/output direction control.
- Schmitt Trigger Input function enable control.
- Each pin weak pull-up/pull-down function.
- Output push-pull/open drain enable control.
- Output set/reset control.
- Output drive speed selection.
- Analog input/output configurations.
- Alternate function input/output configurations.
- Port configuration lock.

**For GD32F170xx and GD32F190xx devices, additional features are shown below.**

- Single cycle toggle output capability.

### 6.3. Function overview

Each of the general-purpose I/O ports can be configured as GPIO inputs, GPIO outputs, AF function or analog mode by GPIO 32-bit configuration registers (GPIOx\_CTL). AFIO input or output is selected by AFIO output enable. When the port is output (GPIO output or AFIO

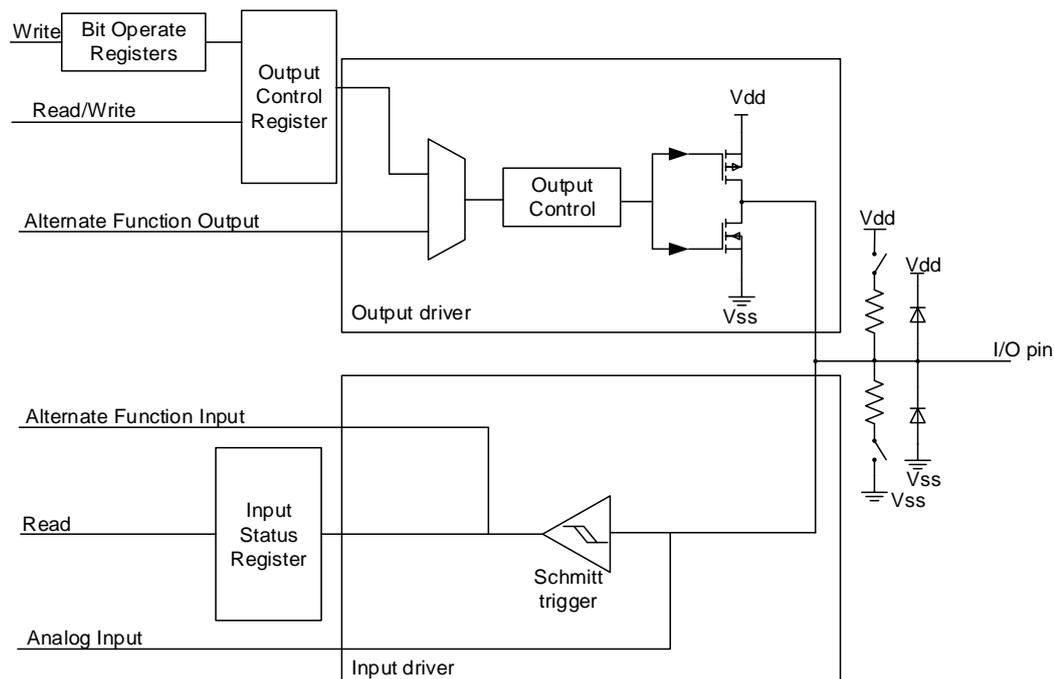
output), it can be configured as push-pull or open drain mode by GPIO output mode registers (GPIOx\_OMODE). And the port max speed can be configured by GPIO output speed registers (GPIOx\_OSPD). Each port can be configured as floating (no pull-up and pull-down), pull-up or pull-down function by GPIO pull-up/pull-down registers (GPIOx\_PUD).

**Table 6-1. GPIO configuration table**

PAD TYPE			CTLn	OMn	PUDn
GPIO INPUT	X	Floating	00	X	00
		Pull-up			01
		Pull-down			10
GPIO OUTPUT	Push-pull	Floating	01	0	00
		Pull-up			01
		Pull-down			10
	Open-drain	Floating		1	00
		Pull-up			01
		Pull-down			10
AFIO INPUT	X	Floating	10	X	00
		Pull-up			01
		Pull-down			10
AFIO OUTPUT	Push-pull	Floating	10	0	00
		Pull-up			01
		Pull-down			10
	Open-drain	Floating		1	00
		Pull-up			01
		Pull-down			10
ANALOG	X	X	11	X	XX

**Figure 6-1. Basic structure of a standard I/O port bit** shows the basic structure of an I/O Port bit.

**Figure 6-1. Basic structure of a standard I/O port bit**



### 6.3.1. GPIO pin configuration

During or just after the reset period, the alternative functions are all inactive and the GPIO ports are configured into the input floating mode that input disabled without Pull-Up(PU)/Pull-Down(PD) resistors. But the Serial-Wired Debug pins are in input PU/PD mode after reset:

PA14: SWCLK in input pull-down mode

PA13: SWDIO in input pull-up mode

The GPIO pins can be configured as inputs or outputs. And all GPIO pins have an internal weak pull-up and weak pull-down which can be chosen. When the GPIO pins are configured as input pins, the data on the external pads can be captured at every AHB2 clock cycle to the port input status register (GPIO\_ISTAT).

When the GPIO pins are configured as output pins, user can configure the speed of the ports. And chooses the output driver mode: Push-Pull or Open-Drain mode. The value of the port output control register (GPIO\_OCTL) is output on the I/O pin.

When programming the GPIO\_OCTL at bit level is not need to disable interrupts, user can modify only one or several bits in a single atomic AHB2 write access by programming '1' to the Bit Operate Register (GPIO\_BOP, or for clearing only GPIO\_BC , or for toggle only GPIO\_TG). The other bits will not be affected.

### 6.3.2. Alternate functions (AF)

When the port is configured as AFIO (set CTLY to "10" bits, which is in GPIOx\_CTL registers),

the port is used as peripheral alternate functions. Each port has sixteen alternate functions can be configured by GPIO alternate functions select registers (GPIOx\_AFSELY(y=0,1)). The detail alternate function assignments for each port are in the device datasheet.

### 6.3.3. Additional functions

Some pins have additional functions, which have priority over the configuration in the standard GPIO registers. When for ADC or DAC additional functions, the port must be configured as analog mode. When for RTC, WKUPx and oscillators additional functions, the port type is set automatically by related RTC, PMU and RCU registers. These ports can be used as normal GPIO when the additional functions disabled.

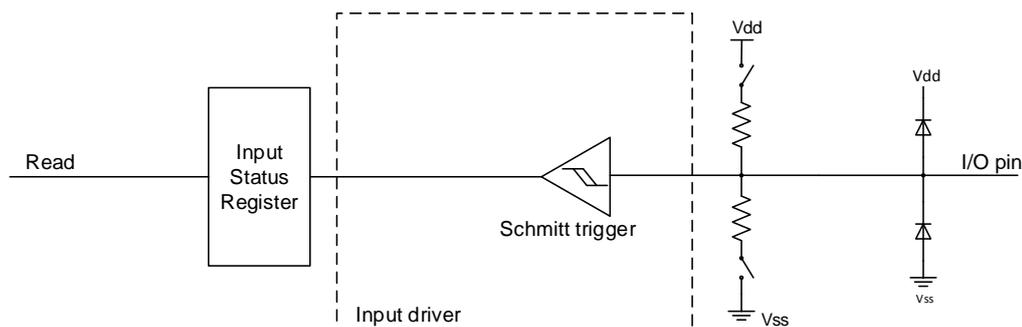
### 6.3.4. Input configuration

When GPIO pin is configured as input:

- The schmitt trigger input is activated.
- The weak pull-up and pull-down resistors could be chosen.
- Every AHB2 clock cycle the data present on the I/O pad is got to the port input status register.
- The output buffer is disabled.

[Figure 6-2. Input floating/pull up/pull down configurations](#) shows the input configuration of the I/O Port bit.

**Figure 6-2. Input floating/pull up/pull down configurations**



### 6.3.5. Output configuration

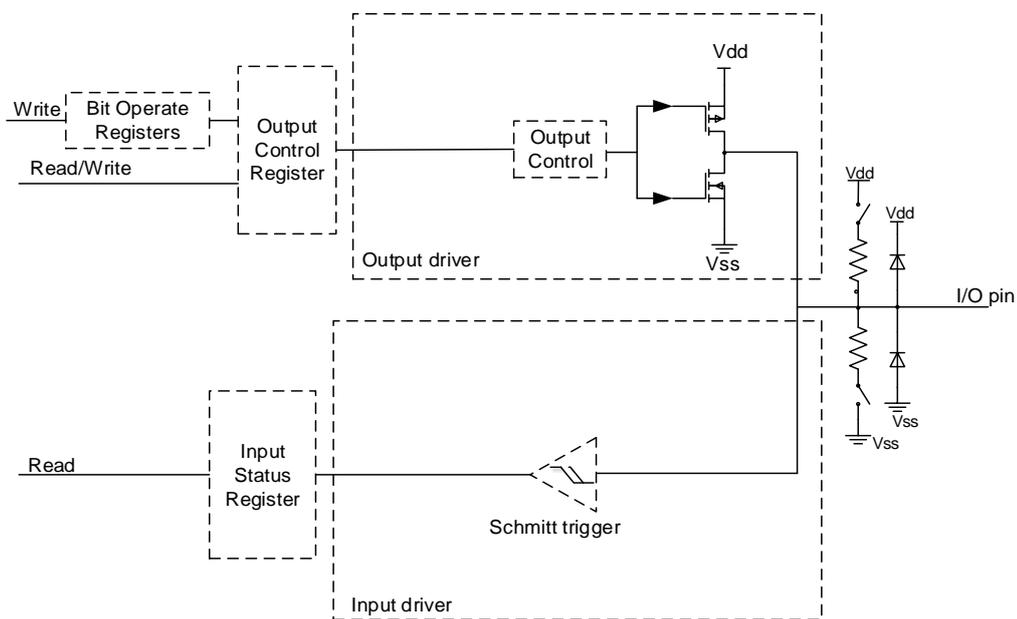
When GPIO pin is configured as output:

- The schmitt trigger input is activated.
- The weak pull-up and pull-down resistors could be chosen.
- The output buffer is enabled:

- Open-Drain mode: The pad outputs “0” when a “0” in the output control register; while the pad leaves Hi-Z when a “1” in the output control register.
- Push-Pull mode: The pad outputs “0” when a “0” in the output control register; while the pad outputs “1” when a “1” in the output control register.
- A read access to the port output control register gets the last written value in Push-Pull mode.
- A read access to the port input status register gets the I/O state in Open-Drain mode.

[Figure 6-3. Output configuration](#) shows the Output configuration of the I/O port bit.

**Figure 6-3. Output configuration**

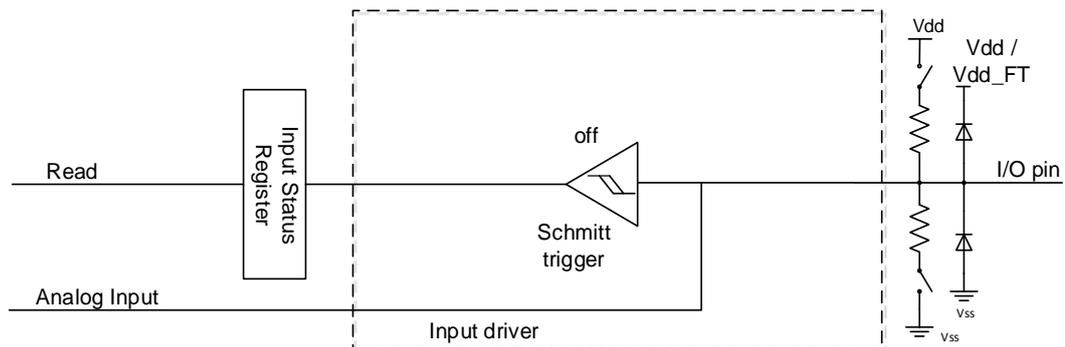


### 6.3.6. Analog configuration

When GPIO pin is used as analog configuration:

- The weak pull-up and pull-down resistors are disabled.
- The output buffer is disabled.
- The schmitt trigger input is de-activated.
- Read access to the port input status register gets the value “0”.

[Figure 6-4. High impedance-analog configuration](#) shows the high impedance-analog configuration.

**Figure 6-4. High impedance-analog configuration**


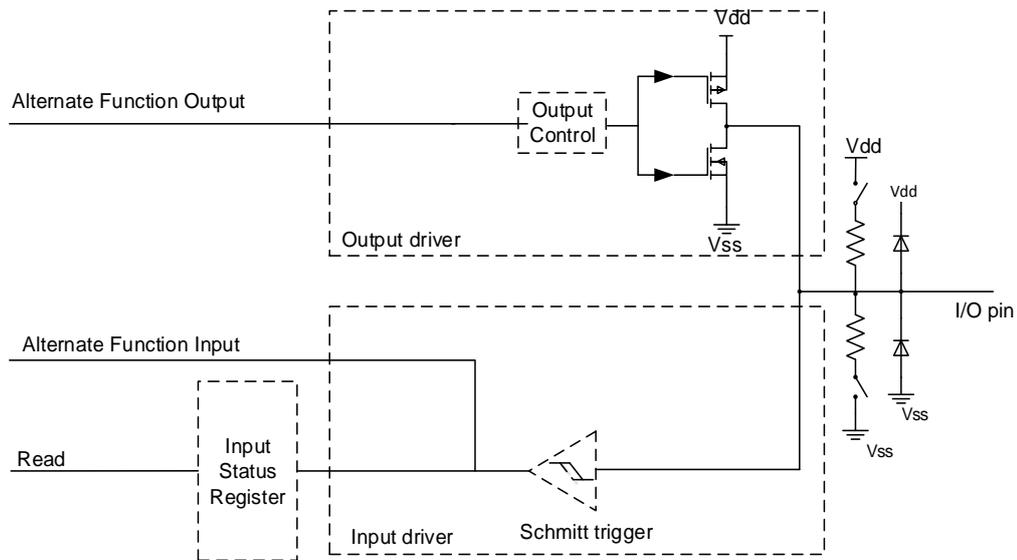
### 6.3.7. Alternate function (AF) configuration

To suit for different device packages, the GPIO supports some alternate functions to some other pins by software.

When be configured as Alternate Function:

- The output buffer is turned on in Open-Drain or Push-Pull configuration.
- The output buffer is driven by the peripheral.
- The schmitt trigger input is activated.
- The weak pull-up and pull-down resistors could be chosen.
- The data present on the I/O pin is sampled into the port input status register every AHB2 clock cycle.
- A read access to the port input status register gets the I/O state in open drain mode.
- A read access to the port output control register gets the last written value in Push-Pull mode.

[Figure 6-5. Alternate function configuration](#) shows the alternate function configuration of the I/O Port bit.

**Figure 6-5. Alternate function configuration**


### 6.3.8. GPIO locking function

The locking mechanism allows the IO configuration to be protected.

The protected registers are GPIOx\_CTL, GPIOx\_OMODE, GPIOx\_OSPD, GPIOx\_PUD, GPIOx\_AFSELY(y=0,1). It allows the I/O configuration to be frozen by the 32-bit locking register (GPIOx\_LOCK). When the LOCK sequence has been applied on a port bit, it is no longer able to modify the value of the port bit until the next reset. It should be recommended to be used in the configuration of driving a power module.

### 6.3.9. GPIO single cycle toggle function

GPIO could toggle the I/O output level in single AHB cycle by writing 1 to the corresponding bit of GPIOx\_TG register. The output signal frequency could up to the half of the AHB clock.

## 6.4. Register definition

### 6.4.1. Port control register (GPIOx\_CTL) (x=A..D,F)

Address offset: 0x00

Reset value: 0x2800 0000 for port A; 0x0000 0000 for others.

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CTL15[1:0]		CTL14[1:0]		CTL13[1:0]		CTL12[1:0]		CTL11[1:0]		CTL10[1:0]		CTL9[1:0]		CTL8[1:0]	
rw		rw		rw		rw		rw		rw		rw		rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTL7[1:0]		CTL6[1:0]		CTL5[1:0]		CTL4[1:0]		CTL3[1:0]		CTL2[1:0]		CTL1[1:0]		CTL0[1:0]	
rw		rw		rw		rw		rw		rw		rw		rw	

Bits	Fields	Descriptions
31:30	CTL15[1:0]	Pin 15 configuration bits These bits are set and cleared by software. Refer to CTL0[1:0] description
29:28	CTL14[1:0]	Pin 14 configuration bits These bits are set and cleared by software. Refer to CTL0[1:0] description
27:26	CTL13[1:0]	Pin 13 configuration bits These bits are set and cleared by software. Refer to CTL0[1:0] description
25:24	CTL12[1:0]	Pin 12 configuration bits These bits are set and cleared by software. Refer to CTL0[1:0] description
23:22	CTL11[1:0]	Pin 11 configuration bits These bits are set and cleared by software. Refer to CTL0[1:0] description
21:20	CTL10[1:0]	Pin 10 configuration bits These bits are set and cleared by software. Refer to CTL0[1:0] description
19:18	CTL9[1:0]	Pin 9 configuration bits These bits are set and cleared by software. Refer to CTL0[1:0] description
17:16	CTL8[1:0]	Pin 8 configuration bits These bits are set and cleared by software.

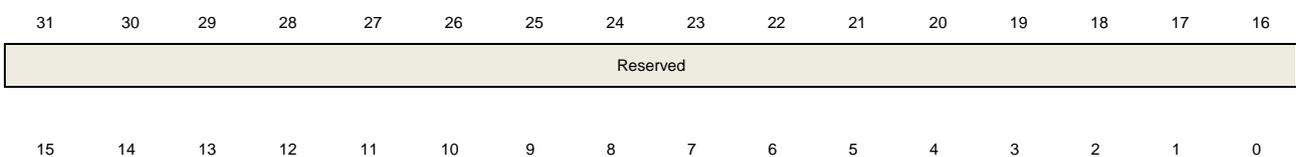
		Refer to CTL0[1:0] description
15:14	CTL7[1:0]	Pin 7 configuration bits These bits are set and cleared by software. Refer to CTL0[1:0] description
13:12	CTL6[1:0]	Pin 6 configuration bits These bits are set and cleared by software. Refer to CTL0[1:0] description
11:10	CTL5[1:0]	Pin 5 configuration bits These bits are set and cleared by software. Refer to CTL0[1:0] description
9:8	CTL4[1:0]	Pin 4 configuration bits These bits are set and cleared by software. Refer to CTL0[1:0] description
7:6	CTL3[1:0]	Pin 3 configuration bits These bits are set and cleared by software. Refer to CTL0[1:0] description
5:4	CTL2[1:0]	Pin 2 configuration bits These bits are set and cleared by software. Refer to CTL0[1:0] description
3:2	CTL1[1:0]	Pin 1 configuration bits These bits are set and cleared by software. Refer to CTL0[1:0] description
1:0	CTL0[1:0]	Pin 0 configuration bits These bits are set and cleared by software. 00: Input mode (reset state) 01: GPIO output mode 10: Alternate function mode. 11: Analog mode

### 6.4.2. Port output mode register (GPIOx\_OMODE) (x=A..D,F)

Address offset: 0x04

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



OM15	OM14	OM13	OM12	OM11	OM10	OM9	OM8	OM7	OM6	OM5	OM4	OM3	OM2	OM1	OM0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15	OM15	Pin 15 output mode bit These bits are set and cleared by software. Refer to OM0 description
14	OM14	Pin 14 output mode bit These bits are set and cleared by software. Refer to OM0 description
13	OM13	Pin 13 output mode bit These bits are set and cleared by software. Refer to OM0 description
12	OM12	Pin 12 output mode bit These bits are set and cleared by software. Refer to OM0 description
11	OM11	Pin 11 output mode bit These bits are set and cleared by software. Refer to OM0 description
10	OM10	Pin 10 output mode bit These bits are set and cleared by software. Refer to OM0 description
9	OM9	Pin 9 output mode bit These bits are set and cleared by software. Refer to OM0 description
8	OM8	Pin 8 output mode bit These bits are set and cleared by software. Refer to OM0 description
7	OM7	Pin 7 output mode bit These bits are set and cleared by software. Refer to OM0 description
6	OM6	Pin 6 output mode bit These bits are set and cleared by software. Refer to OM0 description
5	OM5	Pin 5 output mode bit These bits are set and cleared by software. Refer to OM0 description

4	OM4	Pin 4 output mode bit These bits are set and cleared by software. Refer to OM0 description
3	OM3	Pin 3 output mode bit These bits are set and cleared by software. Refer to OM0 description
2	OM2	Pin 2 output mode bit These bits are set and cleared by software. Refer to OM0 description
1	OM1	Pin 1 output mode bit These bits are set and cleared by software. Refer to OM0 description
0	OM0	Pin 0 output mode bit These bits are set and cleared by software. 0: Output push-pull mode (reset) 1: Output open-drain mode

### 6.4.3. Port output speed register (GPIOx\_OSPD) (x=A..D,F)

Address offset: 0x08

Reset value: 0x0C00 0000 for port A; 0x0000 0000 for others.

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSPD15[1:0]		OSPD14[1:0]		OSPD13[1:0]		OSPD12[1:0]		OSPD11[1:0]		OSPD10[1:0]		OSPD9[1:0]		OSPD8[1:0]	
rw		rw		rw		rw		rw		rw		rw		rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OSPD7[1:0]		OSPD6[1:0]		OSPD5[1:0]		OSPD4[1:0]		OSPD3[1:0]		OSPD2[1:0]		OSPD1[1:0]		OSPD0[1:0]	
rw		rw		rw		rw		rw		rw		rw		rw	

Bits	Fields	Descriptions
31:30	OSPD15[1:0]	Pin 15 output max speed bits These bits are set and cleared by software. Refer to OSPD0[1:0] description
29:28	OSPD14[1:0]	Pin 14 output max speed bits These bits are set and cleared by software. Refer to OSPD0[1:0] description
27:26	OSPD13[1:0]	Pin 13 output max speed bits These bits are set and cleared by software. Refer to OSPD0[1:0] description

25:24	OSPD12[1:0]	Pin 12 output max speed bits These bits are set and cleared by software. Refer to OSPD0[1:0] description
23:22	OSPD11[1:0]	Pin 11 output max speed bits These bits are set and cleared by software. Refer to OSPD0[1:0] description
21:20	OSPD10[1:0]	Pin 10 output max speed bits These bits are set and cleared by software. Refer to OSPD0[1:0] description
19:18	OSPD9[1:0]	Pin 9 output max speed bits These bits are set and cleared by software. Refer to OSPD0[1:0] description
17:16	OSPD8[1:0]	Pin 8 output max speed bits These bits are set and cleared by software. Refer to OSPD0[1:0] description
15:14	OSPD7[1:0]	Pin 7 output max speed bits These bits are set and cleared by software. Refer to OSPD0[1:0] description
13:12	OSPD6[1:0]	Pin 6 output max speed bits These bits are set and cleared by software. Refer to OSPD0[1:0] description
11:10	OSPD5[1:0]	Pin 5 output max speed bits These bits are set and cleared by software. Refer to OSPD0[1:0] description
9:8	OSPD4[1:0]	Pin 4 output max speed bits These bits are set and cleared by software. Refer to OSPD0[1:0] description
7:6	OSPD3[1:0]	Pin 3 output max speed bits These bits are set and cleared by software. Refer to OSPD0[1:0] description
5:4	OSPD2[1:0]	Pin 2 output max speed bits These bits are set and cleared by software. Refer to OSPD0[1:0] description
3:2	OSPD1[1:0]	Pin 1 output max speed bits These bits are set and cleared by software. Refer to OSPD0[1:0] description
1:0	OSPD0[1:0]	Pin 0 output max speed bits These bits are set and cleared by software.

x0: Output max speed 2M (reset state)  
 01: Output max speed 10M  
 11: Output max speed 50M

## 6.4.4. Port pull-up/down register (GPIOx\_PUD) (x=A..D,F)

Address offset: 0x0C

Reset value: 0x2400 0000 for port A; 0x0000 0000 for others.

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUD15[1:0]		PUD14[1:0]		PUD13[1:0]		PUD12[1:0]		PUD11[1:0]		PUD10[1:0]		PUD9[1:0]		PUD8[1:0]	
rw		rw		rw		rw		rw		rw		rw		rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PUD7[1:0]		PUD6[1:0]		PUD5[1:0]		PUD4[1:0]		PUD3[1:0]		PUD2[1:0]		PUD1[1:0]		PUD0[1:0]	
rw		rw		rw		rw		rw		rw		rw		rw	

Bits	Fields	Descriptions
31:30	PUD15[1:0]	Pin 15 pull-up or pull-down bits These bits are set and cleared by software. Refer to PUD0[1:0] description
29:28	PUD14[1:0]	Pin 14 pull-up or pull-down bits These bits are set and cleared by software. Refer to PUD0[1:0] description
27:26	PUD13[1:0]	Pin 13 pull-up or pull-down bits These bits are set and cleared by software. Refer to PUD0[1:0] description
25:24	PUD12[1:0]	Pin 12 pull-up or pull-down bits These bits are set and cleared by software. Refer to PUD0[1:0] description
23:22	PUD11[1:0]	Pin 11 pull-up or pull-down bits These bits are set and cleared by software. Refer to PUD0[1:0] description
21:20	PUD10[1:0]	Pin 10 pull-up or pull-down bits These bits are set and cleared by software. Refer to PUD0[1:0] description
19:18	PUD9[1:0]	Pin 9 pull-up or pull-down bits These bits are set and cleared by software. Refer to PUD0[1:0] description
17:16	PUD8[1:0]	Pin 8 pull-up or pull-down bits

		These bits are set and cleared by software. Refer to PUD0[1:0] description
15:14	PUD7[1:0]	Pin 7 pull-up or pull-down bits These bits are set and cleared by software. Refer to PUD0[1:0] description
13:12	PUD6[1:0]	Pin 6 pull-up or pull-down bits These bits are set and cleared by software. Refer to PUD0[1:0] description
11:10	PUD5[1:0]	Pin 5 pull-up or pull-down bits These bits are set and cleared by software. Refer to PUD0[1:0] description
9:8	PUD4[1:0]	Pin 4 pull-up or pull-down bits These bits are set and cleared by software. Refer to PUD0[1:0] description
7:6	PUD3[1:0]	Pin 3 pull-up or pull-down bits These bits are set and cleared by software.  Refer to PUD0[1:0] description
5:4	PUD2[1:0]	Pin 2 pull-up or pull-down bits These bits are set and cleared by software. Refer to PUD0[1:0] description
3:2	PUD1[1:0]	Pin 1 pull-up or pull-down bits These bits are set and cleared by software. Refer to PUD0[1:0] description
1:0	PUD0[1:0]	Pin 0 pull-up or pull-down bits These bits are set and cleared by software. 00: Floating mode, no pull-up and pull-down (reset state) 01: With pull-up mode 10: With pull-down mode 11: Reserved

### 6.4.5. Port input status register (GPIOx\_ISTAT) (x=A..D,F)

Address offset: 0x10  
reset value: 0x0000 XXXX.

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISTAT15	ISTAT14	ISTAT13	ISTAT12	ISTAT11	ISTAT10	ISTAT9	ISTAT8	ISTAT7	ISTAT6	ISTAT5	ISTAT4	ISTAT3	ISTAT2	ISTAT1	ISTAT0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	ISTAT <sub>y</sub>	Port input status (y=0..15) These bits are set and cleared by hardware. 0: Input signal low 1: Input signal high

### 6.4.6. Port output control register (GPIOx\_OCTL) (x=A..D,F)

Address offset: 0x14

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

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
OCTL15	OCTL14	OCTL13	OCTL12	OCTL11	OCTL10	OCTL9	OCTL8	OCTL7	OCTL6	OCTL5	OCTL4	OCTL3	OCTL2	OCTL1	OCTL0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	OCTL <sub>y</sub>	Port output control (y=0..15) These bits are set and cleared by software. 0: Pin output low 1: Pin output high

### 6.4.7. Port bit operate register (GPIOx\_BOP) (x=A..D,F)

Address offset: 0x18

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CR15	CR14	CR13	CR12	CR11	CR10	CR9	CR8	CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0
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
BOP15	BOP14	BOP13	BOP12	BOP11	BOP10	BOP9	BOP8	BOP7	BOP6	BOP5	BOP4	BOP3	BOP2	BOP1	BOP0

w w w w w w w w w w w w w w w w

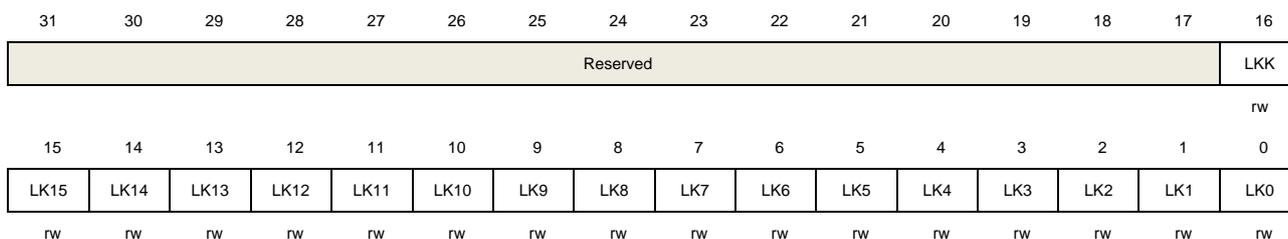
Bits	Fields	Descriptions
31:16	CRy	Port Clear bit y(y=0..15) These bits are set and cleared by software. 0: No action on the corresponding OCTLY bit 1: Clear the corresponding OCTLY bit
15:0	BOPy	Port Set bit y(y=0..15) These bits are set and cleared by software. 0: No action on the corresponding OCTLY bit 1: Set the corresponding OCTLY bit

### 6.4.8. Port configuration lock register (GPIOx\_LOCK) (x=A, B)

Address offset: 0x1C

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)



Bits	Fields	Descriptions
31:17	Reserved	Must be kept at reset value
16	LKK	Lock key It can only be setted using the Lock Key Writing Sequence.And can always be read. 0: Port configuration lock key not active 1: Port configuration lock key active. GPIO_LOCK register is locked until an MCU reset..  LOCK key writing sequence Write 1→Write 0→Write 1→ Read 0→ Read 1 <b>Note:</b> The value of LKy(y=0..15) must hold during the LOCK Key Writing sequence.
15:0	LKy	Port Lock bit y(y=0..15) These bits are set and cleared by software 0: Port configuration not locked 1: Port configuration locked

### 6.4.9. Alternate function selected register0 (GPIOx\_AFSEL0) (x=A, B, C)

Address offset: 0x20

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



Bits	Fields	Descriptions
31:28	SEL7[3:0]	Pin 7 alternate function selected These bits are set and cleared by software. Refer to SEL0[3:0] description
27:24	SEL6[3:0]	Pin 6 alternate function selected These bits are set and cleared by software. Refer to SEL0[3:0] description
23:20	SEL5[3:0]	Pin 5 alternate function selected These bits are set and cleared by software. Refer to SEL0[3:0] description
19:16	SEL4[3:0]	Pin 4 alternate function selected These bits are set and cleared by software. Refer to SEL0[3:0] description
15:12	SEL3[3:0]	Pin 3 alternate function selected These bits are set and cleared by software. Refer to SEL0[3:0] description
11:8	SEL2[3:0]	Pin 2 alternate function selected These bits are set and cleared by software. Refer to SEL0[3:0] description
7:4	SEL1[3:0]	Pin 1 alternate function selected These bits are set and cleared by software. Refer to SEL0[3:0] description
3:0	SEL0[3:0]	Pin 0 alternate function selected These bits are set and cleared by software. 0000: AF0 selected (reset value) 0001: AF1 selected 0010: AF2 selected

- 0011: AF3 selected
- 0100: AF4 selected (Port A,B only)
- 0101: AF5 selected (Port A,B only)
- 0110: AF6 selected (Port A,B only)
- 0111: AF7 selected (Port A,B only)
- 1000: Reserved
- 1001: AF9 selected (Port A,B only) (For GD32F170xx and GD32F190xx devices)
- 1010: Reserved
- 1011: AF11 selected (For GD32F170xx and GD32F190xx devices)
- 1100 ~ 1111: Reserved

## 6.4.10. Alternate function selected register1 (GPIOx\_AFSEL1) (x=A,B,C)

Address offset: 0x24

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



Bits	Fields	Descriptions
31:28	SEL15[3:0]	Pin 15 alternate function selected These bits are set and cleared by software. Refer to SEL8[3:0] description
27:24	SEL14[3:0]	Pin 14 alternate function selected These bits are set and cleared by software. Refer to SEL8[3:0] description
23:20	SEL13[3:0]	Pin 13 alternate function selected These bits are set and cleared by software. Refer to SEL8[3:0] description
19:16	SEL12[3:0]	Pin 12 alternate function selected These bits are set and cleared by software. Refer to SEL8[3:0] description
15:12	SEL11[3:0]	Pin 1 alternate function selected These bits are set and cleared by software. Refer to SEL8[3:0] description
11:8	SEL10[3:0]	Pin 10 alternate function selected

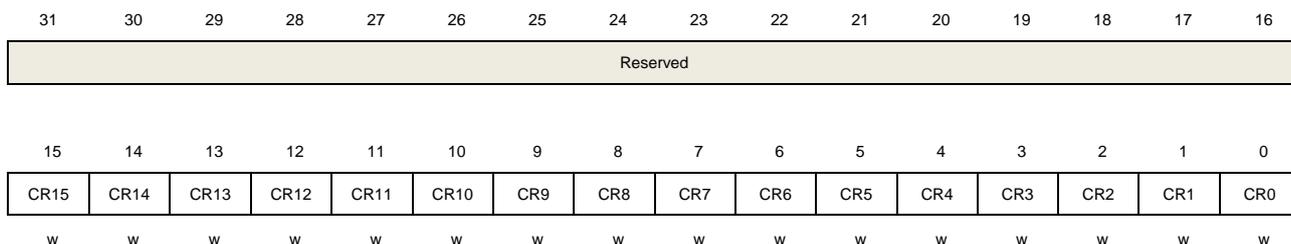
		These bits are set and cleared by software. Refer to SEL8[3:0] description
7:4	SEL9[3:0]	Pin 9 alternate function selected These bits are set and cleared by software. Refer to SEL8[3:0] description
3:0	SEL8[3:0]	Pin 8 alternate function selected These bits are set and cleared by software. 0000: AF0 selected (reset value) 0001: AF1 selected 0010: AF2 selected 0011: AF3 selected 0100: AF4 selected (Port A,B only) 0101: AF5 selected (Port A,B only) 0110: AF6 selected (Port A,B only) 0111: AF7 selected (Port A,B only) 1000: Reserved 1001: AF9 selected (Port A,B only) (For GD32F170xx and GD32F190xx devices) 1010: Reserved 1011: AF11 selected (For GD32F170xx and GD32F190xx devices) 1100 ~ 1111: Reserved

### 6.4.11. Bit clear register (GPIOx\_BC) (x=A..D,F)

Address offset: 0x28

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



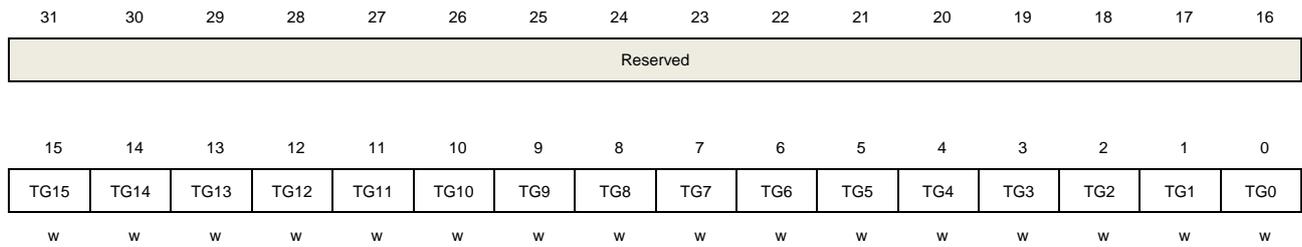
Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	CRy	Port Clear bit y(y=0..15) These bits are set and cleared by software. 0: No action on the corresponding OCTLY bit 1: Clear the corresponding OCTLY bit

### 6.4.12. Port bit toggle register (GPIOx\_TGx) (x=A..D,F) of GD32F170xx and GD32F190xx devices

Address offset: 0x2C

Reset value: 0x0000 0000

This register can be accessed by byte(8-bit), half-word(16-bit) and word(32-bit)



Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	TGy	Port Toggle bit y(y=0..15) These bits are set and cleared by software. 0: No action on the corresponding OCTLY bit 1: Toggle the corresponding OCTLY bit

## 7. CRC calculation unit

### 7.1. Overview

A cyclic redundancy check (CRC) is an error-detecting code commonly used in digital networks and storage devices to detect accidental changes to raw data.

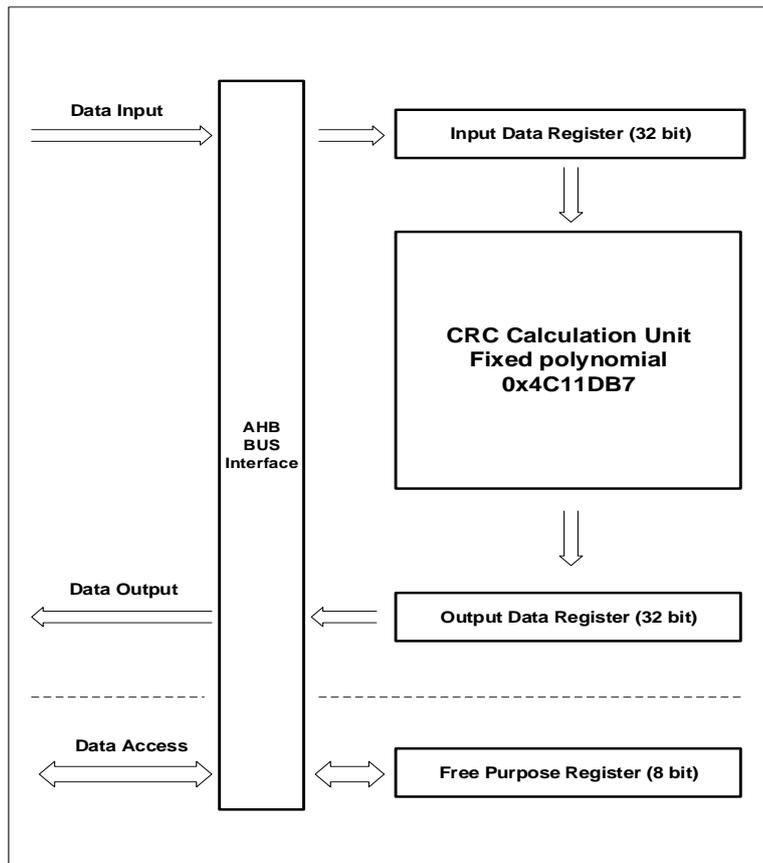
This CRC calculation unit can be used to calculate 32/16/8 bit CRC code within fixed polynomial.

### 7.2. Characteristics

- Input data supports 8/16/32 size bit.
- Different input size for different calculation time. 1/2/4 cycle for 8/16/32 bits.
- Input and output data can be reversed.
- Fixed polynomial: 0x4C11DB7  
$$X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$$

This 32-bit CRC polynomial is a common polynomial used in Ethernet.
- User configurable initial value after CRC reset.
- Free 8 bit register is unrelated for calculation and can be used for any other goals by any other peripheral devices.

Figure 7-1. Block Diagram of CRC Calculation Unit



### 7.3. Function overview

- CRC calculation unit is used to calculate the 32-bit raw data, and CRC\_DATA register will receive the raw data and store the calculation result.

If do not clear the CRC\_DATA register by software setting CRC\_CTL register, the new input raw data will calculate based on the result of previous value of CRC\_DATA.

CRC calculation will spend 4/2/1 AHB clock cycles for 32/16/8 bit data size, during this period AHB will not be hanged because of the existence of the 32bit input buffer.
- This module supplies an 8-bit free register CRC\_FDATA.

CRC\_FDATA is unrelated to the CRC calculation, any value you write in will be read out at anytime.
- Reversible function can reverse the input data and output data.

For input data, 3 reverse types can be selected.

Original data is 0x1A2B3C4D:

  - 1) byte reverse:  
32-bit data is divided into 4 groups and reverse implement in group inside. Reversed data:0x58D43CB2
  - 2)half-word reverse:  
32-bit data is divided into 2 groups and reverse implement in group inside. Reversed

data: 0xD458B23C

3)word reverse:

32-bit data is divided into 1 groups and reverse implement in group inside. Reversed

data: 0xB23CD458

For output data, reverse type is word reverse.

For example: when REV\_O=1, calculation result 0x22CC4488 will be converted to 0x11223344.

- Multiple input data size support function will make users have the flexibility to adjust the combination of the calculation data.  
For example: 6 bytes input data can be combined by 1 word and 1 half-word, also it can be combined by 3 half-word.
- User configurable initial calculation data function will support user calculate CRC data value under any initial value.

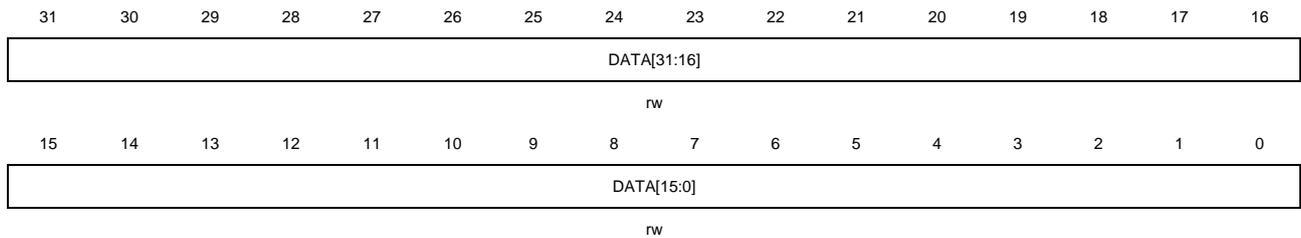
## 7.4. Register definition

### 7.4.1. Data Register (CRC\_DATA)

Address offset: 0x00

Reset value: 0xFFFF FFFF

This register has to be accessed by word(32-bit)



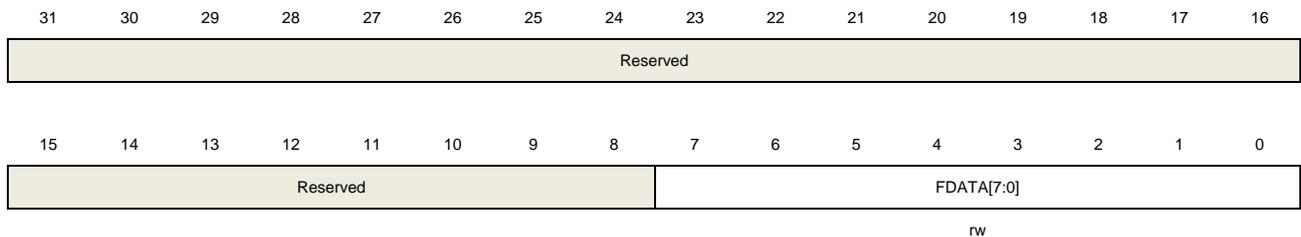
Bits	Fields	Descriptions
31:0	DATA[31:0]	CRC calculation result bits Software write and read. This register is used to calculate new data, and the register can be written the new data directly. Write value cannot be read because the read value is the CRC calculation result.

### 7.4.2. Free Data Register (CRC\_FDATA)

Address offset: 0x04

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)



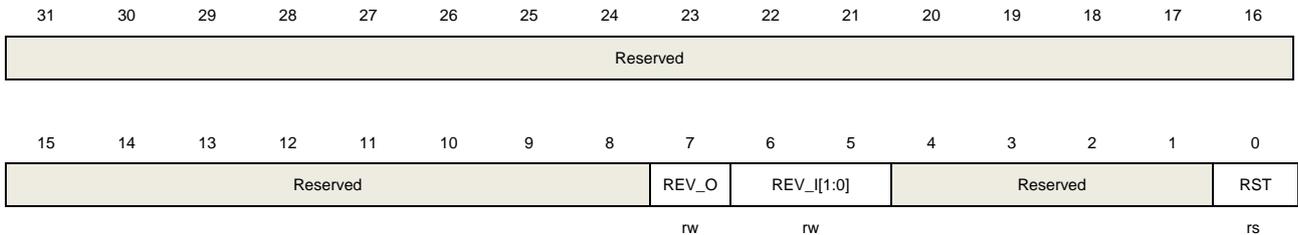
Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value
7:0	FDATA[7:0]	Free Data Register Bits Software write and read. These bits are unrelated with CRC calculation. This byte can be used for any goals by any other peripheral. The CRC_CTL register will generate no effect to the byte.

## 7.4.3. Control Register (CRC\_CTL)

Address offset: 0x08

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)



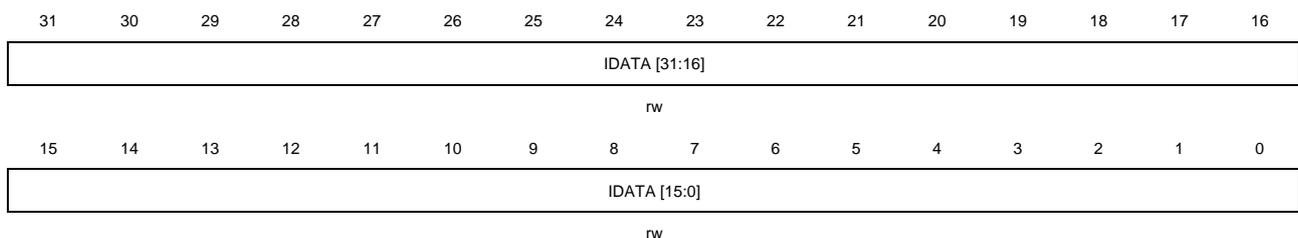
Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value
7	REV_O	Reverse output data value in bit order 0:Not bit reversed for output data 1:Bit reversed for output data
6:5	REV_I[1:0]	Reverse type for input data 0: Dot not use reverse for input data 1: Reverse input data with every 8-bit length 2: Reverse input data with every 16-bit length 3: Reverse input data with whole 32-bit length
4:1	Reserved	Must be kept at reset value
0	RST	This bit can reset the CRC_DATA register to the value in CRC_IDATA then automatically cleared itself to 0 by hardware. This bit will generate no effect to CRC_FDATA. Software write and read.

## 7.4.4. Initialization Data Register (CRC\_IDATA)

Address offset: 0x10

Reset value: 0xFFFF FFFF

This register has to be accessed by word(32-bit)



---

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:0	IDATA[31:0]	Configurable initial CRC data value When RST bit in CRC_CTL asserted, CRC_DATA will be programmed to this value.

## 8. Direct memory access controller (DMA)

### 8.1. Overview

The direct memory access (DMA) controller provides a hardware method of transferring data between peripherals and/or memory without intervention from the CPU, thereby freeing up bandwidth for other system functions. Data can be quickly moved by DMA between peripherals and memory as well as memory and memory without any CPU actions. There are 7 channels in the DMA controller. Each channel is dedicated to manage memory access requests from one or more peripherals. An arbiter is implemented inside to handle the priority among DMA requests.

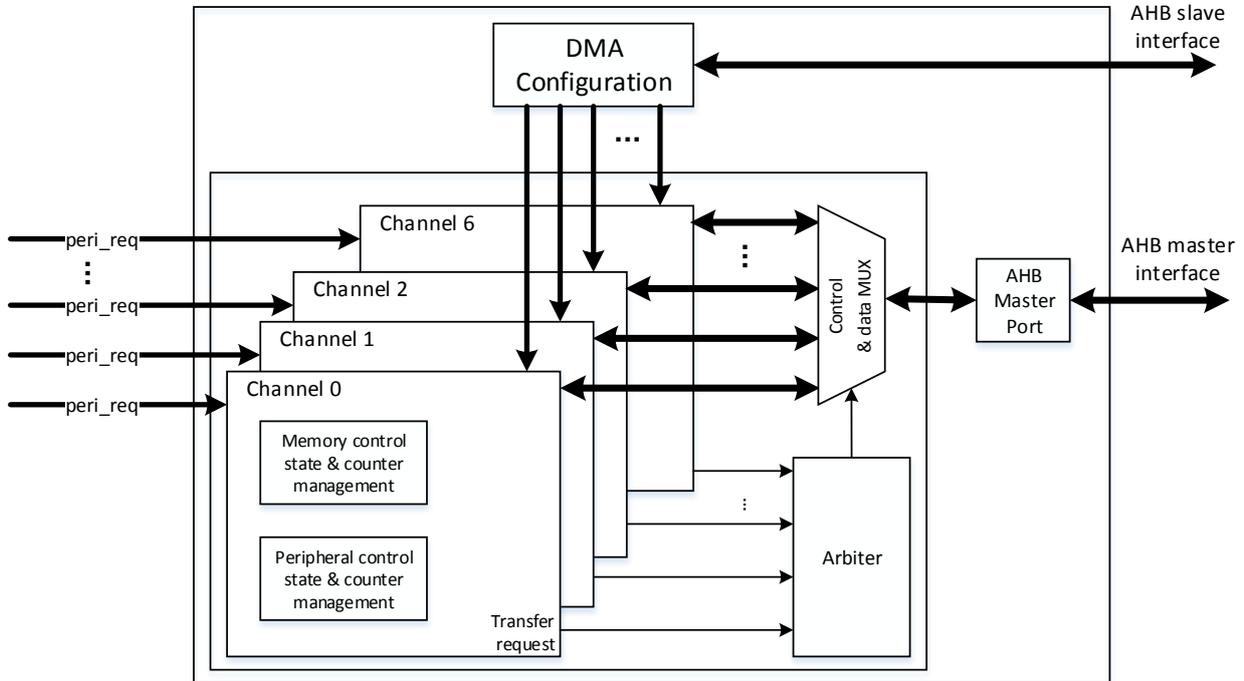
The system bus is shared by the DMA controller and the Cortex™-M3 core. When the DMA and the CPU are targeting the same destination, the DMA access may stop the CPU access to the system bus for some bus cycles. Round-robin scheduling is implemented in the bus matrix to ensure at least half of the system bus bandwidth for the CPU.

### 8.2. Characteristics

- Programmable length of data to be transferred, max to 65536
- 7 channels and each channel is configurable
- AHB and APB peripherals, FLASH, SRAM can be accessed as source and destination
- Each channel is connected to fixed hardware DMA request
- Software DMA channel priority (low, medium, high, ultra high) and hardware DMA channel priority (DMA channel 0 has the highest priority and DMA channel 6 has the lowest priority)
- Support independent 8, 16, 32-bit memory and peripheral transfer
- Support independent fixed and increasing address generation algorithm of memory and peripheral
- Support circular transfer mode
- Support peripheral to memory, memory to peripheral, and memory to memory transfers
- One separate interrupt per channel with three types of event flags
- Support interrupt enable and clear

### 8.3. Block diagram

Figure 8-1. Block diagram of DMA



As shown in [Figure 8-1. Block diagram of DMA](#), a DMA controller consists of four main parts:

- DMA configuration through AHB slave interface
- Data transmission through two AHB master interfaces for memory access and peripheral access
- An arbitrer inside to manage multiple peripheral requests coming at the same time
- Channel management to control address/data selection and data counting

### 8.4. Function overview

#### 8.4.1. DMA operation

Each DMA transfer consists of two operations, including the loading of data from the source and the storage of the loaded data to the destination. The source and destination addresses are computed by the DMA controller based on the programmed values in the DMA\_CHxPADDR, DMA\_CHxMADDR, and DMA\_CHxCTL registers. The DMA\_CHxCNT register controls how many transfers to be transmitted on the channel. The PWIDTH and MWIDTH bits in the DMA\_CHxCTL register determine how many bytes to be transmitted in a transfer.

Suppose DMA\_CHxCNT is 4, and both PNAGA and MNAGA are set. The DMA transfer operations for each combination of PWIDTH and MWIDTH are shown in the following [Table 8-1. DMA transfer operation](#).

**Table 8-1. DMA transfer operation**

Transfer size		Transfer operations	
Source	Destination	Source	Destination
32 bits	32 bits	1: Read B3B2B1B0[31:0] @0x0 2: Read B7B6B5B4[31:0] @0x4 3: Read BBBAB9B8[31:0] @0x8 4: Read BFBEBDBC[31:0] @0xC	1: Write B3B2B1B0[31:0] @0x0 2: Write B7B6B5B4[31:0] @0x4 3: Write BBBAB9B8[31:0] @0x8 4: Write BFBEBDBC[31:0] @0xC
32 bits	16 bits	1: Read B3B2B1B0[31:0] @0x0 2: Read B7B6B5B4[31:0] @0x4 3: Read BBBAB9B8[31:0] @0x8 4: Read BFBEBDBC[31:0] @0xC	1: Write B1B0[7:0] @0x0 2: Write B5B4[7:0] @0x2 3: Write B9B8[7:0] @0x4 4: Write BDBC[7:0] @0x6
32 bits	8 bits	1: Read B3B2B1B0[31:0] @0x0 2: Read B7B6B5B4[31:0] @0x4 3: Read BBBAB9B8[31:0] @0x8 4: Read BFBEBDBC[31:0] @0xC	1: Write B0[7:0] @0x0 2: Write B4[7:0] @0x1 3: Write B8[7:0] @0x2 4: Write BC[7:0] @0x3
16 bits	32 bits	1: Read B1B0[15:0] @0x0 2: Read B3B2[15:0] @0x2 3: Read B5B4[15:0] @0x4 4: Read B7B6[15:0] @0x6	1: Write 0000B1B0[31:0] @0x0 2: Write 0000B3B2[31:0] @0x4 3: Write 0000B5B4[31:0] @0x8 4: Write 0000B7B6[31:0] @0xC
16 bits	16 bits	1: Read B1B0[15:0] @0x0 2: Read B3B2[15:0] @0x2 3: Read B5B4[15:0] @0x4 4: Read B7B6[15:0] @0x6	1: Write B1B0[15:0] @0x0 2: Write B3B2[15:0] @0x2 3: Write B5B4[15:0] @0x4 4: Write B7B6[15:0] @0x6
16 bits	8 bits	1: Read B1B0[15:0] @0x0 2: Read B3B2[15:0] @0x2 3: Read B5B4[15:0] @0x4 4: Read B7B6[15:0] @0x6	1: Write B0[7:0] @0x0 2: Write B2[7:0] @0x1 3: Write B4[7:0] @0x2 4: Write B6[7:0] @0x3
8 bits	32 bits	1: Read B0[7:0] @0x0 2: Read B1[7:0] @0x1 3: Read B2[7:0] @0x2 4: Read B3[7:0] @0x3	1: Write 000000B0[31:0] @0x0 2: Write 000000B1[31:0] @0x4 3: Write 000000B2[31:0] @0x8 4: Write 000000B3[31:0] @0xC
8 bits	16 bits	1: Read B0[7:0] @0x0 2: Read B1[7:0] @0x1 3: Read B2[7:0] @0x2 4: Read B3[7:0] @0x3	1, Write 00B0[15:0] @0x0 2, Write 00B1[15:0] @0x2 3, Write 00B2[15:0] @0x4 4, Write 00B3[15:0] @0x6
8 bits	8 bits	1: Read B0[7:0] @0x0 2: Read B1[7:0] @0x1 3: Read B2[7:0] @0x2 4: Read B3[7:0] @0x3	1, Write B0[7:0] @0x0 2, Write B1[7:0] @0x1 3, Write B2[7:0] @0x2 4, Write B3[7:0] @0x3

The CNT bits in the DMA\_CHxCNT register control how many data to be transmitted on the channel and must be configured before enable the CHEN bit in the register. During the transmission, the CNT bits indicate the remaining number of data items to be transferred.

The DMA transmission is disabled by clearing the CHEN bit in the DMA\_CHxCTL register.

- If the DMA transmission is not completed when the CHEN bit is cleared, two situations may be occurred when restart this DMA channel:
  - If no register configuration operations of the channel occurs before restart the DMA channel, the DMA will continue to complete the rest of the transmission.
  - If any register configuration operations occur, the DMA will restart a new transmission.
- If the DMA transmission has been finished when clearing the CHEN bit, enable the DMA channel without any register configuration operation will not launch any DMA transfer.

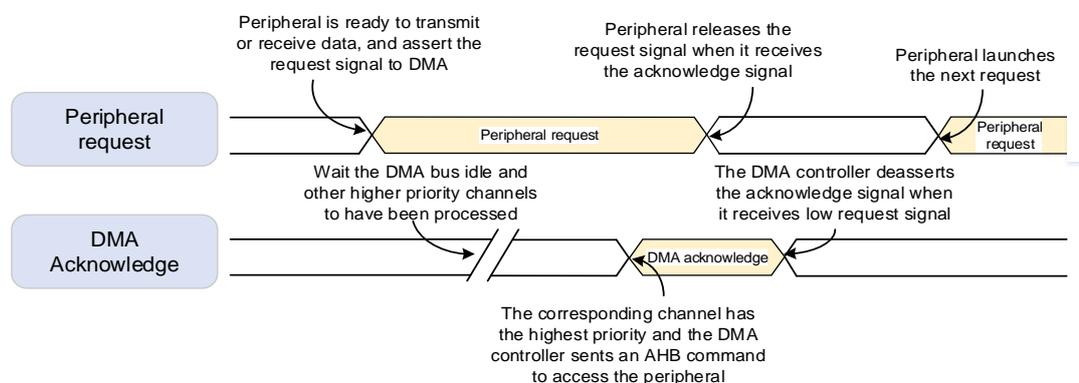
## 8.4.2. Peripheral handshake

To ensure a well-organized and efficient data transfer, a handshake mechanism is introduced between the DMA and peripherals, including a request signal and a acknowledge signal:

- Request signal asserted by peripheral to DMA controller, indicating that the peripheral is ready to transmit or receive data
- Acknowledge signal responded by DMA to peripheral, indicating that the DMA controller has initiated an AHB command to access the peripheral

[Figure 8-2. Handshake mechanism](#) shows how the handshake mechanism works between the DMA controller and peripherals.

**Figure 8-2. Handshake mechanism**



### 8.4.3. Arbitration

When two or more requests are received at the same time, the arbiter determines which request is served based on the priorities of channels. There are two-stage priorities, including the software priority and the hardware priority. The arbiter determines which channel is selected to respond according to the following priority rules:

- Software priority: Four levels, including low, medium, high and ultra-high by configuring the PRIO bits in the DMA\_CHxCTL register.
- For channels with equal software priority level, priority is given to the channel with lower channel number.

### 8.4.4. Address generation

Two kinds of address generation algorithm are implemented independently for memory and peripheral, including the fixed mode and the increased mode. The PNAGA and MNAGA bit in the DMA\_CHxCTL register are used to configure the next address generation algorithm of peripheral and memory.

In the fixed mode, the next address is always equal to the base address configured in the base address registers (DMA\_CHxPADDR, DMA\_CHxMADDR).

In the increasing mode, the next address is equal to the current address plus 1 or 2 or 4, depending on the transfer data width.

### 8.4.5. Circular mode

Circular mode is implemented to handle continue peripheral requests (for example, ADC scan mode). The circular mode is enabled by setting the CMEN bit in the DMA\_CHxCTL register.

In circular mode, the CNT bits are automatically reloaded with the pre-programmed value and the full transfer finish flag is asserted at the end of every DMA transfer. DMA can always responds the peripheral request until the CHEN bit in the DMA\_CHxCTL register is cleared.

### 8.4.6. Memory to memory mode

The memory to memory mode is enabled by setting the M2M bit in the DMA\_CHxCTL register. In this mode, the DMA channel can also work without being triggered by a request from a peripheral. The DMA channel starts transferring as soon as it is enabled by setting the CHEN bit in the DMA\_CHxCTL register, and completed when the DMA\_CHxCNT register reaches zero.

### 8.4.7. Channel configuration

When starting a new DMA transfer, it is recommended to respect the following steps:

1. Read the CHEN bit and judge whether the channel is enabled or not. If the channel is enabled, clear the CHEN bit by software. When the CHEN bit is read as '0', configuring and starting a new DMA transfer is allowed.
2. Configure the M2M bit and DIR bit in the DMA\_CHxCTL register to set the transfer mode.
3. Configure the CMEN bit in the DMA\_CHxCTL register to enable/disable the circular mode.
4. Configure the PRIO bits in the DMA\_CHxCTL register to set the channel software priority.
5. Configure the memory and peripheral transfer width, memory and peripheral address generation algorithm in the DMA\_CHxCTL register.
6. Configure the enable bit for full transfer finish interrupt, half transfer finish interrupt, transfer error interrupt in the DMA\_CHxCTL register.
7. Configure the DMA\_CHxPADDR register for setting the peripheral base address.
8. Configure the DMA\_CHxMADDR register for setting the memory base address.
9. Configure the DMA\_CHxCNT register to set the total transfer data number.
10. Configure the CHEN bit with '1' in the DMA\_CHxCTL register to enable the channel.

## 8.4.8. Interrupt

Each DMA channel has a dedicated interrupt. There are three types of interrupt event, including full transfer finish, half transfer finish, and transfer error.

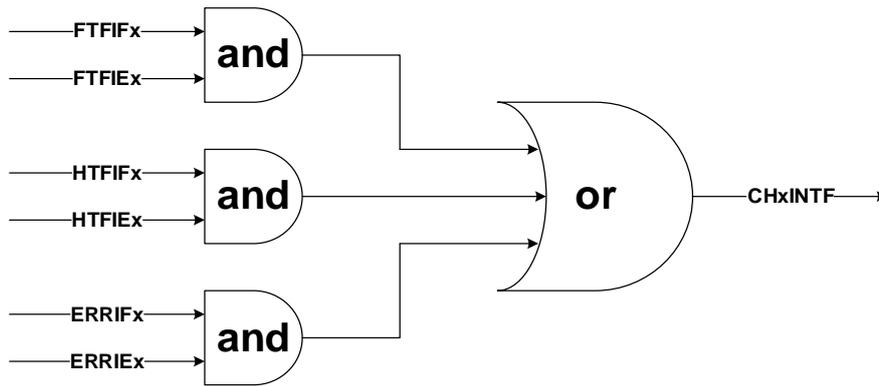
Each interrupt event has a dedicated flag bit in the DMA\_INTF register, a dedicated clear bit in the DMA\_INTC register, and a dedicated enable bit in the DMA\_CHxCTL register. The relationship is described in the following [Table 8-2. Interrupt events](#).

**Table 8-2. Interrupt events**

Interrupt event	Flag bit	Clear bit	Enable bit
	DMA_INTF	DMA_INTC	DMA_CHxCTL
Full transfer finish	FTFIF	FTFIFC	FTFIE
Half transfer finish	HTFIF	HTFIFC	HTFIE
Transfer error	ERRIF	ERRIFC	ERRIE

The DMA interrupt logic is shown in the [Figure 8-3. DMA interrupt logic](#), an interrupt can be produced when any type of interrupt event occurs and enabled on the channel.

Figure 8-3. DMA interrupt logic



NOTE: “x” indicates channel number (x=0..6).

### 8.4.9. DMA request mapping

Several requests from peripherals may be mapped to one DMA channel. They are logically ORed before entering the DMA. For details, see the following [Figure 8-4. DMA request mapping](#). The request of each peripheral can be independently enabled or disabled by programming the registers of the corresponding peripheral. The user has to ensure that only one request is enabled at a time on one channel. [Table 8-3. DMA requests for each channel](#) lists the support request from peripheral for each channel of DMA.

Figure 8-4. DMA request mapping

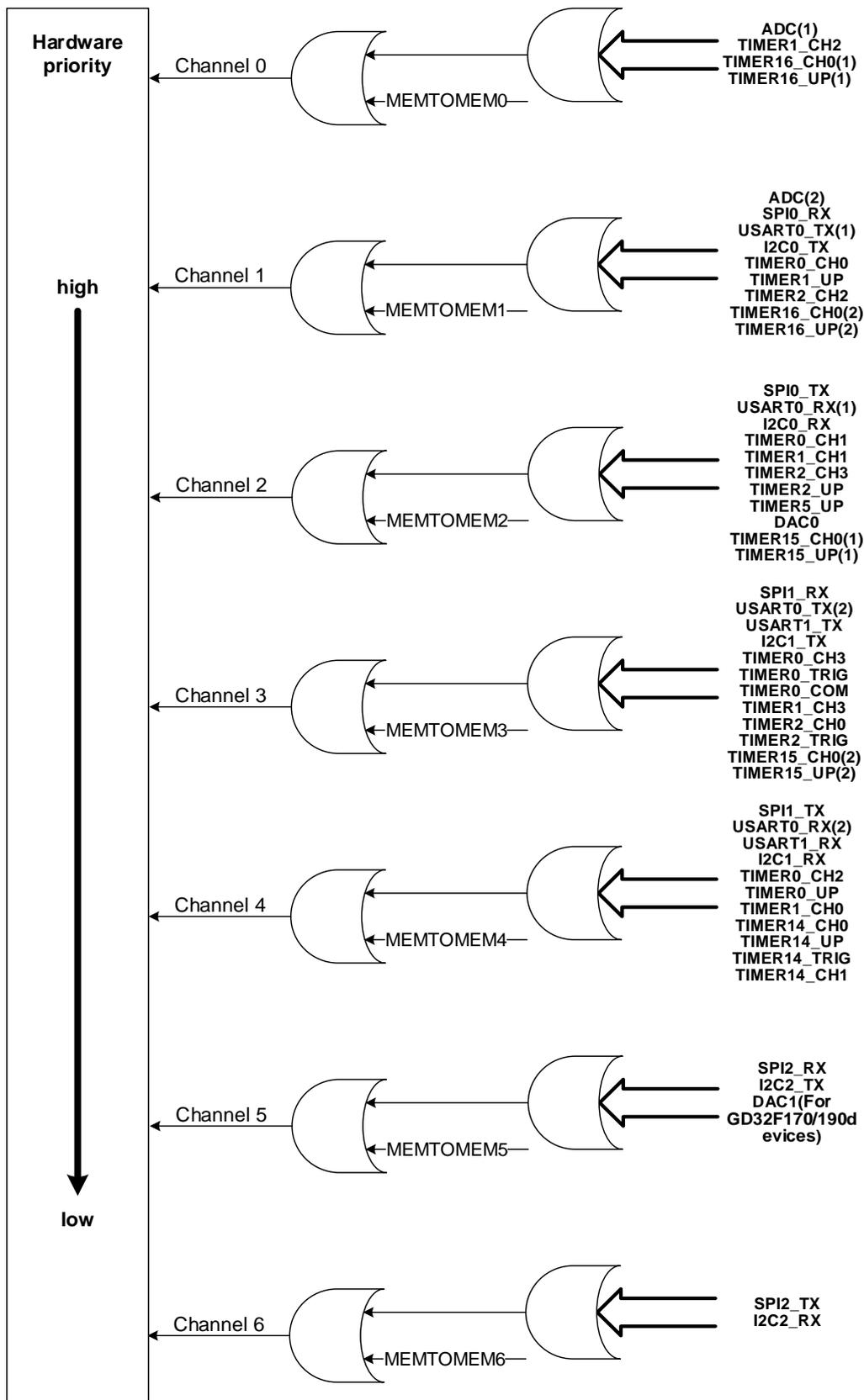


Table 8-3. DMA requests for each channel

Peripheral	Channel 0	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5	Channel 6
ADC	ADC(1)	ADC(2)	•	•	•	•	•
SPI/I2S	•	SPI/I2S0_RX	SPI/I2S0_TX	SPI1_RX	SPI1_TX	SPI2_RX	SPI2_TX
USART	•	USART0_TX(1)	USART0_RX(1)	USART0_TX(2) USART1_TX	USART0_RX(2) USART1_RX	•	•
I <sup>2</sup> C	•	I2C0_TX	I2C0_RX	I2C1_TX	I2C1_RX	I <sup>2</sup> C2_TX	I <sup>2</sup> C2_RX
TIMER0	•	TIMER0_CH0	TIMER0_CH1	TIMER0_CH3 TIMER0_TRIG TIMER0_COM	TIMER0_CH2 TIMER0_UP	•	•
TIMER1	TIMER1_CH2	TIMER1_UP	TIMER1_CH1	TIMER1_CH3	TIMER1_CH0	•	•
TIMER2	•	TIMER2_CH2	TIMER2_CH3 TIMER2_UP	TIMER2_CH0 TIMER2_TRIG	•	•	•
TIMER5/ DAC	•	•	TIMER5_UP DAC	•	•	DAC1 (For GD32F170/190 devices)	•
TIMER14	•	•	•	•	TIMER14_CH0 TIMER14_UP TIMER14_TRIG TIMER14_CH1	•	•
TIMER15	•	•	TIMER15_CH0(1) TIMER15_UP(1)	TIMER15_CH0(2) TIMER15_UP(2)	•	•	•
TIMER16	TIMER16_CH0(1) TIMER16_UP(1)	TIMER16_CH0(2) TIMER16_UP(2)	•	•	•	•	•

1. When the corresponding remapping bit in the SYSCFG\_CFGR0 register is cleared, the request is mapped on the channel.
2. When the corresponding remapping bit in the SYSCFG\_CFGR0 register is set, the request is mapped on the channel.

## 8.5. Register definition

### 8.5.1. Interrupt flag register (DMA\_INTF)

Address offset: 0x00

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				ERRIF6	HTFIF6	FTFIF6	GIF6	ERRIF5	HTFIF5	FTFIF5	GIF5	ERRIF4	HTFIF4	FTFIF4	GIF4
				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
ERRIF3	HTFIF3	FTFIF3	GIF3	ERRIF2	HTFIF2	FTFIF2	GIF2	ERRIF1	HTFIF1	FTFIF1	GIF1	ERRIF0	HTFIF0	FTFIF0	GIF0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits	Fields	Descriptions
31:28	Reserved	Must be kept at reset value
27/23/19/ 15/11/7/3	ERRIFx	Error flag of channel x (x=0...6) Hardware set and software cleared by configuring DMA_INTC register. 0: Transfer error has not occurred on channel x 1: Transfer error has occurred on channel x
26/22/18/ 14/10/6/2	HTFIFx	Half transfer finish flag of channel x (x=0...6) Hardware set and software cleared by configuring DMA_INTC register. 0: Half number of transfer has not finished on channel x 1: Half number of transfer has finished on channel x
25/21/17/ 13/9/5/1	FTFIFx	Full Transfer finish flag of channel x (x=0...6) Hardware set and software cleared by configuring DMA_INTC register. 0: Transfer has not finished on channel x 1: Transfer has finished on channel x
24/20/16/ 12/8/4/0	GIFx	Global interrupt flag of channel x (x=0...6) Hardware set and software cleared by configuring DMA_INTC register. 0: None of ERRIF, HTFIF or FTFIF occurs on channel x 1: At least one of ERRIF, HTFIF or FTFIF occurs on channel x

### 8.5.2. Interrupt flag clear register (DMA\_INTC)

Address offset: 0x04

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				ERRIFC6	HTFIFC6	FTFIFC6	GIFC6	ERRIFC5	HTFIFC5	FTFIFC5	GIFC5	ERRIFC4	HTFIFC4	FTFIFC4	GIFC4
				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
ERRIFC3	HTFIFC3	FTFIFC3	GIFC3	ERRIFC2	HTFIFC2	FTFIFC2	GIFC2	ERRIFC1	HTFIFC1	FTFIFC1	GIFC1	ERRIFC0	HTFIFC0	FTFIFC0	GIFC0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits	Fields	Descriptions
31:28	Reserved	Must be kept at reset value
27/23/19/ 15/11/7/3	ERRIFCx	Clear bit for error flag of channel x (x=0...6) 0: No effect 1: Clear error flag
26/22/18/ 14/10/6/2	HTFIFCx	Clear bit for half transfer finish flag of channel x (x=0...6) 0: No effect 1: Clear half transfer finish flag
25/21/17/ 13/9/5/1	FTFIFCx	Clear bit for full transfer finish flag of channel x (x=0...6) 0: No effect 1: Clear full transfer finish flag
24/20/16/ 12/8/4/0	GIFCx	Clear global interrupt flag of channel x (x=0...6) 0: No effect 1: Clear GIFx, ERRIFx, HTFIFx and FTFIFx bits in the DMA_INTF register

### 8.5.3. Channel x control register (DMA\_CHxCTL)

x = 0...6, where x is a channel number

Address offset: 0x08 + 0x14 × x

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

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	M2M	PRIO[1:0]		MWIDTH[1:0]		PWIDTH[1:0]		MNAGA	PNAGA	CMEN	DIR	ERRIE	HTFIE	FTFIE	CHEN
	rw	rw		rw		rw		rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:15	Reserved	Must be kept at reset value
14	M2M	Memory to Memory Mode Software set and cleared

		0: Disable Memory to Memory Mode 1: Enable Memory to Memory mode This bit can not be written when CHEN is '1'.
13:12	PRI0[1:0]	Priority level Software set and cleared 00: Low 01: Medium 10: High 11: Ultra high These bits can not be written when CHEN is '1'.
11:10	MWIDTH[1:0]	Transfer data size of memory Software set and cleared 00: 8-bit 01: 16-bit 10: 32-bit 11: Reserved These bits can not be written when CHEN is '1'.
9:8	PWIDTH[1:0]	Transfer data size of peripheral Software set and cleared 00: 8-bit 01: 16-bit 10: 32-bit 11: Reserved These bits can not be written when CHEN is '1'.
7	MNAGA	Next address generation algorithm of memory Software set and cleared 0: Fixed address mode 1: Increasing address mode This bit can not be written when CHEN is '1'.
6	PNAGA	Next address generation algorithm of peripheral Software set and cleared 0: Fixed address mode 1: Increasing address mode This bit can not be written when CHEN is '1'.
5	CMEN	Circular mode enable Software set and cleared 0: Disable circular mode 1: Enable circular mode This bit can not be written when CHEN is '1'.
4	DIR	Transfer direction

		Software set and cleared 0: Read from peripheral and write to memory 1: Read from memory and write to peripheral This bit can not be written when CHEN is '1'.
3	ERRIE	Enable bit for channel error interrupt Software set and cleared 0: Disable the channel error interrupt 1: Enable the channel error interrupt
2	HTFIE	Enable bit for channel half transfer finish interrupt Software set and cleared 0:Disable channel half transfer finish interrupt 1:Enable channel half transfer finish interrupt
1	FTFIE	Enable bit for channel full transfer finish interrupt Software set and cleared 0:Disable channel full transfer finish interrupt 1:Enable channel full transfer finish interrupt
0	CHEN	Channel enable Software set and cleared 0:Disable channel 1:Enable channel

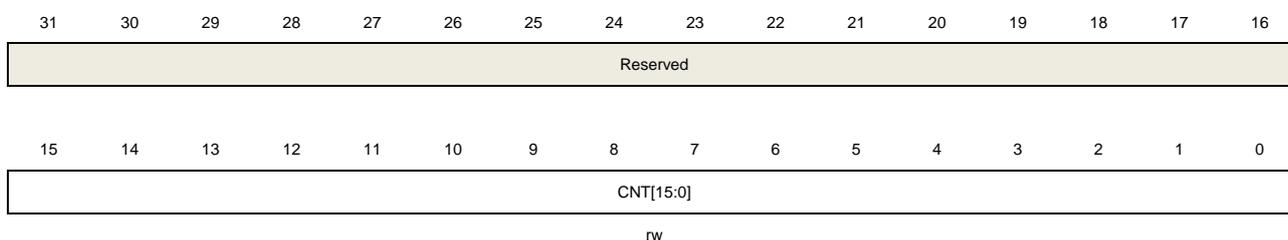
### 8.5.4. Channel x counter register (DMA\_CHxCNT)

x = 0...6, where x is a channel number

Address offset: 0x0C + 0x14 × x

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)



Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	CNT[15:0]	Transfer counter These bits can not be written when CHEN in the DMA_CHxCTL register is '1'.

This register indicates how many transfers remain. Once the channel is enabled, it is read-only, and decreases after each DMA transfer. If the register is zero, no transaction can be issued whether the channel is enabled or not. Once the transmission of the channel is complete, the register can be reloaded automatically by the previously programmed value if the channel is configured in circular mode.

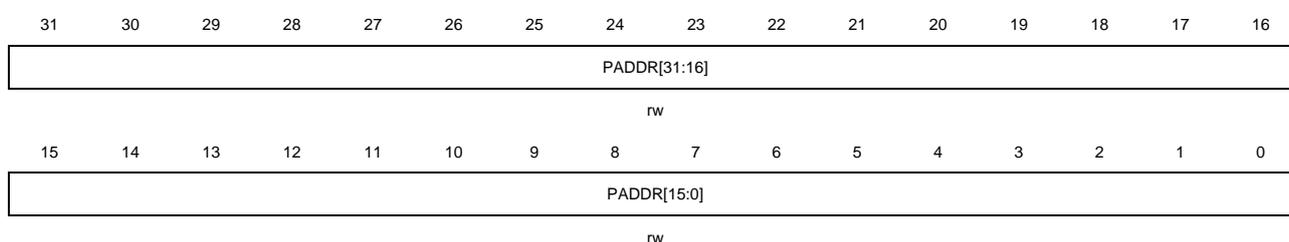
## 8.5.5. Channel x peripheral base address register (DMA\_CHxPADDR)

$x = 0..6$ , where  $x$  is a channel number

Address offset:  $0x10 + 0x14 \times x$

Reset value:  $0x0000\ 0000$

This register has to be accessed by word(32-bit)



Bits	Fields	Descriptions
31:0	PADDR[31:0]	Peripheral base address These bits can not be written when CHEN in the DMA_CHxCTL register is '1'. When PWIDTH is 01 (16-bit), the LSB of these bits is ignored. Access is automatically aligned to a half word address. When PWIDTH is 10 (32-bit), the two LSBs of these bits are ignored. Access is automatically aligned to a word address.

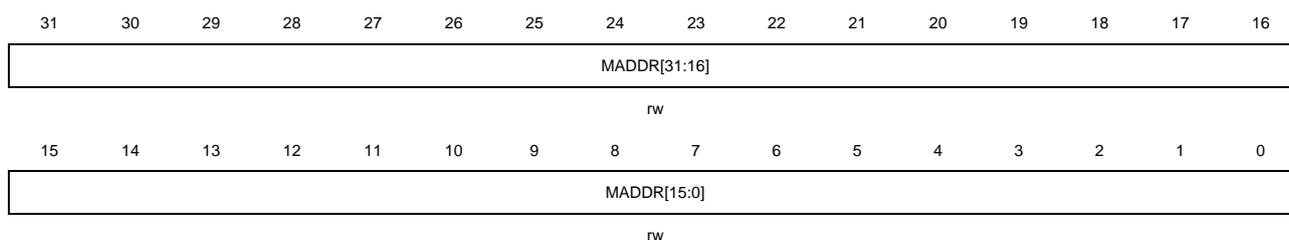
## 8.5.6. Channel x memory base address register (DMA\_CHxMADDR)

$x = 0..6$ , where  $x$  is a channel number

Address offset:  $0x14 + 0x14 \times x$

Reset value:  $0x0000\ 0000$

This register has to be accessed by word(32-bit)



Bits	Fields	Descriptions
31:0	MADDR[31:0]	<p>Memory base address</p> <p>These bits can not be written when CHEN in the DMA_CHxCTL register is '1'.</p> <p>When MWIDTH in the DMA_CHxCTL register is 01 (16-bit), the LSB of these bits is ignored. Access is automatically aligned to a half word address.</p> <p>When MWIDTH in the DMA_CHxCTL register is 10 (32-bit), the two LSBs of these bits are ignored. Access is automatically aligned to a word address.</p>

## 9. Debug (DBG)

### 9.1. Introduction

The GD32F1x0 series provide a large variety of debug, trace and test features. They are implemented with a standard configuration of the ARM CoreSight™ module together with a daisy chained standard TAP controller. Debug and trace functions are integrated into the ARM Cortex-M3. The debug system supports serial wire debug (SWD) and trace functions. The debug and trace functions refer to the following documents:

- Cortex-M3 Technical Reference Manual
- ARM Debug Interface v5 Architecture Specification

The DBG hold unit helps debugger to debug power saving mode, TIMER, I2C, RTC, WWDGT, FWDGT and CAN. When corresponding bit is set, provide clock when in power saving mode or hold the state for TIMER, I2C, RTC, WWDGT, FWDGT and CAN. (The contents of CAN is only for GD32F170xx and GD32F190xx devices)

### 9.2. Serial Wire Debug port introduction

Debug capabilities can be accessed by a debug tool via Serial Wire (SW - Debug Port).

#### 9.2.1. Pin assignment

The synchronous serial wire debug (SWD) provide 2-pin SW interface, known as SW data input/output (SWDIO) and SW clock (SWCLK).

The pin assignment are:

PA14 : SWCLK

PA13 : SWDIO

If SWD not used, all 2-pin can be released to other GPIO functions. Please refer to [GPIO pin configuration](#)

#### 9.2.2. JEDEC-106 ID code

The Cortex-M3 integrates JEDEC-106 ID code, which is located in ROM table and mapped on the address of 0xE00FF000\_0xE00FFFFF.

## 9.3. Debug hold function description

### 9.3.1. Debug support for power saving mode

When STB\_HOLD bit in DBG control register 0 (DBG\_CTL0) is set and entering the standby mode, the clock of AHB bus and system clock are provided by CK\_IRC8M, and the debugger can debug in standby mode. When exit the standby mode, a system reset generated.

When DSPLP\_HOLD bit in DBG control register 0 (DBG\_CTL0) is set and entering the Deep-sleep mode, the clock of AHB bus and system clock are provided by CK\_IRC8M, and the debugger can debug in Deep-sleep mode.

When SLP\_HOLD bit in DBG control register 0 (DBG\_CTL0) is set and entering the sleep mode, the clock of AHB bus for CPU is not closed, and the debugger can debug in sleep mode.

### 9.3.2. Debug support for TIMER, I2C, RTC, WWDGT and FWDGT

When the core halted and the corresponding bit in DBG control register 0 or 1 (DBG\_CTL0 or DBG\_CTL1) is set, the following behaved.

For TIMER, the timer counters stopped and hold for debug.

For I2C, SMBUS timeout hold for debug.

For RTC, the counter stopped for debug.

For WWDGT or FWDGT, the counter clock stopped for debug.

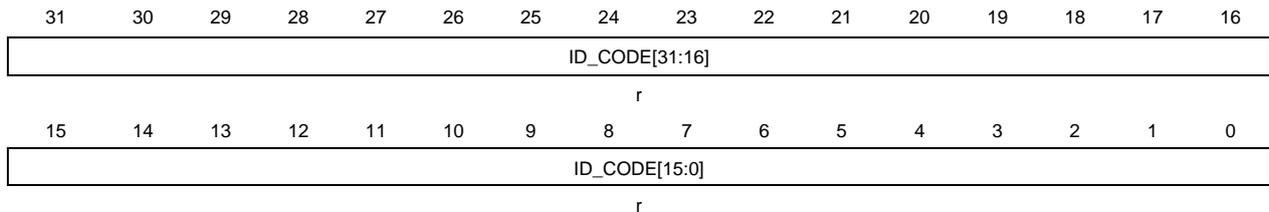
For CAN, the receive register stopped counting for debug.

## 9.4. DBG registers

### 9.4.1. ID code register (DBG\_ID)

Address: 0xE004 2000

This register has to be accessed by word(32-bit)



Bits	Fields	Descriptions
31:0	ID_CODE[31:0]	DBG ID code register These bits can be read by software, These bits are unchanged constant.

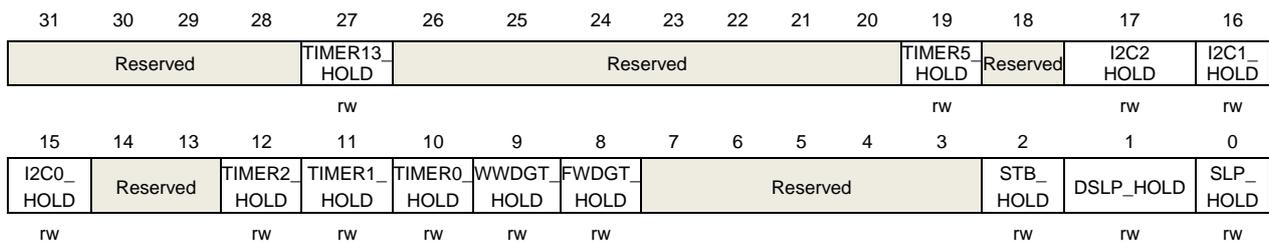
### 9.4.2. Control register 0(DBG\_CTL0)

**For GD32F130xx and GD32F150xx devices**

Address offset: 0x04

Reset value: 0x0000 0000; power reset only

This register has to be accessed by word(32-bit)



Bits	Fields	Descriptions
31:28	Reserved	Must be kept at reset value
27	TIMER13_HOLD	TIMER13 hold register This bit is set and reset by software. 0: No effect 1: Hold the TIMER13 counter for debug when core halted
26:20	Reserved	Must be kept at reset value
19	TIMER5_HOLD	TIMER5 hold register This bit is set and reset by software.

		0: No effect 1: Hold the TIMER5 counter for debug when core halted
18	Reserved	Must be kept at reset value
17	I2C2_HOLD	I2C2 hold register This bit is set and reset by software. 0: No effect 1: Hold the I2C2 SMBUS timeout for debug when core halted
16	I2C1_HOLD	I2C1 hold register This bit is set and reset by software. 0: No effect 1: Hold the I2C1 SMBUS timeout for debug when core halted
15	I2C0_HOLD	I2C0 hold register This bit is set and reset by software. 0: No effect 1: Hold the I2C0 SMBUS timeout for debug when core halted
14:13	Reserved	Must be kept at reset value
12	TIMER2_HOLD	TIMER2 hold register This bit is set and reset by software. 0: No effect 1: Hold the TIMER2 counter for debug when core halted
11	TIMER1_HOLD	TIMER1 hold register This bit is set and reset by software. 0: No effect 1: Hold the TIMER1 counter for debug when core halted
10	TIMER0_HOLD	TIMER0 hold register This bit is set and reset by software. 0: No effect 1: Hold the TIMER0 counter for debug when core halted
9	WWDGT_HOLD	WWDGT hold register This bit is set and reset by software. 0: No effect 1: Hold the WWDGT counter clock for debug when core halted
8	FWDGT_HOLD	FWDGT hold register This bit is set and reset by software. 0: No effect 1: Hold the FWDGT counter clock for debug when core halted
2	STB_HOLD	Standby mode hold register This bit is set and reset by software.

- 0: No effect
  - 1: At the standby mode, the system clock and HCLK are provided by CK\_IRC8M, a system reset generated when exiting standby mode
- 1      DSLP\_HOLD      Deep-sleep mode hold register  
This bit is set and reset by software.
- 0: No effect
  - 1: At the Deep-sleep mode, the system clock and HCLK are provided by CK\_IRC8M
- 0      SLP\_HOLD      Sleep mode hold register  
This bit is set and reset by software.
- 0: No effect
  - 1: At the sleep mode, the HCLK is on

### For GD32F170xx and GD32F190xx devices

Address offset: 0x04

Reset value: 0x0000 0000; power reset only

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Reserved				TIMER13_HOLD	Reserved						CAN1_HOLD	Reserved	TIMER5_HOLD	Reserved	I2C2_HOLD	I2C1_HOLD
				rw							rw		rw		rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
I2C0_HOLD	CAN0_HOLD	Reserved	TIMER2_HOLD	TIMER1_HOLD	TIMER0_HOLD	WWDGT_HOLD	FWDGT_HOLD	Reserved					STB_HOLD	DSLP_HOLD	SLP_HOLD	
rw	rw		rw	rw	rw	rw	rw						rw	rw	rw	

Bits	Fields	Descriptions
31:28	Reserved	Must be kept at reset value
27	TIMER13_HOLD	TIMER13 hold register This bit is set and reset by software. 0: No effect 1: Hold the TIMER13 counter for debug when core halted
26:22	Reserved	Must be kept at reset value
21	CAN1_HOLD	CAN1 hold register This bit is set and reset by software. 0: No effect 1: Hold the CAN1 for debug when core halted
20	Reserved	Must be kept at reset value
19	TIMER5_HOLD	TIMER5 hold register This bit is set and reset by software.

		0: No effect 1: Hold the TIMER5 counter for debug when core halted
18	Reserved	Must be kept at reset value
17	I2C2_HOLD	I2C2 hold register This bit is set and reset by software. 0: No effect 1: Hold the I2C2 SMBUS timeout for debug when core halted
16	I2C1_HOLD	I2C1 hold register This bit is set and reset by software. 0: No effect 1: Hold the I2C1 SMBUS timeout for debug when core halted
15	I2C0_HOLD	I2C0 hold register This bit is set and reset by software. 0: No effect 1: Hold the I2C0 SMBUS timeout for debug when core halted
14	CAN0_HOLD	CAN0 hold register This bit is set and reset by software. 0: No effect 1: Hold the CAN0 for debug when core halted
13	Reserved	Must be kept at reset value
12	TIMER2_HOLD	TIMER2 hold register This bit is set and reset by software. 0: No effect 1: Hold the TIMER2 counter for debug when core halted
11	TIMER1_HOLD	TIMER1 hold register This bit is set and reset by software. 0: No effect 1: Hold the TIMER1 counter for debug when core halted
10	TIMER0_HOLD	TIMER0 hold register This bit is set and reset by software. 0: No effect 1: Hold the TIMER0 counter for debug when core halted
9	WWDGT_HOLD	WWDGT hold register This bit is set and reset by software. 0: No effect 1: Hold the WWDGT counter clock for debug when core halted
8	FWDGT_HOLD	FWDGT hold register This bit is set and reset by software.

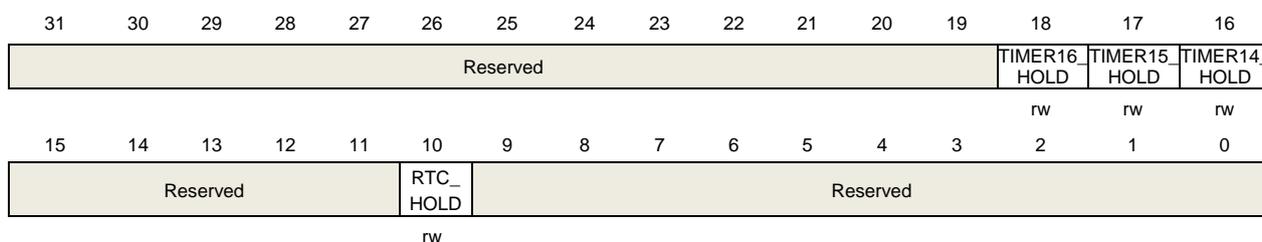
		0: No effect 1: Hold the FWDGT counter clock for debug when core halted
2	STB_HOLD	Standby mode hold register This bit is set and reset by software. 0: No effect 1: At the standby mode, the system clock and HCLK are provided by CK_IRC8M, a system reset generated when exit stanby mode
1	DSLP_HOLD	Deep-sleep mode hold register This bit is set and reset by software. 0: No effect 1: At the Deep-sleep mode, the system clock and HCLK are provided by CK_IRC8M
0	SLP_HOLD	Sleep mode hold register This bit is set and reset by software. 0: No effect 1: At the sleep mode, the HCLK is on

### 9.4.3. Control register 1 (DBG\_CTL1)

Address offset: 0x08

Reset value: 0x0000 0000; power reset only

This register has to be accessed by word(32-bit)



Bits	Fields	Descriptions
31:19	Reserved	Must be kept at reset value
18	TIMER16_HOLD	TIMER16 hold register This bit is set and reset by software. 0: No effect 1: Hold the TIMER16 counter for debug when core halted
17	TIMER15_HOLD	TIMER15 hold register This bit is set and reset by software. 0: No effect 1: Hold the TIMER15 counter for debug when core halted

16	TIMER14_HOLD	TIMER14 hold register This bit is set and reset by software. 0: No effect 1: Hold the TIMER14 counter for debug when core halted
15:11	Reserved	Must be kept at reset value
10	RTC_HOLD	RTC hold register This bit is set and reset by software. 0: No effect 1: Hold the RTC counter for debug when core halted
9:0	Reserved	Must be kept at reset value

## 10. Analog to digital converter (ADC)

### 10.1. Overview

The 12-bit ADC is an analog-to-digital converter using successive approximation approach. It has 19 multiplexed channels which are used to measure the signals from 16 external channels, 2 internal channels and the battery voltage ( $V_{BAT}$ ) channel. Analog watchdog allows the application to detect whether the input voltage exceeds the user's set of high and low threshold. The A/D conversion of each channel can be performed in single, continuous, scan, or discontinuous mode. A left-aligned or right-aligned 16-bit data register holds the output of the ADC. An on-chip hardware oversample scheme improves performances while off-loading the related computational burden from the MCU (for GD32F170xx and GD32F190xx devices).

### 10.2. Characteristics

**For GD32F130xx and GD32F150xx devices:**

- High performance:
  - 12-bit resolution
  - ADC conversion time: 1.0 us (1MS/s)
  - Self-calibration time: 83 ADC clock cycles
  - Programmable sampling time
  - Configurable data alignment in data registers
  - DMA request for regular channels data transfer
- Dual clock domain architecture (APB clock and ADC clock)
- Analog input channels:
  - 16 external analog inputs
  - 1 channel for internal temperature sensor ( $V_{SENSE}$ )
  - 1 channel for internal reference voltage ( $V_{REFINT}$ )
  - 1 channel for monitoring external  $V_{BAT}$  power supply pin
- Start of the conversion can be initiated:
  - By software
  - By hardware triggers with configurable polarity (internal timer events from TIMER0, TIMER1, TIMER2 and TIMER14)

- External trigger on regular and inserted channels
- Conversion modes:
  - Can convert a single channel or can scan a sequence of channels.
  - Single mode converts selected inputs once per trigger
  - Continuous mode converts selected inputs continuously
  - Discontinuous mode
- Interrupt generation at the end of regular and inserted group conversions, and in case of analog watchdog events
- Analog watchdog
- ADC supply requirements: from 2.6V to 3.6V, and typical power supply voltage is 3.3V.
- ADC input range:  $V_{SSA} \leq V_{IN} \leq V_{DDA}$

### For GD32F170xx and GD32F190xx devices:

- High performance:
  - 12-bit, 10-bit, 8-bit, or 6-bit configurable resolution
  - ADC conversion time: 0.5 $\mu$ s for 12-bit resolution (2MS/s), about 0.43 $\mu$ s conversion time for 10-bit resolution, about 0.286 $\mu$ s for 6-bit resolution. Faster conversion time can be obtained by lowering the resolution
  - Self-calibration time: 84 ADC clock cycles
  - Programmable sampling time
  - Configurable data alignment in data registers
  - DMA request for regular data transfer
- Dual clock domain architecture (APB clock and ADC clock)
- Analog input channels:
  - 16 external analog inputs
  - 1 channel for internal temperature sensor ( $V_{SENSE}$ )
  - 1 channel for internal reference voltage ( $V_{REFINT}$ )
  - 1 channel for monitoring external  $V_{BAT}$  power supply pin
- Start of the conversion can be initiated:
  - By software
  - By hardware triggers with configurable polarity (internal timer events from TIMER0,

TIMER1, TIMER2 and TIMER14)

- External trigger on regular and inserted channels
- Conversion modes:
  - Can convert a single channel or can scan a sequence of channels.
  - Single mode converts selected inputs once per trigger
  - Continuous mode converts selected inputs continuously
  - Discontinuous mode
- Interrupt generation at the end of regular and inserted group conversions, and in case of analog watchdog events
- Analog watchdog
- ADC supply requirements: from 3V to 5.5V, and typical power supply voltage is 5V.
- Oversampling:
  - 16-bit data register
  - Oversampling ratio adjustable from 2 to 256x
  - Programmable data shift up to 8-bits
- ADC input range:  $V_{SSA} \leq V_{IN} \leq V_{DDA}$

## 10.3. Pins and internal signals

[Figure 10-1. ADC module block diagram of GD32F130xx and GD32F150xx devices](#) and [Figure 10-2. ADC module block diagram of GD32F170xx and GD32F190xx devices](#) show the ADC block diagram. [Table 10-1. ADC internal signals](#), [Table 10-2. ADC pins definition of GD32F130xx and GD32F150xx devices](#) and [Table 10-3. ADC pins definition of GD32F170xx and GD32F190xx devices](#) give the ADC internal signals and pins description.

**Table 10-1. ADC internal signals**

Internal signal name	Signal type	Description
V <sub>SENSE</sub>	Input	Internal temperature sensor output voltage
V <sub>REFINT</sub>	Input	Internal voltage reference output voltage
V <sub>BAT/2</sub>	Input	V <sub>BAT</sub> pin input voltage divided by 2

**Table 10-2. ADC pins definition of GD32F130xx and GD32F150xx devices**

Name	Signal type	Remarks
V <sub>DDA</sub> <sup>(1)</sup>	Input, analog supply	Analog power supply equal to V <sub>DD</sub> and 2.6

		$V \leq V_{DDA} \leq 3.6 V$
$V_{SSA}^{(1)}$	Input, analog supply ground	Ground for analog power supply equal to $V_{SS}$
ADCx_IN [15:0]	Input, Analog signals	Up to 16 analog channels

**Table 10-3. ADC pins definition of GD32F170xx and GD32F190xx devices**

Name	Signal type	Remarks
$V_{DDA}^{(1)}$	Input, analog supply	Analog power supply equal to $V_{DD}$ , and $3 V \leq V_{DDA} \leq 5.5 V$
$V_{SSA}^{(1)}$	Input, analog supply ground	Ground for analog power supply equal to $V_{SS}$
ADCx_IN [15:0]	Input, Analog signals	Up to 16 analog channels

1.  $V_{DDA}$  and  $V_{SSA}$  have to be connected to  $V_{DD}$  and  $V_{SS}$ , respectively.

## 10.4. Function overview

**Figure 10-1. ADC module block diagram of GD32F130xx and GD32F150xx devices**

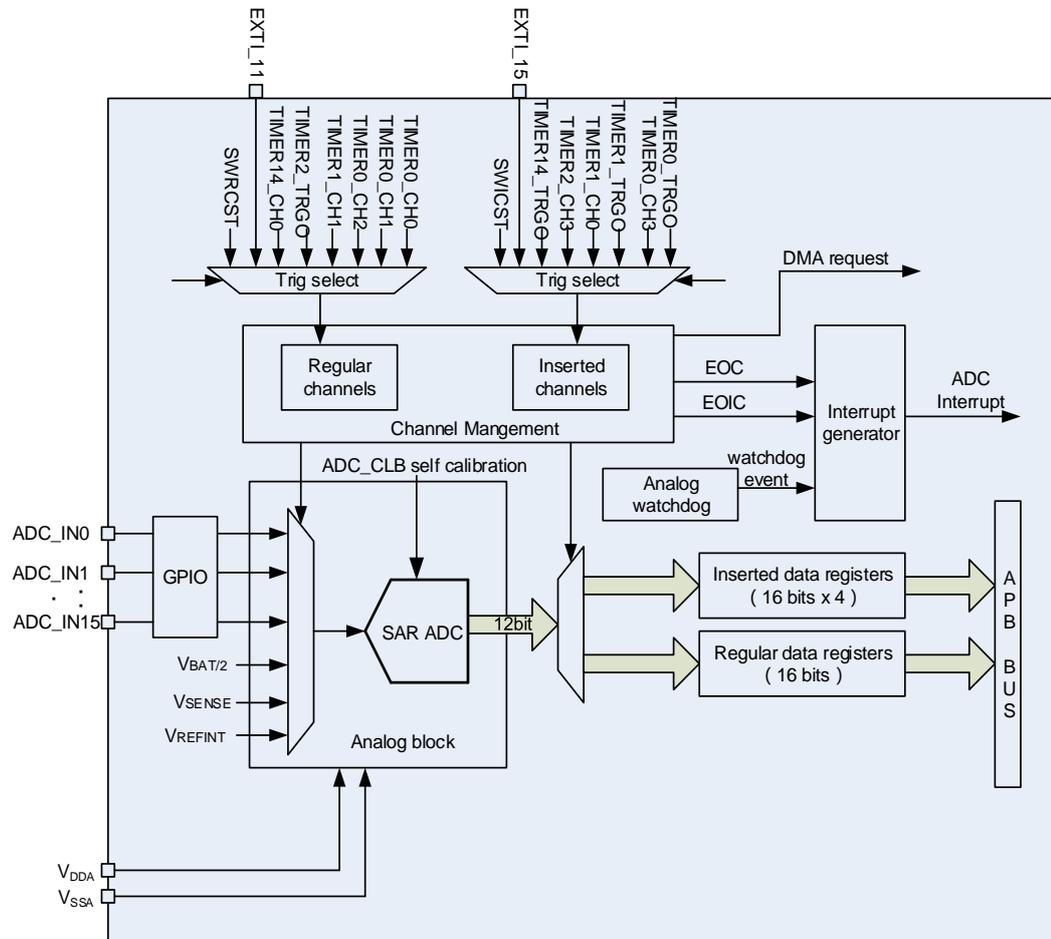
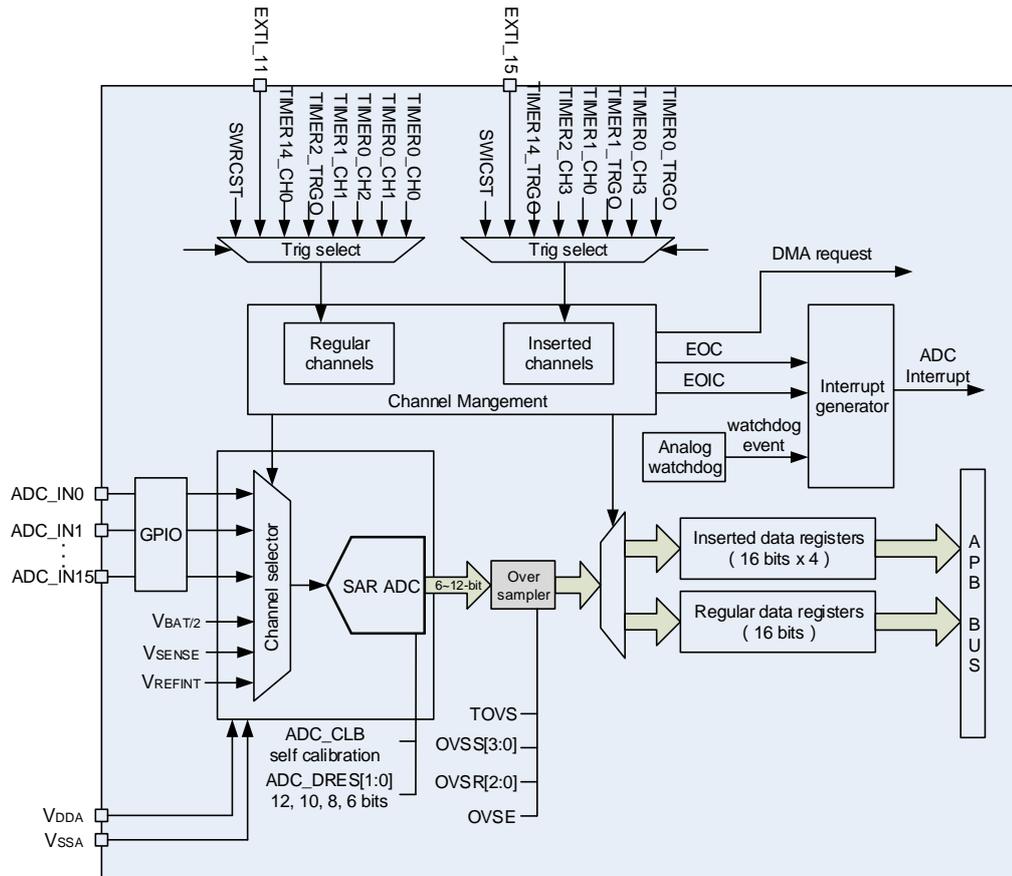


Figure 10-2. ADC module block diagram of GD32F170xx and GD32F190xx devices



### 10.4.1. Calibration (ADC\_CLB)

The ADC has a calibration feature. During the procedure, the ADC calculates a calibration factor which is internally applied to the ADC until the next ADC power-off. The application must not use the ADC during calibration and must wait until it is completed. Calibration should be performed before starting A/D conversion. For GD32F130xx and GD32F150xx devices, the calibration needs 83 clock periods to remove the comparator offset error and capacitor mismatch error which may vary from chip to chip due to process variation. But it needs 84 clock periods for GD32F170xx and GD32F190xx devices.

The calibration is initiated by software by setting bit ADC\_CLB=1. ADC\_CLB bit stays at 1 during all the calibration sequence. It is then cleared by hardware as soon as the calibration completes.

When the ADC's operating conditions change ( $V_{DDA}$  changes are the main contributor to ADC offset variations and temperature changes are the secondary contributor), it is recommended to re-run a calibration cycle.

The internal analog calibration can be reset by setting the RSTCLB bit in ADC\_CTL1 register.

Calibration software procedure:

1. Ensure that ADCON=1
2. Delay 14 ADCCLK to wait for ADC stability
3. Set RSTCLB (optional)
4. Set CLB=1
5. Wait until CLB =0

#### 10.4.2. Dual clock domain architecture

The ADC sub-module, with exception of the APB interface block, is feed by an ADC clock, which can be asynchronous and independent from the APB clock.

Application can reduce PLCK frequency for low power operation while still keeping optimum ADC performance.

Refer to [RCU Section 4.2.1](#) for more information on generating this clock source.

#### 10.4.3. ADCON switch

The ADCON bit in the ADC\_CTL1 register is the enable switch of the ADC module. The ADC module will keep in reset state if this bit is 0. For power saving, when this bit is reset, the analog submodule will be put into power down mode

#### 10.4.4. Regular and inserted channel groups

The ADC supports 19 multiplexed channels and organizes the conversion results into two groups: a regular channel group and an inserted channel group.

In the regular group, a sequence of up to 16 conversions can be organized in a specific sequence. The ADC\_RSQ0~ADC\_RSQ2 registers specify the selected channels of the regular group. The RL[3:0] bits in the ADC\_RSQ0 register specify the total conversion sequence length.

In the inserted group, a sequence of up to 4 conversions can be organized in a specific sequence. The ADC\_ISQ register specifies the selected channels of the inserted group. The IL[1:0] bits in the ADC\_ISQ register specify the total conversion sequence length.

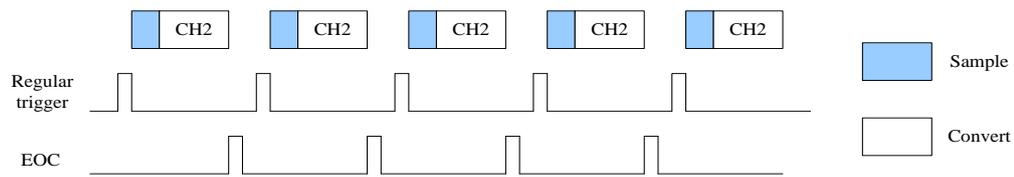
#### 10.4.5. Conversion modes

##### Single conversion mode

This mode can be running on both regular and inserted channel group. In the single conversion mode, the ADC performs conversion on the channel specified in the RSQ0[4:0] bits of ADC\_RSQ2 at a regular trigger or the channel specified in the ISQ3[4:0] bits of ADC\_ISQ. When the ADCON has been set high, the ADC samples and converts a single

channel, once the corresponding software trigger or external trigger is active.

**Figure 10-3. Single conversion mode**



After conversion of a single regular channel, the conversion data will be stored in the ADC\_RDATA register, the EOC will be set. An interrupt will be generated if the EOCIE bit is set.

After conversion of a single injected channel, the conversion data will be stored in the ADC\_IDATA0 register, the EOC and EOIC will be set. An interrupt will be generated if the EOCIE or EOICIE bit is set.

Software procedure for a single conversion of a regular channel:

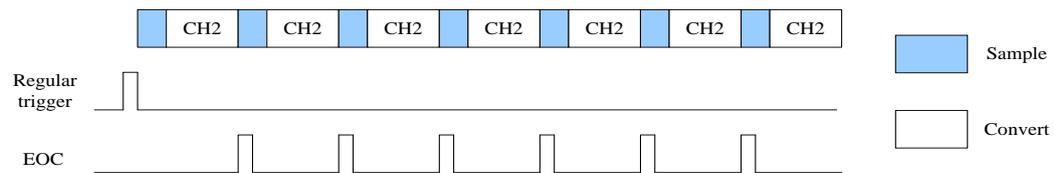
1. Make sure the DISRC, SM in the ADC\_CTL0 register and CTN bit in the ADC\_CTL1 register are reset
2. Configure RSQ0 with the analog channel number
3. Configure ADC\_SAMPTx register
4. Configure ETERC and ETSRC bits in the ADC\_CTL1 register if in need
5. Set the SWRCST bit, or generate an external trigger for the regular group
6. Wait the EOC flag to be set
7. Read the converted result in the ADC\_RDATA register
8. Clear the EOC flag by writing 0 to it

Software procedure for a single conversion of an inserted channel:

1. Make sure the DISIC, SM in the ADC\_CTL0 register are reset
2. Configure ISQ3 with the analog channel number
3. Configure ADC\_SAMPTx register
4. Configure ETEIC and ETSIC bits in the ADC\_CTL1 register if in need
5. Set the SWICST bit, or generate an external trigger for the inserted group
6. Wait the EOC/EOIC flags to be set
7. Read the converted result in the ADC\_IDATA0 register
8. Clear the EOC/EOIC flags by writing 0 to them

### Continuous conversion mode

This mode can be run on the regular channel group. The continuous conversion mode will be enabled when CTN bit in the ADC\_CTL1 register is set. In this mode, the ADC performs conversion on the channel specified in the RSQ0[4:0]. When the ADCON has been set high, the ADC samples and converts specified channel, once the corresponding software trigger or external trigger is active. The conversion data will be stored in the ADC\_RDATA register.

**Figure 10-4. Continuous conversion mode, scan disable**


Software procedure for continuous conversion on a regular channel:

1. Set the CTN bit in the ADC\_CTL1 register
2. Configure RSQ0 with the analog channel number
3. Configure ADC\_SAMPTx register
4. Configure ETERC and ETSRC bits in the ADC\_CTL1 register if in need
5. Set the SWRCST bit, or generate an external trigger for the regular group
6. Wait the EOC flag to be set
7. Read the converted result in the ADC\_RDATA register
8. Clear the EOC flag by writing 0 to it
9. Repeat steps 6~8 as soon as the conversion is in need

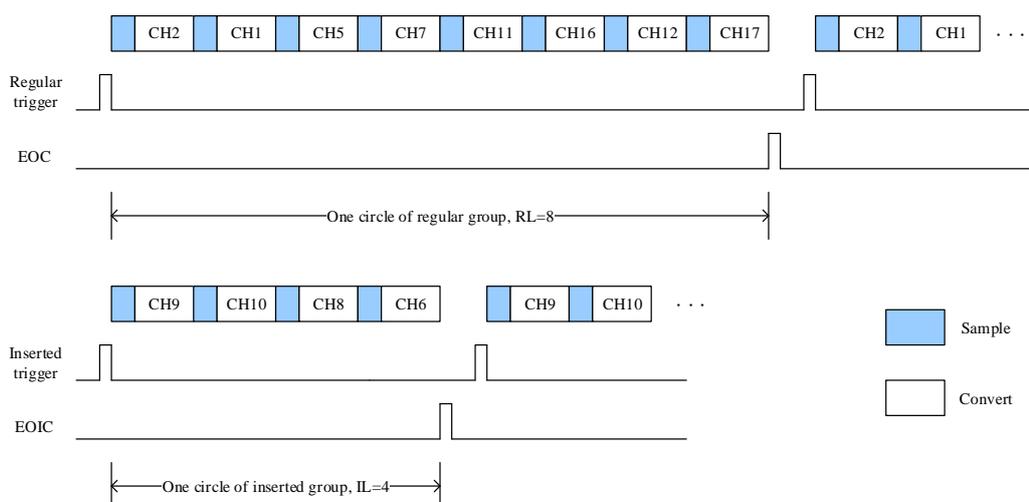
As while as loop up the EOC flag in loop, DMA can be used to transfer the converted data:

1. Set the CTN and DMA bit in the ADC\_CTL1 register
2. Configure RSQ0 with the analog channel number
3. Configure ADC\_SAMPTx register
4. Configure ETERC and ETSRC bits in the ADC\_CTL1 register in if need
5. Prepare the DMA module to transfer data from the ADC\_RDATA (refer to the spec of the DMA module).
6. Set the SWRCST bit, or generate an external trigger for the regular group

### Scan conversion mode

The scan conversion mode will be enabled when SM bit in the ADC\_CTL0 register is set. In this mode, the ADC performs conversion on the channels with a specific sequence specified in the ADC\_RSQ0~ADC\_RSQ2 registers or ADC\_ISQ register. When the ADCON has been set high, the ADC samples and converts specified channels one by one in the regular or inserted group till the end of the regular or inserted group, once the corresponding software trigger or external trigger is active. The conversion data will be stored in the ADC\_RDATA or ADC\_IDATAx register. After conversion of the regular or inserted channel group, the EOC or EOIC will be set. An interrupt will be generated if the EOCIE or EOICIE bit is set. The DMA bit in ADC\_CTL1 register must be set when the regular channel group works in scan mode.

After conversion of a regular channel group, the conversion can be restarted automatically if the CTN bit in the ADC\_CTL1 register is set.

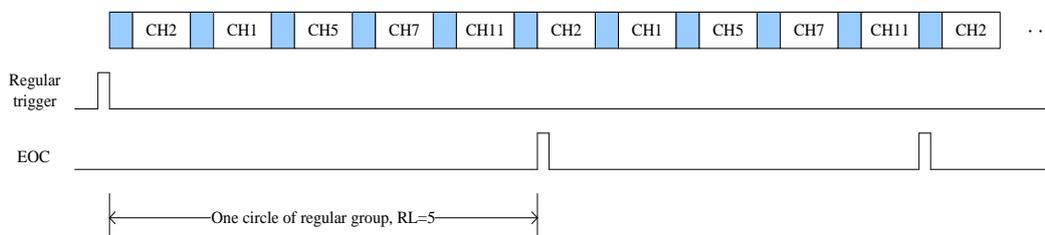
**Figure 10-5. Scan conversion mode, continuous disable**


Software procedure for scan conversion on a regular channel group:

1. Set the SM bit in the ADC\_CTL0 register and the DMA bit in the ADC\_CTL1 register
2. Configure ADC\_RSQx and ADC\_SAMPTx registers
3. Configure ETERC and ETSRC bits in the ADC\_CTL1 register if in need
4. Prepare the DMA module to transfer data from the ADC\_RDATA (refer to the spec of the DMA module).
5. Set the SWRCST bit, or generate an external trigger for the regular group
6. Wait the EOC flag to be set
7. Clear the EOC flag by writing 0 to it

Software procedure for scan conversion on an inserted channel group:

1. Set the SM bit in the ADC\_CTL0 register
2. Configure ADC\_ISQ and ADC\_SAMPTx registers
3. Configure ETEIC and ETSIC bits in the ADC\_CTL1 register if in need
4. Set the SWICST bit, or generate an external trigger for the inserted group
5. Wait the EOC/EOIC flags to be set
6. Read the converted result in the ADC\_IDATAx register
7. Clear the EOC/EOIC flag by writing 0 to them

**Figure 10-6. Scan conversion mode, continuous enable**


## Discontinuous mode

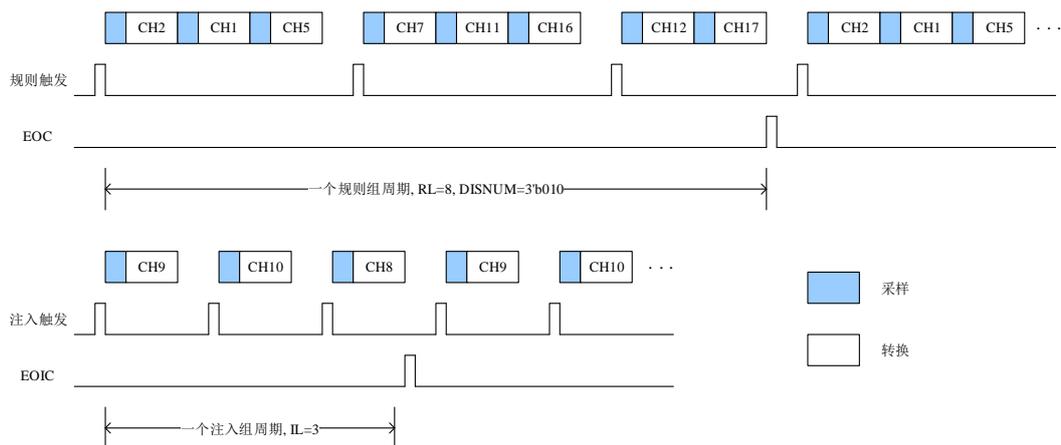
For regular channel group, the discontinuous conversion mode will be enabled when DISRC

bit in the ADC\_CTL0 register is set. In this mode, the ADC performs a short sequence of  $n$  conversions ( $n \leq 8$ ) which is a part of the sequence of conversions selected in the ADC\_RSQ0~ADC\_RSQ2 registers. The value of  $n$  is defined by the DISNUM [2:0] bits in the ADC\_CTL0 register. When the corresponding software trigger or external trigger is active, the ADC samples and converts the next  $n$  channels selected in the ADC\_RSQ0~ADC\_RSQ2 registers until all the channels in the regular sequence are done. The EOC will be set after every circle of the regular channel group. An interrupt will be generated if the EOCIE bit is set.

For inserted channel group, the discontinuous conversion mode will be enabled when DISIC bit in the ADC\_CTL0 register is set. In this mode, the ADC performs one conversion which is a part of the sequence of conversions selected in the ADC\_ISQ register. When the corresponding software trigger or external trigger is active, the ADC samples and converts the next channel selected in the ADC\_ISQ register until all the channels in the inserted sequence are done. The EOIC will be set after every circle of the inserted channel group. An interrupt will be generated if the EOICIE bit is set.

The regular and inserted groups cannot both work in discontinuous conversion mode. Only one group conversion can be set in discontinuous conversion mode at a time.

**Figure 10-7. Discontinuous conversion mode**



Software procedure for discontinuous conversion on a regular channel group:

1. Set the DISRC bit in the ADC\_CTL0 register and the DMA bit in the ADC\_CTL1 register
2. Configure DISNUM [2:0] bits in the ADC\_CTL0 register
3. Configure ADC\_RSQx and ADC\_SAMPTx registers
4. Configure ETERC and ETSRC bits in the ADC\_CTL1 register if in need
5. Prepare the DMA module to transfer data from the ADC\_RDATA (refer to the spec of the DMA module).
6. Set the SWRCST bit, or generate an external trigger for the regular group
7. Repeat step6 if in need.
8. Wait the EOC flag to be set
9. Clear the EOC flag by writing 0 to it

Software procedure for discontinuous conversion on an inserted channel group:

1. Set the DISIC bit in the ADC\_CTL0 register

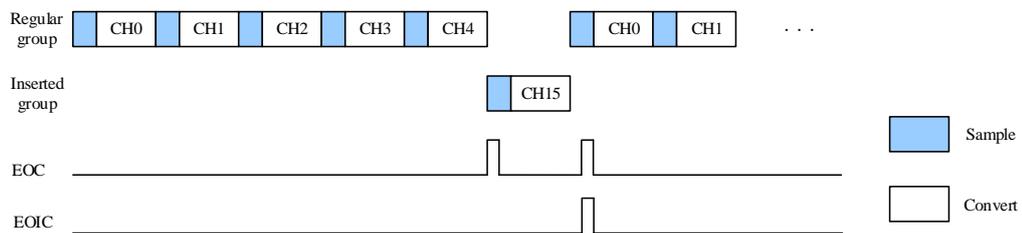
2. Configure ADC\_ISQ and ADC\_SAMPTx registers
3. Configure ETEIC and ETSIC bits in the ADC\_CTL1 register if in need
4. Set the SWICST bit, or generate an external trigger for the inserted group
5. Repeat step4 if in need
6. Wait the EOC/EOIC flags to be set
7. Read the converted result in the ADC\_IDATAx register
8. Clear the EOC/EOIC flag by writing 0 to them

### 10.4.6. Inserted channel management

#### Auto-insertion

The inserted group channels are automatically converted after the regular group channels when the ICA bit in ADC\_CTL0 register is set. In this mode, external trigger on inserted channels cannot be enabled. A sequence of up to 20 conversions programmed in the ADC\_RSQ0~ADC\_RSQ2 and ADC\_ISQ registers can be used to convert in this mode. In addition to the ICA bit, if the CTN bit is also set, regular channels followed by inserted channels are continuously converted.

**Figure 10-8. Auto-insertion, CTN = 1**

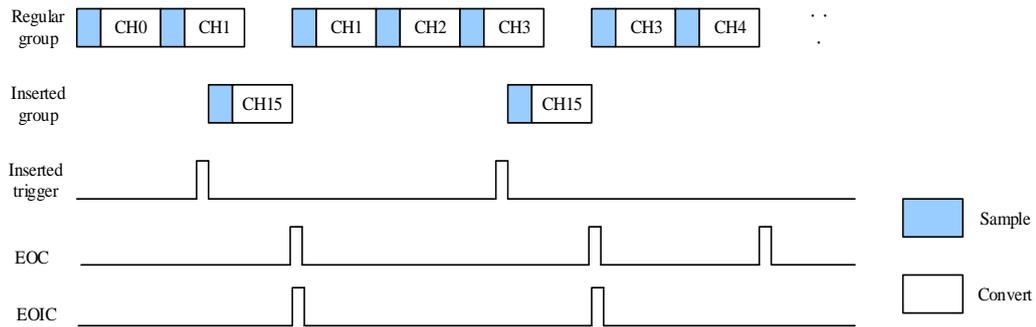


The auto insertion mode cannot be enabled when the discontinuous conversion mode is set.

#### Triggered insertion

If the ICA bit is cleared, the triggered insertion occurs if a software or external trigger occurs during the regular group channel conversion. In this situation, the ADC aborts from the current conversion and starts the conversion of inserted channel sequence in scan mode. After the inserted channel group is done, the regular group channel conversion is resumed from the last aborted conversion.

**Figure 10-9. Triggered insertion**



### 10.4.7. Analog watchdog

The analog watchdog is enabled when the RWDEN and IW DEN bits in the ADC\_CTL0 register are set for regular and inserted channel groups respectively. When the analog voltage converted by the ADC is below a low threshold or above a high threshold, the WDE bit in ADC\_STAT register will be set. An interrupt will be generated if the WDEIE bit is set. The ADC\_WDHT and ADC\_WDLT registers are used to specify the high and low threshold. The comparison is done before the alignment, so the threshold value is independent of the alignment, which is specified by the DAL bit in the ADC\_CTL1 register. One or more channels to be monitored by the analog watchdog are selected by the RW DEN, IW DEN, WDSC and WDCHSEL [4:0] bits in ADC\_CTL0 register.

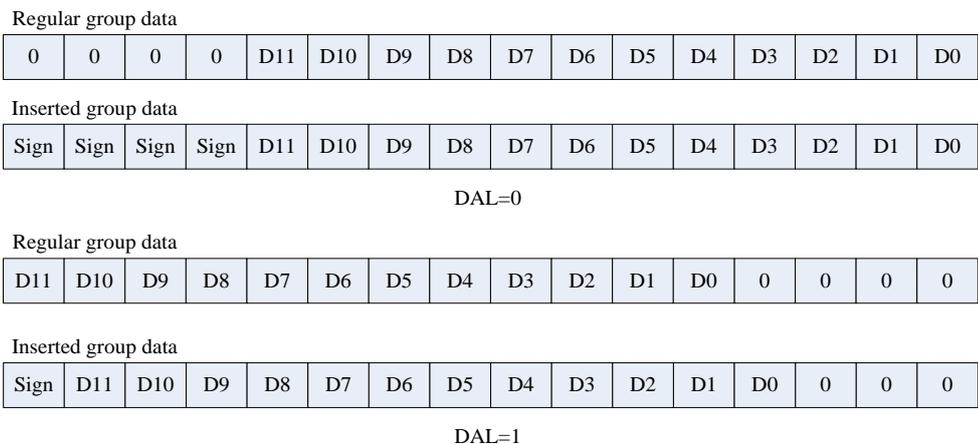
### 10.4.8. Data alignment

The alignment of data stored after conversion can be specified by DAL bit in the ADC\_CTL1 register.

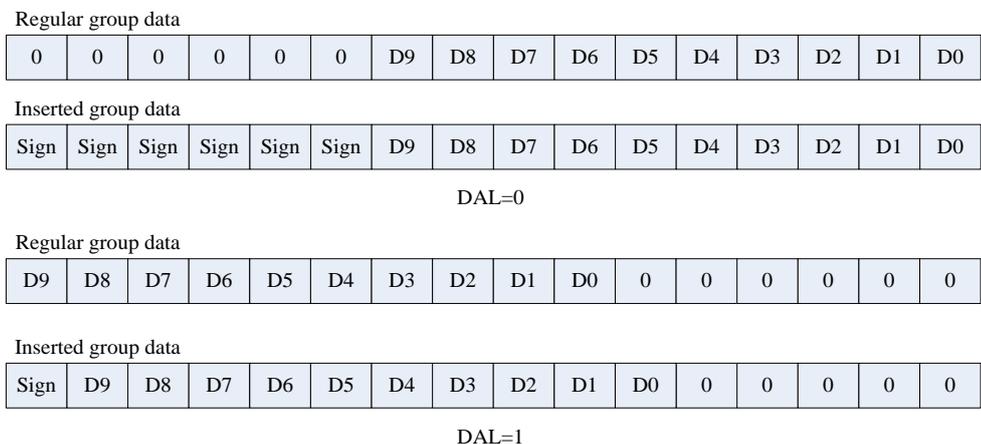
After decreased by the user-defined offset written in the ADC\_IOFFx registers, the inserted group data value may be a negative value. The sign value is extended.

When left-aligned, the 12/10/8-bit data are aligned on a half-word, while the 6-bit data are aligned on a byte basis as shown below [Figure 10-10. Data alignment of 12-bit resolution](#), [Figure 10-11. Data alignment of 10-bit resolution](#), [Figure 10-12. Data alignment of 8-bit resolution](#) and [Figure 10-13. Data alignment of 6-bit resolution](#).

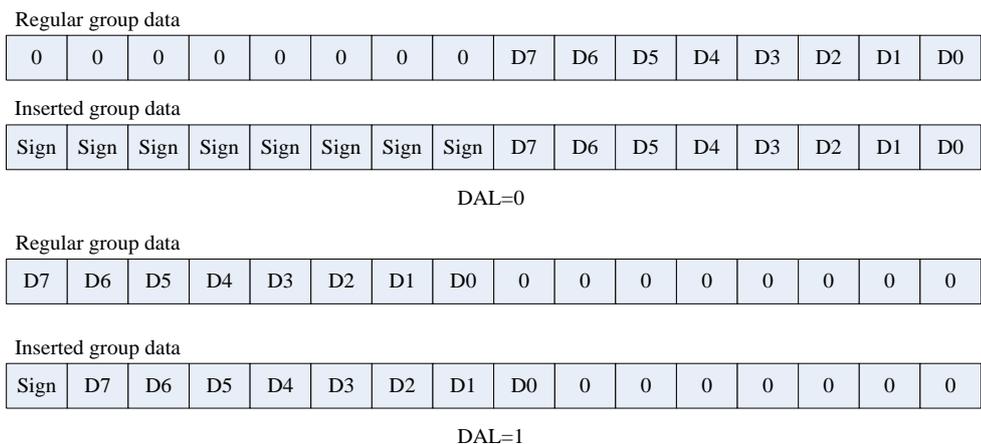
**Figure 10-10. Data alignment of 12-bit resolution**



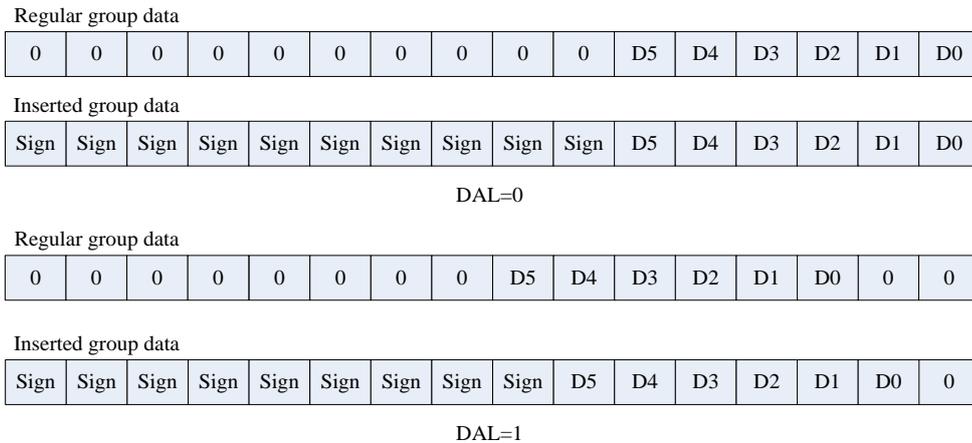
**Figure 10-11. Data alignment of 10-bit resolution**



**Figure 10-12. Data alignment of 8-bit resolution**



**Figure 10-13. Data alignment of 6-bit resolution**



### 10.4.9. Programmable sample time

The number of ADC\_CLK cycles which is used to sample the input voltage can be specified by the SPTn[2:0] bits in the ADC\_SAMPT0 and ADC\_SAMPT1 registers. A different sample time can be specified for each channel. Take the 12-bit resolution as an example, its total conversion time is “sampling time + 12.5” ADC\_CLK cycles.

Example:

ADCCLK = 28MHz and sample time is 1.5 cycles, the total conversion time is “1.5+12.5” ADCCLK cycles, that means 0.500us.

### 10.4.10. External trigger

The conversion of regular or inserted group can be triggered by rising edge of external trigger inputs. The external trigger source of regular channel group is controlled by the ETSRC [2:0] bits in the ADC\_CTL1 register, while the external trigger source of inserted channel group is controlled by the ETSIC [2:0] bits in the ADC\_CTL1 register

The ETSRC [2:0] and ETSIC [2:0] control bits are used to specify which out of 8 possible events can trigger conversion for the regular and inserted groups.

**Table 10-4. External trigger for regular channels of ADC**

ETSRC [2:0]	Trigger Source	Trigger Type
000	TIMER0_CH0	Internal on-chip signal
001	TIMER0_CH1	
010	TIMER0_CH2	
011	TIMER1_CH1	
100	TIMER2_TRGO	
101	TIMER14_CH0	

ETSRC [2:0]	Trigger Source	Trigger Type
110	EXTI_11	External signal
111	SWRCST	Software trigger

**Table 10-5. External trigger for inserted channels of ADC**

ETSIC [2:0]	Trigger Source	Trigger Type
000	TIMER0_TRGO	Internal on-chip signal
001	TIMER0_CH3	
010	TIMER1_TRGO	
011	TIMER1_CH0	
100	TIMER2_CH3	
101	TIMER14_TRGO	
110	EXTI_15	External signal
111	SWICST	Software trigger

#### 10.4.11. DMA request

The DMA request is used to transfer data of regular group for conversion of more than one channel. The DMA request, which is enabled by the DMA bit of ADC\_CTL1 register. The ADC generates a DMA request at the end of conversion of a regular channel. When this request is received, the DMA will transfer the converted data from the ADC\_RDATA register to the destination location which is specified by the user.

#### 10.4.12. Temperature sensor and internal reference voltage $V_{REFINT}$

When the TSVREN bit in ADC\_CTL1 register is set, the temperature sensor channel (ADC\_IN16) and  $V_{REFINT}$  channel (ADC\_IN17) is enabled. The temperature sensor can be used to measure the ambient temperature of the device. The sensor output voltage can be converted into a digital value by ADC. The sampling time for the temperature sensor is recommended to be set to 17.1 $\mu$ s. When this sensor is not in use, it can be put in power down mode by resetting the TSVREN bit.

The output voltage of the temperature sensor changes linearly with temperature. Because there is an offset, which is up to 45°C and varies from chip to chip due to process variation, on this line, the internal temperature sensor is more suited for applications that detect temperature variations instead of absolute temperatures. When it is used to detect accurate temperature, an external temperature sensor part should be used.

The internal voltage reference ( $V_{REFINT}$ ) provides a stable (bandgap) voltage output for the ADC and Comparators.  $V_{REFINT}$  is internally connected to the ADC\_IN17 input channel.

To use the temperature sensor:

1. Configure the conversion sequence (ADC\_IN16) and the sampling time (17.1μs) for the channel.
2. Enable the temperature sensor by setting the TSVREN bit in the ADC control register 1 (ADC\_CTL1).
3. Start the ADC conversion by setting the ADCON bit (or by external trigger).
4. Read the resulting temperature data ( $V_{\text{temperature}}$ ) in the ADC data register, and get the temperature using the following formula:

$$\text{Temperature (}^{\circ}\text{C)} = \{(V_{25} - V_{\text{temperature}}(\text{digit})) / \text{Avg\_Slope}\} + 25.$$

$V_{25}$ :  $V_{\text{temperature}}$  value at 25°C, the typical value is 1.43 V.

Avg\_Slope : Average Slope for curve between Temperature vs.  $V_{\text{temperature}}$ , the typical value is 4.3 mV/°C.

### 10.4.13. Battery voltage monitoring

The  $V_{\text{BAT}}$  channel can be used to measure the backup battery voltage on the  $V_{\text{BAT}}$  pin. When the VBATEN bit in ADC\_CTL1 register is set,  $V_{\text{BAT}}$  channel (ADC\_IN18) is enabled and a bridge divider by 2 integrated on the  $V_{\text{BAT}}$  pin is also enabled automatically with it. As  $V_{\text{BAT}}$  may be higher than  $V_{\text{DDA}}$ , this bridge is used to ensure the ADC correct operation. And it connect  $V_{\text{BAT}}/2$  to the ADC\_IN18 input channel. So, the converted digital value is  $V_{\text{BAT}}/2$ . In order to prevent unnecessary battery energy consumption, it is recommended that the bridge will be enabled only when it is required.

### 10.4.14. ADC interrupts

The interrupt can be produced on one of the events:

- End of conversion for regular and inserted groups
- The analog watchdog event (the analog watchdog status bit is set)

Separate interrupt enable bits are available for flexibility.

### 10.4.15. Programmable resolution (DRES) - fast conversion mode for GD32F170xx and GD32F190xx devices

It is possible to obtain faster conversion time ( $t_{\text{ADC}}$ ) by reducing the ADC resolution.

The resolution can be configured to be either 12, 10, 8, or 6 bits by programming the DRES[1:0] bits in the ADC\_CTL0 register. Lower resolution allows faster conversion time for applications where high data precision is not required.

The DRES [1:0] bits must only be changed when the ADCON bit is reset.

The result of the conversion is always 12 bits wide and any unused LSB bits are read as zeroes.

Lower resolution reduces the conversion time needed for the successive approximation steps as shown in [Table 10-6. t<sub>CONV</sub> timings depending on resolution.](#)

**Table 10-6. t<sub>CONV</sub> timings depending on resolution**

DRES [1:0] bits	t <sub>CONV</sub> (ADC clock cycles)	t <sub>CONV</sub> (ns) at f <sub>ADC</sub> =28MHz	t <sub>SMPL</sub> (ADC clock cycles)	t <sub>ADC</sub> (ADC clock cycles)	t <sub>ADC</sub> (ns) at f <sub>ADC</sub> =28MHz
12	12.5	446ns	1.5	14	500ns
10	10.5	375ns	1.5	12	429ns
8	8.5	304ns	1.5	10	357ns
6	6.5	232ns	1.5	8	286ns

## 10.4.16. On-chip hardware oversampling for GD32F170xx and GD32F190xx devices

The on-chip hardware oversampling unit, which is enabled by OVSEN bit in the ADC\_OVSAMPCTL register, provides higher data resolution at the cost of lower output data rate.

The on-chip hardware oversampling unit performs data preprocessing to offload the CPU. It can handle multiple conversions and average them into a single data with increased data width, up to 16-bit.

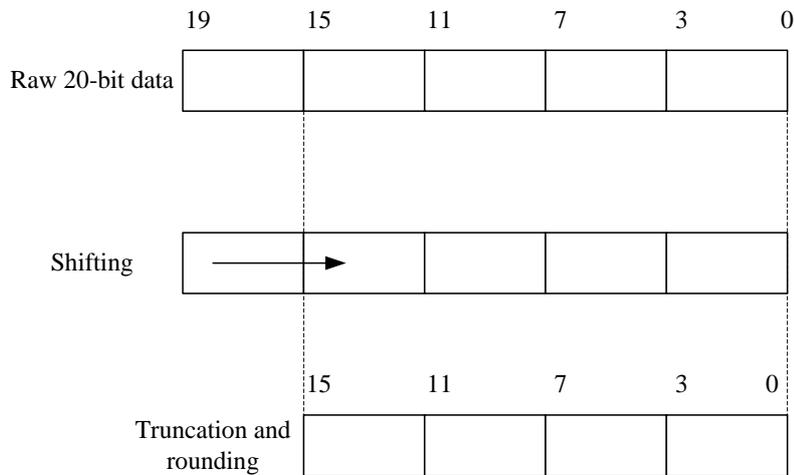
It provides a result with the following form, where N and M can be adjusted, and D<sub>OUT</sub>(n) is the n-th output digital signal of the ADC:

$$\text{Result} = \frac{1}{M} * \sum_{n=0}^{n=N-1} D_{OUT}(n) \tag{11-1}$$

The on-chip hardware oversampling circuit performs the following functions: summing and bit right shifting. The oversampling ratio N is defined using the OVSR[2:0] bits in the ADC\_OVSAMPCTL register. It can range from 2x to 256x. The division coefficient M consists of a right bit shift up to 8 bits. It is configured through the OVSS[3:0] bits in the ADC\_OVSAMPCTL register.

The summation unit can yield a result up to 20 bits (256 x 12-bit), which is first shifted right. The upper bits of the result are then truncated, keeping only the 16 least significant bits rounded to the nearest value using the least significant bits left apart by the shifting, before being finally transferred into the data register.

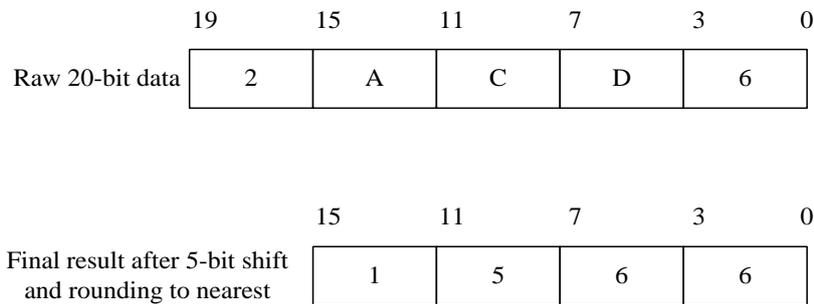
**Figure 10-14. 20-bit to 16-bit result truncation**



**Note:** If the intermediate result after the shifting exceeds 16 bits, the upper bits of the result are simply truncated.

[Figure 10-15. Numerical example with 5-bits shift and rounding](#) shows a numerical example of the processing, from a raw 20-bit accumulated data to the final 16-bit result.

**Figure 10-15. Numerical example with 5-bits shift and rounding**



[Table 10-7. Maximum output results vs N and M \(Grayed values indicates truncation\)](#)

gives the data format for the various N and M combination, for a raw conversion data equal to 0xFFFF.

**Table 10-7. Maximum output results vs N and M (Grayed values indicates truncation)**

Oversampling ratio	Max Raw data	No-shift OVSS= 0000	1-bit shift OVSS= 0001	2-bit shift OVSS= 0010	3-bit shift OVSS= 0011	4-bit shift OVSS= 0100	5-bit shift OVSS= 0101	6-bit shift OVSS= 0110	7-bit shift OVSS= 0111	8-bit shift OVSS= 1000
2x	0x1FFE	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100	0x0080	0x0040	0x0020
4x	0x3FFC	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100	0x0080	0x0040
8x	0x7FF8	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100	0x0080
16x	0xFFF0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100

32x	0x1FFE0	0xFFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200
64x	0x3FFC0	0xFFC0	0xFFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400
128x	0x7FF80	0xFF80	0xFFC0	0xFFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800
256x	0xFFF00	0xFF00	0xFF80	0xFFC0	0xFFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF

The conversion time in oversampling mode do not change compared to standard conversion mode: the sampling time is maintained equal during the whole oversampling sequence. New data are provided every N conversion, with an equivalent delay equal to  $N \times t_{ADC} = N \times (t_{SAMPL} + t_{conv})$ .

### Oversampling work with ADC modes

Most of the ADC work modes are available when oversampling is enabled.

- Regular or inserted channels
- ADC started by software or external triggers
- Single or scan, continuous or discontinuous conversion modes
- Programmable sample time
- Analog watchdog

The oversampling configuration can only be changed when ADCON is reset. Make sure configuring the oversampling before setting ADCON to 1.

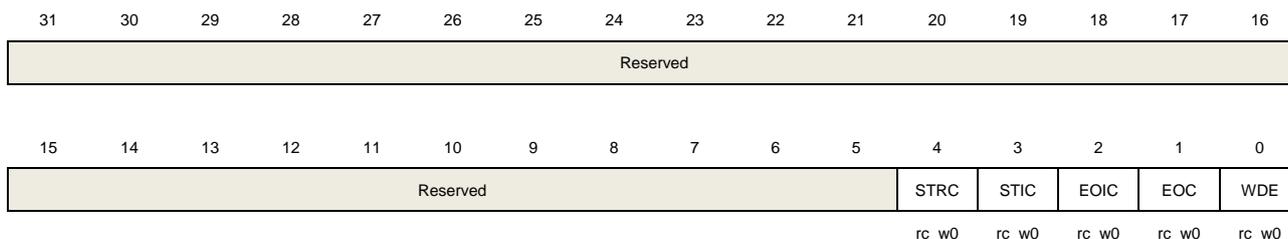
## 10.5. Register definition

### 10.5.1. Status register (ADC\_STAT)

Address offset: 0x00

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



Bits	Fields	Descriptions
31:5	Reserved	Must be kept at reset value
4	STRC	Start flag of regular channel group 0: No regular channel group started 1: Regular channel group started Set by hardware when regular channel conversion starts. Cleared by software writing 0 to it.
3	STIC	Start flag of inserted channel group 0: No inserted channel group started 1: Inserted channel group started Set by hardware when inserted channel group conversion starts. Cleared by software writing 0 to it.
2	EOIC	End of inserted group conversion flag 0: No end of inserted group conversion 1: End of inserted group conversion Set by hardware at the end of all inserted group channel conversion. Cleared by software writing 0 to it.
1	EOC	End of group conversion flag 0: No end of group conversion 1: End of group conversion Set by hardware at the end of a regular or inserted group channel conversion. Cleared by software writing 0 to it or by reading the ADC_RDATA register.
0	WDE	Analog watchdog event flag 0: No analog watchdog event 1: Analog watchdog event

Set by hardware when the converted voltage crosses the values programmed in the ADC\_WDLT and ADC\_WDHT registers.  
Cleared by software writing 0 to it.

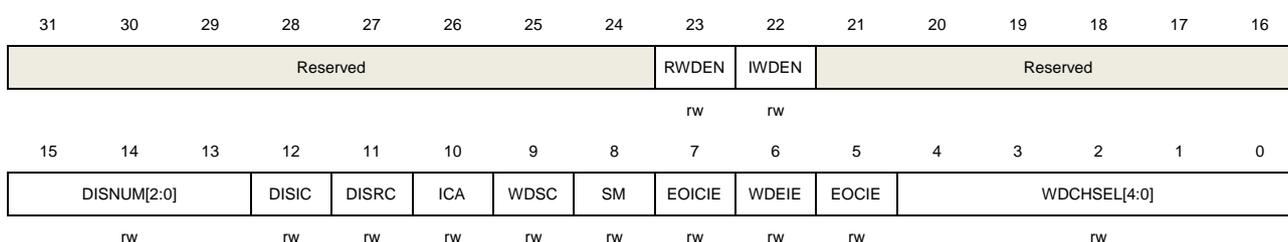
## 10.5.2. Control register 0 (ADC\_CTL0)

**For GD32F130xx and GD32F150xx devices:**

Address offset: 0x04

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)



Bits	Fields	Descriptions
31:24	Reserved	Must be kept at reset value
23	RWDEN	Regular channel analog watchdog enable 0: Analog watchdog regular channel disable 1: Analog watchdog regular channel enable
22	IWDEN	Inserted channel analog watchdog enable 0: Analog watchdog inserted channel disable 1: Analog watchdog inserted channel enable
21:16	Reserved	Must be kept at reset value
15:13	DISNUM[2:0]	Number of conversions in discontinuous mode The number of channels to be converted after a trigger will be DISNUM[2:0]+1
12	DISIC	Discontinuous mode on inserted channels 0: Discontinuous mode on inserted channels disable 1: Discontinuous mode on inserted channels enable
11	DISRC	Discontinuous mode on regular channels 0: Discontinuous mode on regular channels disable 1: Discontinuous mode on regular channels enable
10	ICA	Inserted channel group convert automatically 0: Inserted channel group convert automatically disable 1: Inserted channel group convert automatically enable

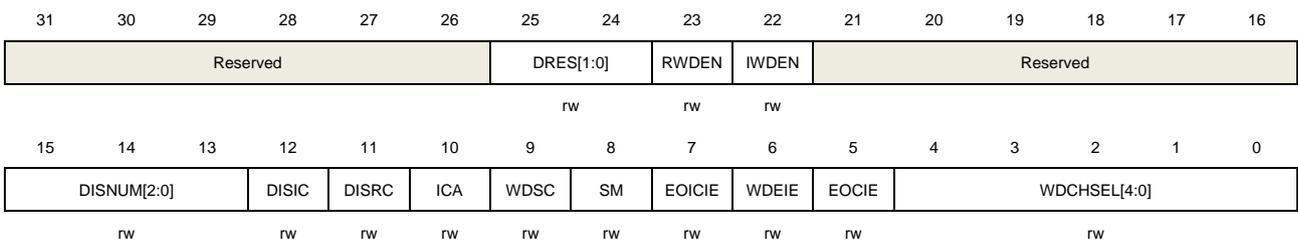
9	WDSC	When in scan mode, analog watchdog is effective on a single channel 0: Analog watchdog is effective on all channels 1: Analog watchdog is effective on a single channel
8	SM	Scan mode 0: scan mode disable 1: scan mode enable
7	EOICIE	Interrupt enable for EOIC 0: EOIC interrupt disable 1: EOIC interrupt enable
6	WDEIE	Interrupt enable for WDE 0: WDE interrupt disable 1: WDE interrupt enable
5	EOCIE	Interrupt enable for EOC 0: EOC interrupt disable 1: EOC interrupt enable
4:0	WDCHSEL[4:0]	Analog watchdog channel select 00000: ADC channel0 00001: ADC channel1 00010: ADC channel2 ..... 01111: ADC channel15 10000: ADC channel16 10001: ADC channel17 10010: ADC channel18 Other values are reserved. Note: ADC analog inputs Channel16, Channel17 and Channel 18 are internally connected to the temperature sensor, to VREFINT and to VBAT analog inputs.

**For GD32F170xx and GD32F190xx devices:**

Address offset: 0x04

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



Bits	Fields	Descriptions
31:26	Reserved	Must be kept at reset value
25:24	DRES[1:0]	ADC resolution 00: 12bit 01: 10bit 10: 8bit 11: 6bit
23	RWDEN	Regular channel analog watchdog enable 0: Analog watchdog regular channel disable 1: Analog watchdog regular channel enable
22	IWDEN	Inserted channel analog watchdog enable 0: Analog watchdog inserted channel disable 1: Analog watchdog inserted channel enable
21:16	Reserved	Must be kept at reset value
15:13	DISNUM[2:0]	Number of conversions in discontinuous mode The number of channels to be converted after a trigger will be DISNUM[2:0]+1
12	DISIC	Discontinuous mode on inserted channels 0: Discontinuous mode on inserted channels disable 1: Discontinuous mode on inserted channels enable
11	DISRC	Discontinuous mode on regular channels 0: Discontinuous mode on regular channels disable 1: Discontinuous mode on regular channels enable
10	ICA	Inserted channel group convert automatically 0: Inserted channel group convert automatically disable 1: Inserted channel group convert automatically enable
9	WDSC	When in scan mode, analog watchdog is effective on a single channel 0: Analog watchdog is effective on all channels 1: Analog watchdog is effective on a single channel
8	SM	Scan mode 0: Scan mode disable 1: Scan mode enable
7	EOICIE	Interrupt enable for EOIC 0: EOIC interrupt disable 1: EOIC interrupt enable
6	WDEIE	Interrupt enable for WDE 0: WDE interrupt disable 1: WDE interrupt enable



21	SWICST	Start on inserted channel. Set 1 on this bit starts a conversion of a group of inserted channels if ETSIC is 111. It is set by software and cleared by software or by hardware after the conversion starts.
20	ETERC	External trigger enable for regular channel 0: External trigger for regular channel disable 1: External trigger for regular channel enable
19:17	ETSRC[2:0]	External trigger select for regular channel 000: TIMER0 CH0 001: TIMER0 CH1 010: TIMER0 CH2 011: TIMER1 CH1 100: TIMER2 TRGO 101: TIMER14 CH1 110: EXTI line 11 111: SWRCST
16	Reserved	Must be kept at reset value
15	ETEIC	External trigger enable for inserted channel 0: External trigger for inserted channel disable 1: External trigger for inserted channel enable
14:12	ETSIC[2:0]	External trigger select for inserted channel 000: TIMER0 TRGO 001: TIMER0 CH3 010: TIMER1 TRGO 011: TIMER1 CH0 100: TIMER2 CH3 101: TIMER14 TRGO 110: EXTI line15 111: SWICST
11	DAL	Data alignment 0: LSB alignment 1: MSB alignment
10:9	Reserved	Must be kept at reset value
8	DMA	DMA request enable. 0: DMA request disable 1: DMA request enable
7:4	Reserved	Must be kept at reset value
3	RSTCLB	Reset calibration This bit is set by software and cleared by hardware after the calibration registers are initialized.

		0: Calibration register initialize done. 1: Initialize calibration register start
2	CLB	ADC calibration 0: Calibration done 1: Calibration start
1	CTN	Continuous mode 0: Continuous mode disable 1: Continuous mode enable
0	ADCON	ADC ON. The ADC will be wake up when this bit is changed from low to high. When this bit is high and “1” is written to it with other bits of this register unchanged, the conversion will start. 0: ADC disable and power down 1: ADC enable

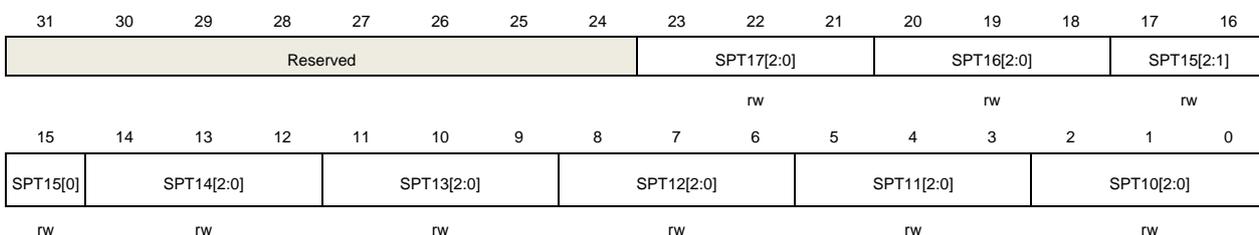
#### 10.5.4. Sampling time register 0 (ADC\_SAMPT0)

##### For GD32F130xx and GD32F150xx devices

Address offset: 0x0C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



Bits	Fields	Descriptions
31:24	Reserved	Must be kept at reset value
23:21	SPT17[2:0]	refer to SPT10[2:0] description
20:18	SPT16[2:0]	refer to SPT10[2:0] description
17:15	SPT15[2:0]	refer to SPT10[2:0] description
14:12	SPT14[2:0]	refer to SPT10[2:0] description
11:9	SPT13[2:0]	refer to SPT10[2:0] description
8:6	SPT12[2:0]	refer to SPT10[2:0] description
5:3	SPT11[2:0]	refer to SPT10[2:0] description
2:0	SPT10[2:0]	Channel sampling time

- 000: 1.5 cycles
- 001: 7.5 cycles
- 010: 13.5 cycles
- 011: 28.5 cycles
- 100: 41.5 cycles
- 101: 55.5 cycles
- 110: 71.5 cycles
- 111: 239.5 cycles

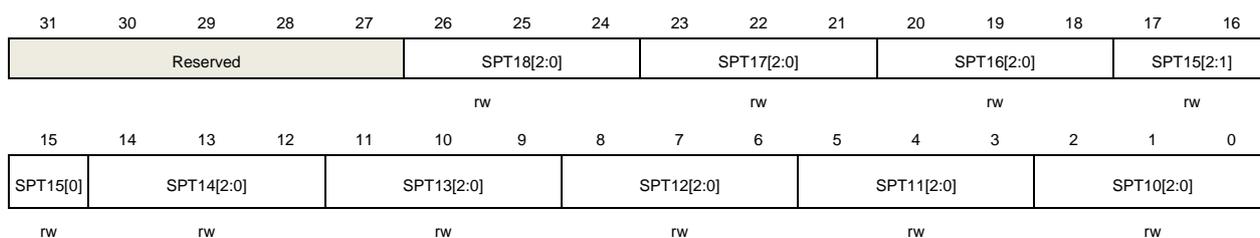
**Note:** For GD32F130xx and GD32F150xx devices, the channel 0 and channel 18 sampling time are specified by the SPT0[2:0] bits in the ADC\_SAMPT1.

### For GD32F170xx and GD32F190xx devices

Address offset: 0x0C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)



Bits	Fields	Descriptions
31:27	Reserved	Must be kept at reset value
26:24	SPT18[2:0]	refer to SPT10[2:0] description
23:21	SPT17[2:0]	refer to SPT10[2:0] description
20:18	SPT16[2:0]	refer to SPT10[2:0] description
17:15	SPT15[2:0]	refer to SPT10[2:0] description
14:12	SPT14[2:0]	refer to SPT10[2:0] description
11:9	SPT13[2:0]	refer to SPT10[2:0] description
8:6	SPT12[2:0]	refer to SPT10[2:0] description
5:3	SPT11[2:0]	refer to SPT10[2:0] description
2:0	SPT10[2:0]	Channel sampling time 000: 1.5 cycles 001: 7.5 cycles 010: 13.5 cycles 011: 28.5 cycles

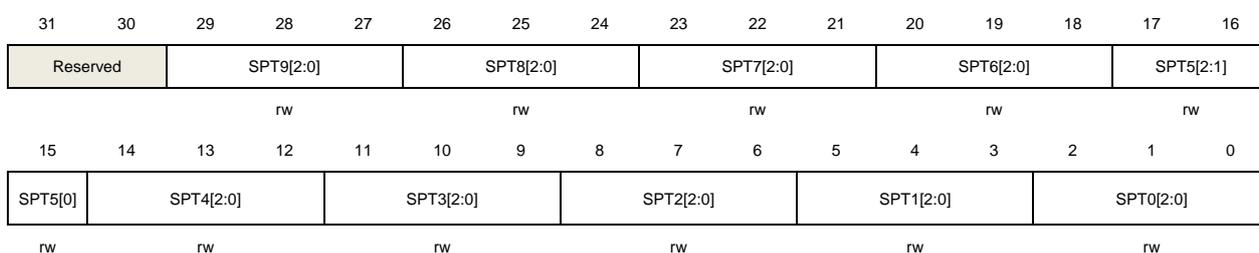
100: 41.5 cycles  
 101: 55.5 cycles  
 110: 71.5 cycles  
 111: 239.5 cycles

### 10.5.5. Sampling time register 1 (ADC\_SAMPT1)

Address offset: 0x10

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



Bits	Fields	Descriptions
31:30	Reserved	Must be kept at reset value
29:27	SPT9[2:0]	refer to SPT0[2:0] description
26:24	SPT8[2:0]	refer to SPT0[2:0] description
23:21	SPT7[2:0]	refer to SPT0[2:0] description
20:18	SPT6[2:0]	refer to SPT0[2:0] description
17:15	SPT5[2:0]	refer to SPT0[2:0] description
14:12	SPT4[2:0]	refer to SPT0[2:0] description
11:9	SPT3[2:0]	refer to SPT0[2:0] description
8:6	SPT2[2:0]	refer to SPT0[2:0] description
5:3	SPT1[2:0]	refer to SPT0[2:0] description
2:0	SPT0[2:0]	Channel sampling time 000: 1.5 cycles 001: 7.5 cycles 010: 13.5 cycles 011: 28.5 cycles 100: 41.5 cycles 101: 55.5 cycles 110: 71.5 cycles 111: 239.5 cycles

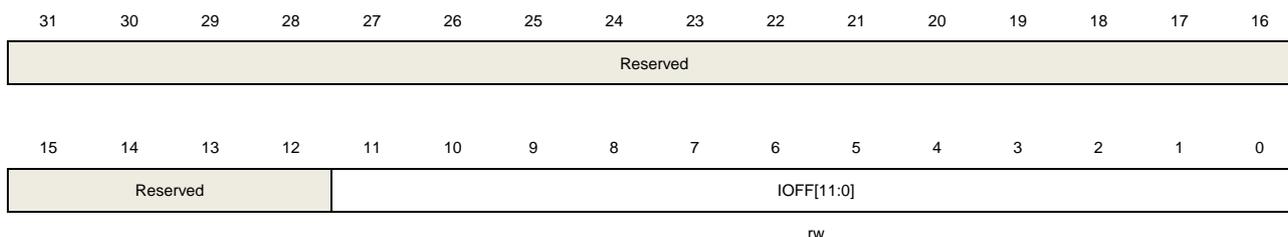
**Note:** For GD32F130xx and GD32F150xx devices, the channel 0 and channel 18 sampling time are specified by the SPT0[2:0] bits in the ADC\_SAMPT1.

### 10.5.6. Inserted channel data offset registers (ADC\_IOFFx) (x=0..3)

Address offset: 0x14-0x20

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



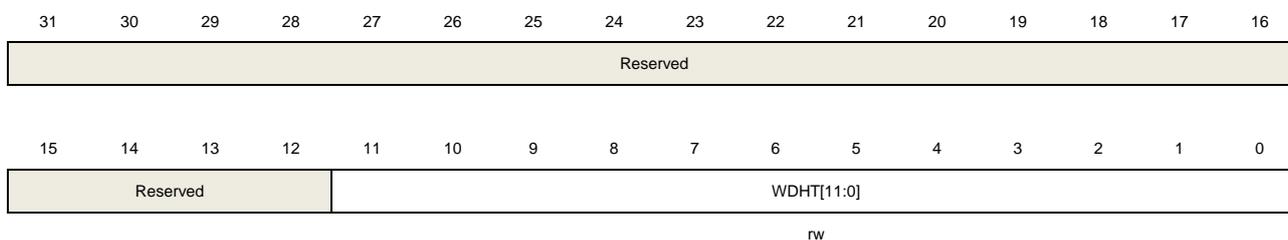
Bits	Fields	Descriptions
31:12	Reserved	Must be kept at reset value
11:0	IOFF[11:0]	Data offset for inserted channel x These bits will be subtracted from the raw converted data when converting inserted channels. The conversion result can be read from the ADC_IDATAx registers.

### 10.5.7. Watchdog high threshold register (ADC\_WDHT)

Address offset: 0x24

Reset value: 0x0000 0FFF

This register has to be accessed by word (32-bit).



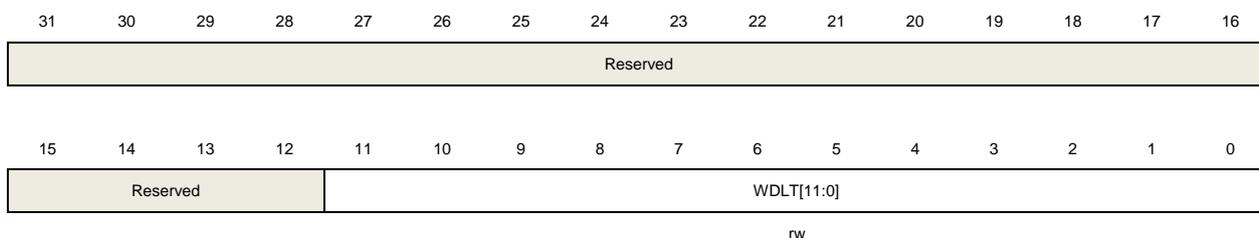
Bits	Fields	Descriptions
31:12	Reserved	Must be kept at reset value
11:0	WDHT[11:0]	Analog watchdog high threshold These bits define the high threshold for the analog watchdog.

### 10.5.8. Watchdog low threshold register (ADC\_WDLT)

Address offset: 0x28

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



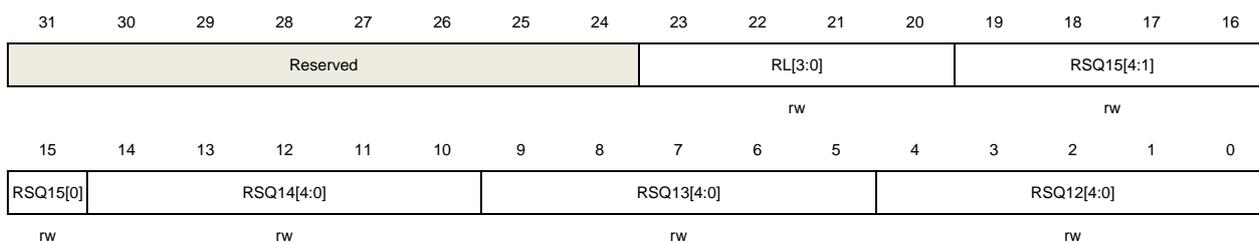
Bits	Fields	Descriptions
31:12	Reserved	Must be kept at reset value
11:0	WDLT[11:0]	Analog watchdog low threshold These bits define the low threshold for the analog watchdog.

### 10.5.9. Regular sequence register 0 (ADC\_RSQ0)

Address offset: 0x2C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



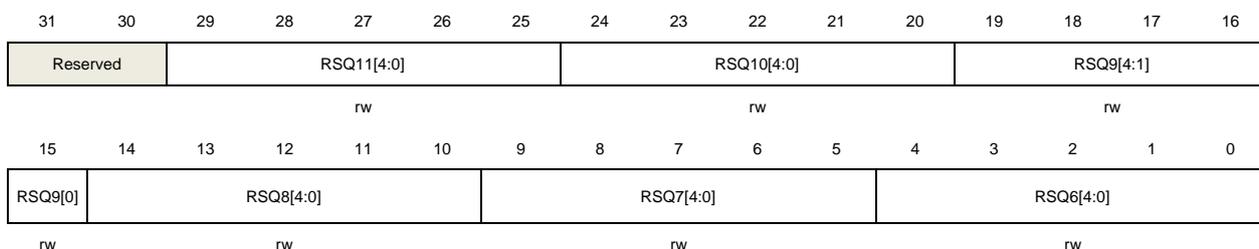
Bits	Fields	Descriptions
31:24	Reserved	Must be kept at reset value
23:20	RL[3:0]	Regular channel group length. The total number of conversion in regular group equals to RL[3:0]+1.
19:15	RSQ15[4:0]	refer to RSQ0[4:0] description
14:10	RSQ14[4:0]	refer to RSQ0[4:0] description
9:5	RSQ13[4:0]	refer to RSQ0[4:0] description
4:0	RSQ12[4:0]	refer to RSQ0[4:0] description

### 10.5.10. Regular sequence register 1 (ADC\_RSQ1)

Address offset: 0x30

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



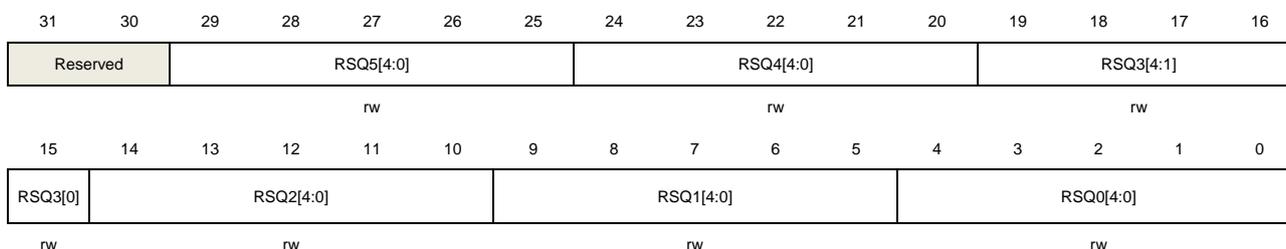
Bits	Fields	Descriptions
31:30	Reserved	Must be kept at reset value
29:25	RSQ11[4:0]	refer to RSQ0[4:0] description
24:20	RSQ10[4:0]	refer to RSQ0[4:0] description
19:15	RSQ9[4:0]	refer to RSQ0[4:0] description
14:10	RSQ8[4:0]	refer to RSQ0[4:0] description
9:5	RSQ7[4:0]	refer to RSQ0[4:0] description
4:0	RSQ6[4:0]	refer to RSQ0[4:0] description

### 10.5.11. Regular sequence register 2 (ADC\_RSQ2)

Address offset: 0x34

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



Bits	Fields	Descriptions
31:30	Reserved	Must be kept at reset value

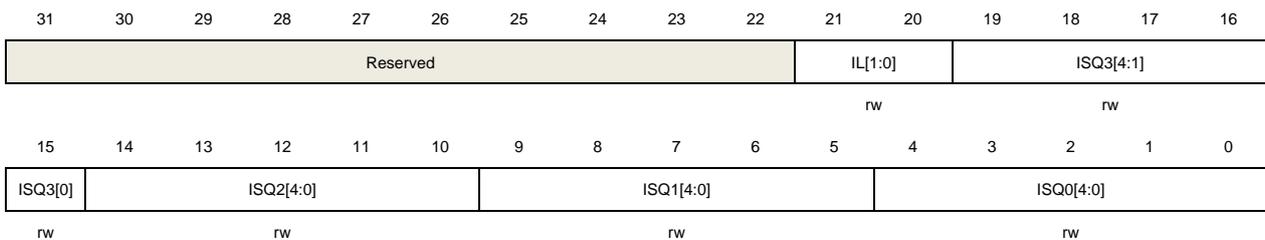
29:25	RSQ5[4:0]	refer to RSQ0[4:0] description
24:20	RSQ4[4:0]	refer to RSQ0[4:0] description
19:15	RSQ3[4:0]	refer to RSQ0[4:0] description
14:10	RSQ2[4:0]	refer to RSQ0[4:0] description
9:5	RSQ1[4:0]	refer to RSQ0[4:0] description
4:0	RSQ0[4:0]	The channel number (0..18) are written to these bits to select a channel at the nth conversion in the regular channel group.

### 10.5.12. Inserted sequence register (ADC\_ISQ)

Address offset: 0x38

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



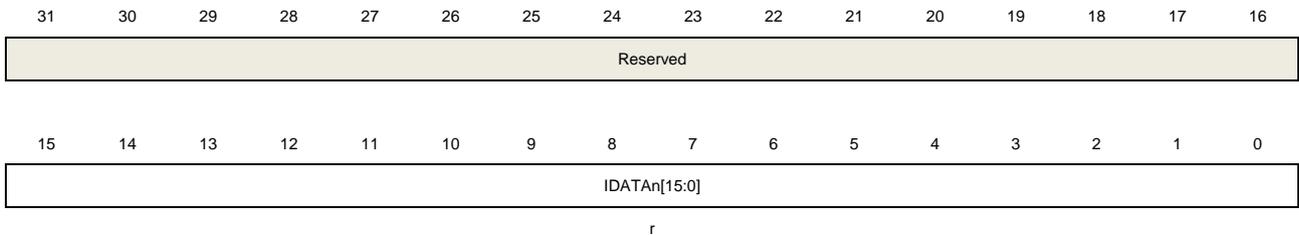
Bits	Fields	Descriptions										
31:22	Reserved	Must be kept at reset value										
21:20	IL[1:0]	Inserted channel group length. The total number of conversion in regular group equals to IL[1:0]+1.										
19:15	ISQ3[4:0]	refer to ISQ0[4:0] description										
14:10	ISQ2[4:0]	refer to ISQ0[4:0] description										
9:5	ISQ1[4:0]	refer to ISQ0[4:0] description										
4:0	ISQ0[4:0]	The channel number (0..18) are written to these bits to select a channel at the nth conversion in the inserted channel group. Unlike the regular conversion sequence, the inserted channels are converted starting from (4-IL [1:0]), if IL [1:0] length is less than 4. <table style="margin-left: 20px; border: none;"> <tr> <td>IL</td> <td>Insert channel order</td> </tr> <tr> <td>11</td> <td>ISQ0&gt;&gt;ISQ1&gt;&gt;ISQ2&gt;&gt;ISQ3</td> </tr> <tr> <td>10</td> <td>ISQ1&gt;&gt;ISQ2&gt;&gt;ISQ3</td> </tr> <tr> <td>01</td> <td>ISQ2&gt;&gt;ISQ3</td> </tr> <tr> <td>00</td> <td>ISQ3</td> </tr> </table>	IL	Insert channel order	11	ISQ0>>ISQ1>>ISQ2>>ISQ3	10	ISQ1>>ISQ2>>ISQ3	01	ISQ2>>ISQ3	00	ISQ3
IL	Insert channel order											
11	ISQ0>>ISQ1>>ISQ2>>ISQ3											
10	ISQ1>>ISQ2>>ISQ3											
01	ISQ2>>ISQ3											
00	ISQ3											

### 10.5.13. Inserted data registers (ADC\_IDATAx) (x= 0..3)

Address offset: 0x3C - 0x48

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



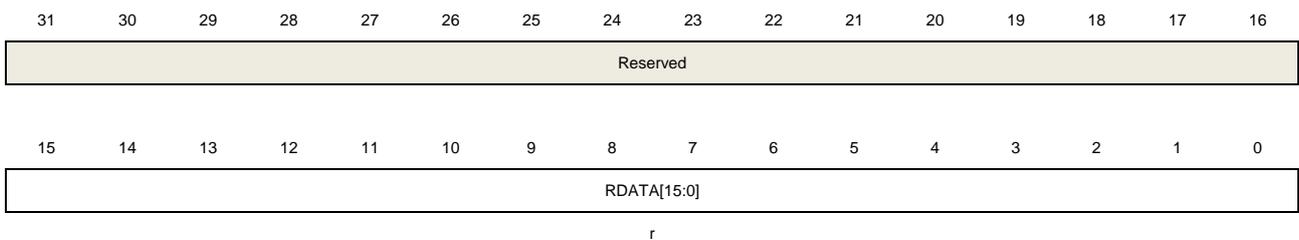
Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	IDATAN[15:0]	Inserted number n conversion data These bits contain the number n conversion result, which is read only.

### 10.5.14. Regular data register (ADC\_RDATA)

Address offset: 0x4C

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit).



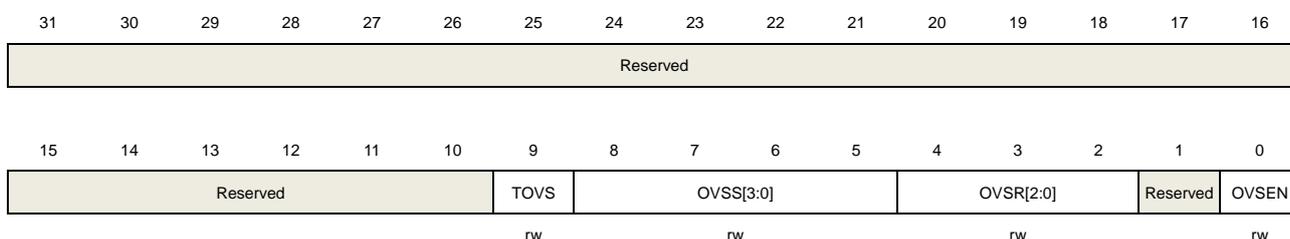
Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	RDATA[15:0]	Regular channel data These bits contain the conversion result from regular channel, which is read only.

### 10.5.15. Oversampling control register (ADC\_OVSAMPCTL) of GD32F170xx and GD32F190xx devices

Address offset: 0x80

Reset value: 0x0000\_0000

This register has to be accessed by word (32-bit).



Bits	Fields	Descriptions
31:10	Reserved	Must be kept at reset value
9	TOVS	<p>Triggered Oversampling</p> <p>This bit is set and cleared by software.</p> <p>0: All oversampled conversions for a channel are done consecutively after a trigger</p> <p>1: Each oversampled conversion for a channel needs a trigger</p> <p><b>Note:</b> Software is allowed to write this bit only when ADCON=0 (which ensures that no conversion is ongoing).</p>
8:5	OVSS[3:0]	<p>Oversampling shift</p> <p>This bit is set and cleared by software.</p> <p>0000: No shift</p> <p>0001: Shift 1-bit</p> <p>0010: Shift 2-bits</p> <p>0011: Shift 3-bits</p> <p>0100: Shift 4-bits</p> <p>0101: Shift 5-bits</p> <p>0110: Shift 6-bits</p> <p>0111: Shift 7-bits</p> <p>1000: Shift 8-bits</p> <p>Other codes reserved.</p> <p><b>Note:</b> Software is allowed to write this bit only when ADCON =0 (which ensures that no conversion is ongoing).</p>
4:2	OVSR[2:0]	<p>Oversampling ratio</p> <p>This bit field defines the number of oversampling ratio.</p> <p>000: 2x</p> <p>001: 4x</p> <p>010: 8x</p> <p>011: 16x</p> <p>100: 32x</p> <p>101: 64x</p> <p>110: 128x</p> <p>111: 256x</p>

**Note:** Software is allowed to write this bit only when ADCON =0 (which ensures that no conversion is ongoing).

1            Reserved            Must be kept at reset value

0            OVSEN            Oversampling Enable  
This bit is set and cleared by software.  
0: Oversampling disabled  
1: Oversampling enabled

**Note:** Software is allowed to write this bit only when ADCON =0 (which ensures that no conversion is ongoing).

## 11. Digital-to-analog converter (DAC)

### 11.1. Overview

The Digital-to-analog converter converts 12-bit digital data to a voltage on the external pins. The digital data can be configured in 8-bit or 12-bit mode, left-aligned or right-aligned mode. DMA can be used to update the digital data on external triggers. The output voltage can be optionally buffered for higher drive capability.

For GD32F150xx devices, the DAC module only has one DAC.

For GD32F190xx devices, the DAC module has two DACs, each with its own converter. In DAC concurrent mode, conversions could be done independently or concurrently when both DACs are grouped together for synchronous update operation.

### 11.2. Characteristic

DAC main features are as follows:

- 12-bit resolution. Left or right data alignment
- DMA capability for each channel
- Conversion update synchronously
- Conversion triggered by external triggers
- Configurable internal buffer
- Input voltage reference,  $V_{DDA}$

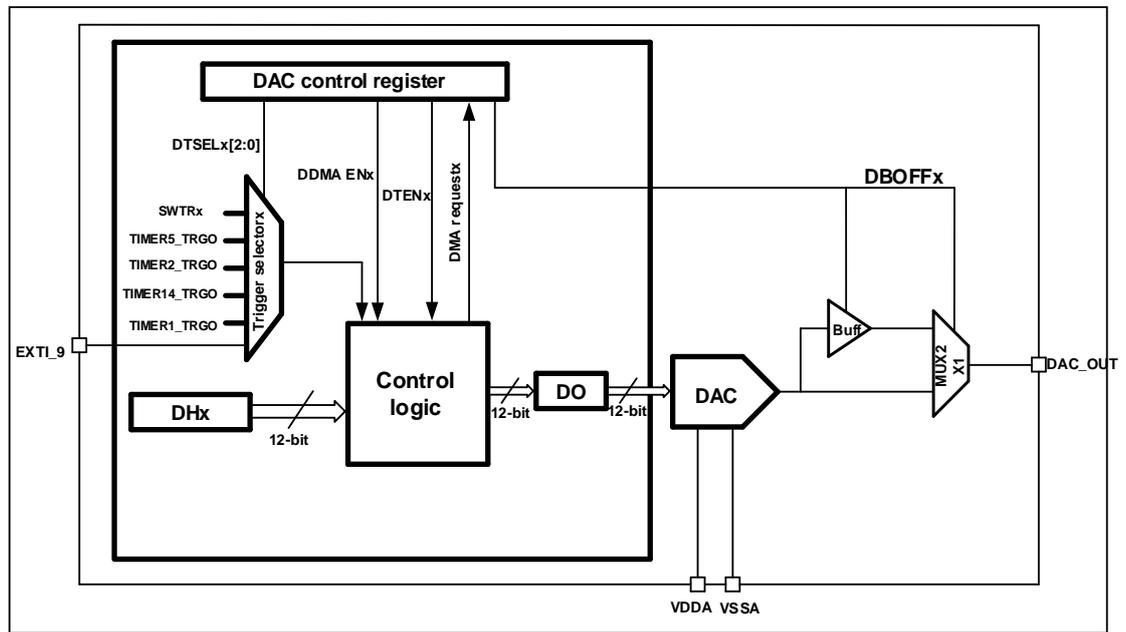
**For GD32F190xx devices, additional features are shown below.**

- Two DAC converters: one output channel each
- DAC independently or concurrently conversions

[Figure 11-1. DAC block diagram](#) shows the block diagram of DAC and [Table 11-1. DAC](#)

[pin](#) gives the pin description.

**Figure 11-1. DAC block diagram**



**Table 11-1. DAC pins**

Name	Description	Signal type
V <sub>DDA</sub>	Analog power supply	Input, analog supply
V <sub>SSA</sub>	Ground for analog power supply	Input, analog supply ground
DAC_OUTx	DACx analog output	Analog output signal

**Note:** The GPIO pin should first be configured to analog mode before enable the DAC module.

For GD32F150xx devices, corresponding GPIO pin is PA4.

For GD32F190xx devices, corresponding GPIO pin is PA4 or PA5.

## 11.3. Function overview

### 11.3.1. DAC enable

The DAC can be powered on by setting the DENx bit in the DAC\_CTL register. A *t<sub>WAKEUP</sub>* time is needed to startup the analog DAC submodule.

### 11.3.2. DAC output buffer

For the concern of reducing output impedance, and driving external loads without an external operational amplifier, an output buffer is integrated inside each DAC module.

The output buffer, which is turned on by default, can be turned off by setting the DBOFFx bit

in the DAC\_CTL register.

### 11.3.3. DAC data configuration

Depending on the selected configuration mode, the data has to be written in the specified register as described below:

- Single DAC, there are three possibilities

1. right alignment 8-bit: DHx[11:4] bits are stored into DACx\_R8DH [7:0] bits
2. right alignment 12-bit: DHx[11:0] bits are stored into DACx\_R12DH [11:0] bits
3. left alignment 12-bit: DHx[11:0] bits are stored into DACx\_L12DH [15:4] bits

The data can be shifted and stored into the DHx(Data Holding Register x, that are internal non-memory-mapped registers) according to the value of the DACx\_yyDH register. When a software trigger or an external event trigger occurs, the DH register will be loaded into DACx\_DO register automatically.

- Two DACs, there are three possibilities ( only for GD32F190xx devices)

1. Right alignment 8-bit: data for DAC0, DH0 [11:4] bits are stored into DACC\_R8DH [7:0] bits. Data for DAC1, DH1 [11:4] bits are stored into DACC\_R8DH [15:8] bits.
2. Right alignment 12-bit: data for DAC0 channel, DH0 [11:0] bits are stored into DACC\_R12DH [11:0] bits. Data for DAC1 channel, DH1 [11:0] bits are stored into DACC\_R12DH [27:16] bits.
3. Left alignment 12-bit: data for DAC0 channel, DH0 [11:0] bits are stored into DACC\_L12DH [15:4] bits. Data for DAC1 channel, DH1 [11:0] bits are stored into DACC\_L12DH [31:20] bits.

The data can be shifted and stored into the DAC0\_DH and DAC1\_DH (Data Holding Register x, that are internal non-memory-mapped registers) according to the value of the DACx\_yyDH register. When a software trigger or an external event trigger occurs, the DH register will be loaded into DAC0\_DO and DAC1\_DO register automatically.

### 11.3.4. DAC trigger

The DAC external trigger is enabled by setting the DTENx bits in the DAC\_CTL register. The DAC external triggers are selected by the DTSELx bit in the DAC\_CTL register.

**Table 11-2. External triggers of DAC**

DTSELx[2:0]	Trigger Source	Trigger Type
000	TIMER5_TRGO	Internal on-chip signal
001	TIMER2_TRGO	
010	Reserved	
011	TIMER14_TRGO	

DTSELx[2:0]	Trigger Source	Trigger Type
100	TIMER1_TRGO	
101	Reserved	
110	EXTI_9	External signal
111	SWTR	Software trigger

The TIMEx\_TRGO signals are generated from the timers, while the software trigger can be generated by setting the SWTRx bit in the DAC\_SWT register.

### 11.3.5. DAC conversion

If the external trigger enabled by setting the DTENx bit in DAC\_CTL register, the DAC holding data is transferred to the DAC output data (DACx\_DO) register at the selected trigger events. Otherwise, when the external trigger is disabled, the transfer is performed automatically.

When the DAC holding data (DACx\_DH) is loaded into the DACx\_DO register, after the time tSETTLING, the analog output is valid, and the value of tSETTLING is related to the power supply voltage and the analog output load.

### 11.3.6. DAC output voltage

The analog output voltages on the DAC pin are determined by the following equation:

$$\text{DAC output} = V_{DDA} * \text{DO} / 4096$$

The digital input is linearly converted to an analog output voltage, its range is 0 to  $V_{DDA}$ .

### 11.3.7. DMA request

Each DAC has a DMA function. For GD32F190xx devices, two DMA channel can be respectively used for DAC DMA requests.

When the external trigger is enabled, the DMA request is enabled by setting the DDMAENx bit of the DAC\_CTL register. A DAC DMA request will be generated when an external hardware trigger (not a software trigger) occurs.

If a second external trigger arrives before the acknowledgement of the previous request, the new request will not be serviced, and an underrun error event occurs. The DDUDRx bit in the DAC\_STAT register is set, an interrupt will be generated if the DDUDRIEx bit in the DAC\_CTL register is set. The DMA request will be stalled until the DDUDRx bit is cleared.

#### GD32F190xx devices:

In concurrent mode, if both DDMAENx bits are set, two DMA requests will be generated.

If you need one DMA request, only the corresponding DDMAENx bit should be set. In addition, the application can handle both DAC channel in concurrent mode using one DMA request and an independent DMA channel.

### 11.3.8. DAC concurrent conversion for GD32F190xx devices

When the two DACs work at the same time, for more effective utilization of bus bandwidth in applications, three concurrent registers can be used: DACC\_R8DH, DACC\_R12DH, DACC\_L12DH. You just need to access a unique register to realize driving both DAC channels at the same time.

For the two DACs and these special registers, several possible conversion modes are possible. The conversion mode in the case of only use one DAC channel, still can operate through independent DHx registers.

All modes are described in the paragraphs below.

#### Independent trigger

To configure the DAC in this conversion mode, the following sequence is required:

- Set the two DAC trigger enable bits DTEN0 and DTEN1
- Set different values in the DTSEL0[2:0] and DTSEL1[2:0] bits to configure different trigger sources
- Load the DACs channel data into the required DACC\_DH register (DACC\_R8DH, DACC\_R12DH, DACC\_L12DH)

When the DAC0 channel is triggered, the value of the DH0 register is transferred to DAC0\_DO (three APB1 clock cycles later).

When the DAC1 channel is triggered, the value of the DH1 register is transferred to DAC1\_DO (three APB1 clock cycles later).

#### Concurrent software start

In this conversion mode, follow the steps below to set the DAC:

- Load the DACs channel data to the required DACC\_DH register (DACC\_R8DH, DACC\_R12DH, DACC\_L12DH)

In this mode, after one APB1 clock cycle, the value of the DH1 and DH2 registers is immediately transferred to DAC0\_DO and DAC1\_DO.

#### Concurrent trigger

In this conversion mode, follow the steps below to set the DAC:

- Set the two DAC trigger enable bits DTEN0 and DTEN1
- Set the DTSEL0[2:0] and DTSEL1[2:0] bit to the same value, to configure the same trigger source for both DAC
- The data of two DACs conversion load to the required DACC\_DH register (DACC\_R8DH,

DACC\_R12DH, DACC\_L12DH)

If detects a rising edge of a trigger, the value of the DH1 and DH2 registers is immediately transferred to DAC0\_DO and DAC1\_DO (three APB1 clock cycles later).

## 11.4. Register definition

### 11.4.1. Control register (DAC\_CTL)

#### For GD32F150xx devices

Address offset: 0x00

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

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		DDUDRIE0	DDMAEN0	Reserved						DTSEL0[2:0]	DTEN0	DBOFF0	DENO		
		rw	rw							rw	rw	rw	rw		

Bits	Fields	Descriptions
31:14	Reserved	Must be kept at reset value
13	DDUDRIE0	DAC0 DMA Underrun Interrupt enable 0: DAC0 DMA Underrun Interrupt disabled 1: DAC0 DMA Underrun Interrupt enabled
12	DDMAEN0	DAC0 DMA enable 0: DAC0 DMA mode disabled 1: DAC0 DMA mode enabled
11:6	Reserved	Must be kept at reset value
5:3	DTSEL0[2:0]	DAC0 trigger selection These bits are only used if bit DTEN0 = 1 and select the external event used to trigger DAC0. 000: TIMER5 TRGO event 001: TIMER2 TRGO event 010: Reserved 011: TIMER14 TRGO event 100: TIMER1 TRGO event 101: Reserved 110: External line9 111: Software trigger
2	DTEN0	DAC0 trigger enable This bit is set and cleared by software to enable/disable DAC0 trigger. 0: DAC0 trigger disabled

		1: DAC0 trigger enabled
1	DBOFF0	DAC0 output buffer turn off This bit is set and cleared by software to enable/disable DAC0 output buffer. 0: DAC0 output buffer enabled 1: DAC0 output buffer turn off
0	DEN0	DAC0 enable This bit is set and cleared by software to enable/disable DAC0. 0: DAC0 disabled 1: DAC0 enabled

### For GD32F190xx devices

Address offset: 0x00

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		DDUDRIE1	DDMAEN1	Reserved				DTSEL1[2:0]			DTEN1	DBOFF1	DEN1		
		rw	rw					rw			rw	rw	rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		DDUDRIE0	DDMAEN0	Reserved				DTSEL0[2:0]			DTEN0	DBOFF0	DEN0		
		rw	rw					rw			rw	rw	rw		

Bits	Fields	Descriptions
31:30	Reserved	Must be kept at reset value
29	DDUDRIE1	DAC1 DMA Underrun Interrupt enable 0: DAC1 DMA Underrun Interrupt disabled 1: DAC1 DMA Underrun Interrupt enabled
28	DDMAEN1	DAC1 DMA enable 0: DAC1 DMA mode disabled 1: DAC1 DMA mode enabled
27:22	Reserved	Must be kept at reset value
21:19	DTSEL1[2:0]	DAC1 trigger selection These bits are only used if bit DTEN1 = 1 and select the external event used to trigger DAC1. 000: TIMER5 TRGO event 001: TIMER2 TRGO event 010: Reserved 011: TIMER14 TRGO event 100: TIMER1 TRGO event 101: Reserved

		110: External line9
		111: Software trigger
18	DTEN1	DAC1 trigger enable This bit is set and cleared by software to enable/disable DAC1 trigger. 0: DAC1 trigger disabled 1: DAC1 trigger enabled
17	DBOFF1	DAC1 output buffer turn off This bit is set and cleared by software to enable/disable DAC1 output buffer. 0: DAC1 output buffer enabled 1: DAC1 output buffer turn off
16	DEN1	DAC1 enable This bit is set and cleared by software to enable/disable DAC1. 0: DAC1 disabled 1: DAC1 enabled
15:14	Reserved	Must be kept at reset value
13	DDUDRIE0	DAC0 DMA Underrun Interrupt enable 0: DAC0 DMA Underrun Interrupt disabled 1: DAC0 DMA Underrun Interrupt enabled
12	DDMAEN0	DAC0 DMA enable 0: DAC0 DMA mode disabled 1: DAC0 DMA mode enabled
11:6	Reserved	Must be kept at reset value
5:3	DTSEL0[2:0]	DAC0 trigger selection These bits are only used if bit DTEN0 = 1 and select the external event used to trigger DAC0. 000: TIMER5 TRGO event 001: TIMER2 TRGO event 010: Reserved 011: TIMER14 TRGO event 100: TIMER1 TRGO event 101: Reserved 110: External line9 111: Software trigger
2	DTEN0	DAC0 trigger enable This bit is set and cleared by software to enable/disable DAC0 trigger. 0: DAC0 trigger disabled 1: DAC0 trigger enabled
1	DBOFF0	DAC0 output buffer turn off This bit is set and cleared by software to enable/disable DAC0 output buffer.

		0: DAC0 output buffer enabled 1: DAC0 output buffer turn off
0	DEN0	DAC0 enable This bit is set and cleared by software to enable/disable DAC0. 0: DAC0 disabled 1: DAC0 enabled

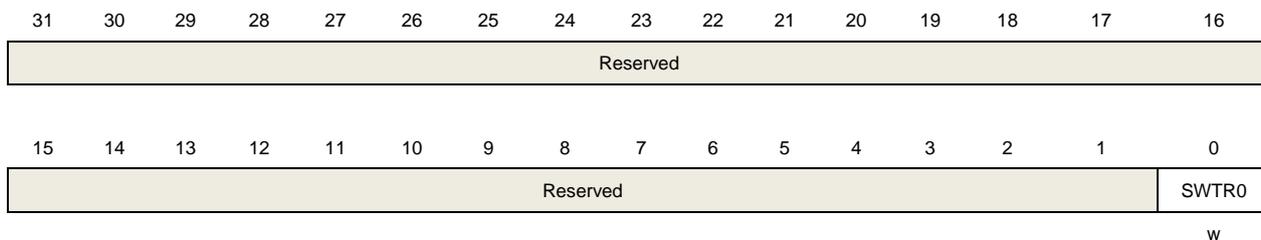
## 11.4.2. Software trigger register (DAC\_SWT)

### For GD32F150xx devices

Address offset: 0x04

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)



w

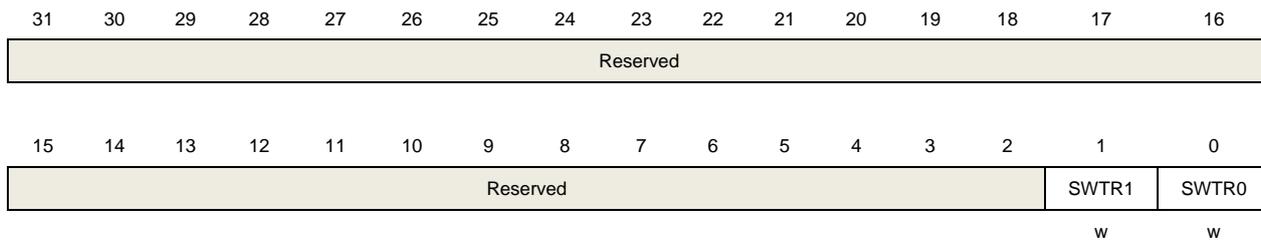
Bits	Fields	Descriptions
31:1	Reserved	Must be kept at reset value
0	SWTR0	DAC0 software trigger, cleared by hardware 0: Software trigger disabled 1: Software trigger enabled

### For GD32F190xx devices

Address offset: 0x04

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)



w

w

Bits	Fields	Descriptions
------	--------	--------------

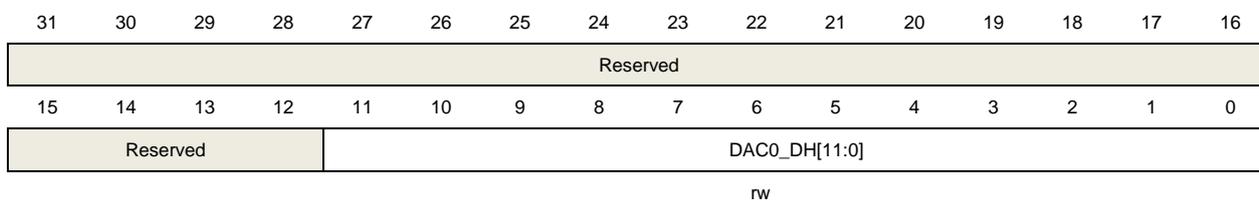
31:2	Reserved	Must be kept at reset value
1	SWTR1	DAC1 software trigger, cleared by hardware 0: Software trigger disabled 1: Software trigger enabled
0	SWTR0	DAC0 software trigger, cleared by hardware 0: Software trigger disabled 1: Software trigger enabled

### 11.4.3. DAC0 12-bit right-aligned data holding register (DAC0\_R12DH)

Address offset: 0x08

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)



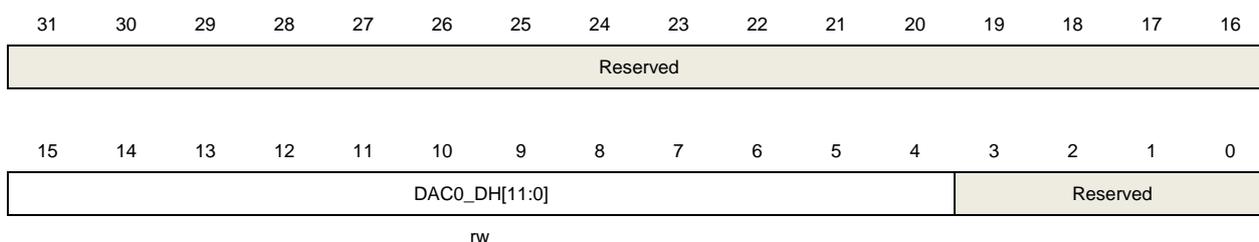
Bits	Fields	Descriptions
31:12	Reserved	Must be kept at reset value
11:0	DAC0_DH[11:0]	DAC0 12-bit right-aligned data These bits specify the data that is to be converted by DAC0.

### 11.4.4. DAC0 12-bit left-aligned data holding register (DAC0\_L12DH)

Address offset: 0x0C

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)



Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value

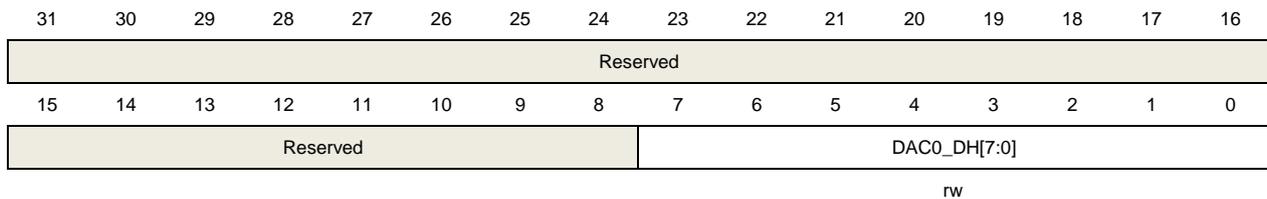
15:4	DAC0_DH[11:0]	DAC0 12-bit left-aligned data These bits specify the data that is to be converted by DAC0.
3:0	Reserved	Must be kept at reset value

#### 11.4.5. DAC0 8-bit right-aligned data holding register (DAC0\_R8DH)

Address offset: 0x10

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)



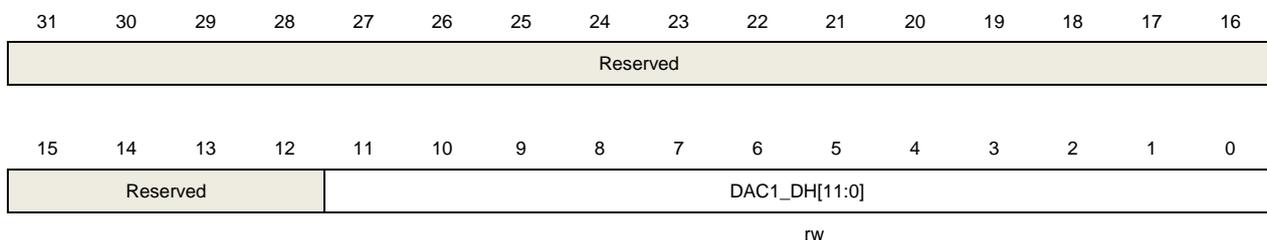
Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value
7:0	DAC0_DH[7:0]	DAC0 8-bit right-aligned data These bits specify the data that is to be converted by DAC0.

#### 11.4.6. DAC1 12-bit right-aligned data holding register (DAC1\_R12DH) of GD32F190xx devices

Address offset: 0x14

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)



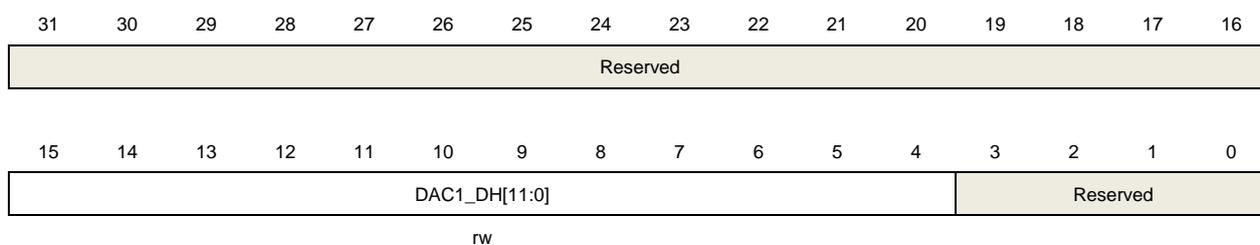
Bits	Fields	Descriptions
31:12	Reserved	Must be kept at reset value
11:0	DAC1_DH[11:0]	DAC1 12-bit right-aligned data These bits specify the data that is to be converted by DAC1.

### 11.4.7. DAC1 12-bit left-aligned data holding register (DAC1\_L12DH) of GD32F190xx devices

Address offset: 0x18

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)



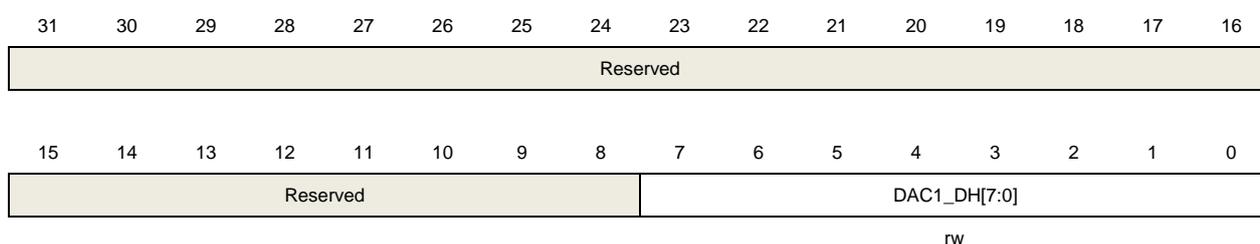
Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:4	DAC1_DH[11:0]	DAC1 12-bit left-aligned data These bits specify the data that is to be converted by DAC1.
3:0	Reserved	Must be kept at reset value

### 11.4.8. DAC1 8-bit right-aligned data holding register (DAC1\_R8DH) of GD32F190xx devices

Address offset: 0x1C

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)



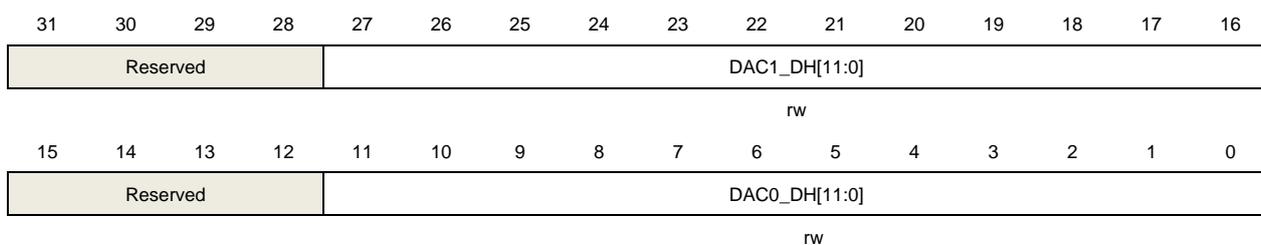
Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value
7:0	DAC1_DH[7:0]	DAC1 8-bit right-aligned data These bits specify the data that is to be converted by DAC1.

### 11.4.9. DAC concurrent mode 12-bit right-aligned data holding register (DACC\_R12DH) of GD32F190xx devices

Address offset: 0x20

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)



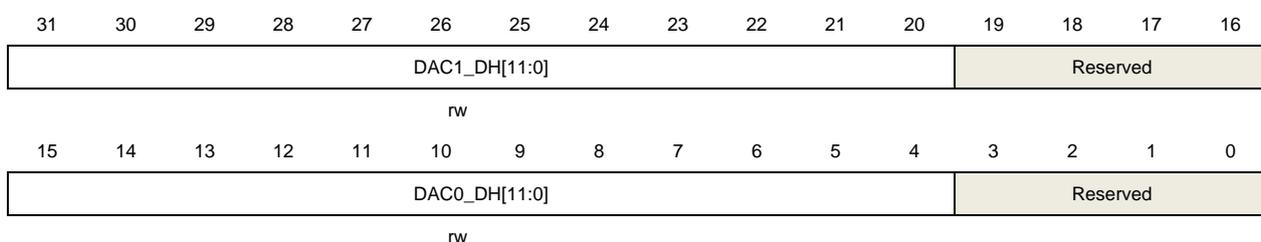
Bits	Fields	Descriptions
31:28	Reserved	Must be kept at reset value
27:16	DAC1_DH[11:0]	DAC1 12-bit right-aligned data These bits specify the data that is to be converted by DAC1.
15:12	Reserved	Must be kept at reset value
11:0	DAC0_DH[11:0]	DAC0 12-bit right-aligned data These bits specify the data that is to be converted by DAC0.

### 11.4.10. DAC concurrent mode 12-bit left-aligned data holding register (DACC\_L12DH) of GD32F190xx devices

Address offset: 0x24

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)



Bits	Fields	Descriptions
31:20	DAC1_DH[11:0]	DAC1 12-bit left-aligned data These bits specify the data that is to be converted by DAC1.

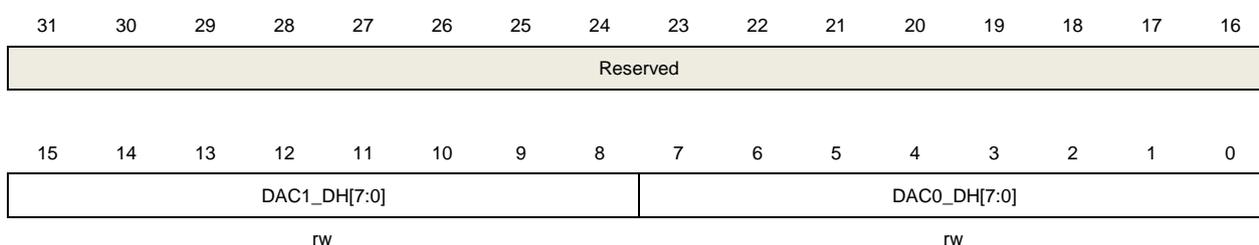
19:16	Reserved	Must be kept at reset value
15:4	DAC0_DH[11:0]	DAC0 12-bit left-aligned data These bits specify the data that is to be converted by DAC0.
3:0	Reserved	Must be kept at reset value

#### 11.4.11. DAC concurrent mode 8-bit right-aligned data holding register (DACC\_R8DH) of GD32F190xx devices

Address offset: 0x28

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)



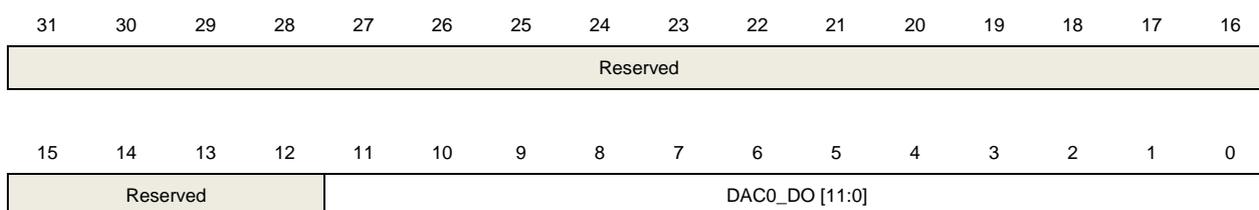
Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:8	DAC1_DH[7:0]	DAC1 8-bit right-aligned data These bits specify the data that is to be converted by DAC1.
7:0	DAC0_DH[7:0]	DAC0 8-bit right-aligned data These bits specify the data that is to be converted by DAC0.

#### 11.4.12. DAC0 data output register (DAC0\_DO)

Address offset: 0x2C

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)



r

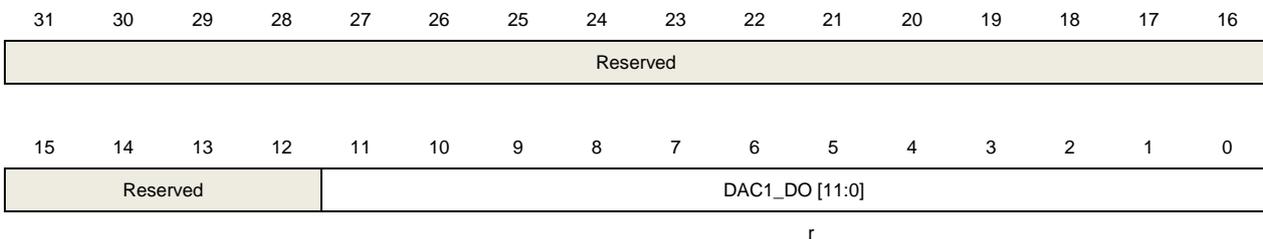
Bits	Fields	Descriptions
31:12	Reserved	Must be kept at reset value
11:0	DAC0_DO [11:0]	DAC0 output data These bits, which are read-only, reflect the data that is being converted by DAC0.

#### 11.4.13. DAC1 data output register (DAC1\_DO) of GD32F190xx devices

Address offset: 0x30

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)



Bits	Fields	Descriptions
31:12	Reserved	Must be kept at reset value
11:0	DAC1_DO [11:0]	DAC1 output data These bits, which are read-only, reflect the data that is being converted by DAC1.

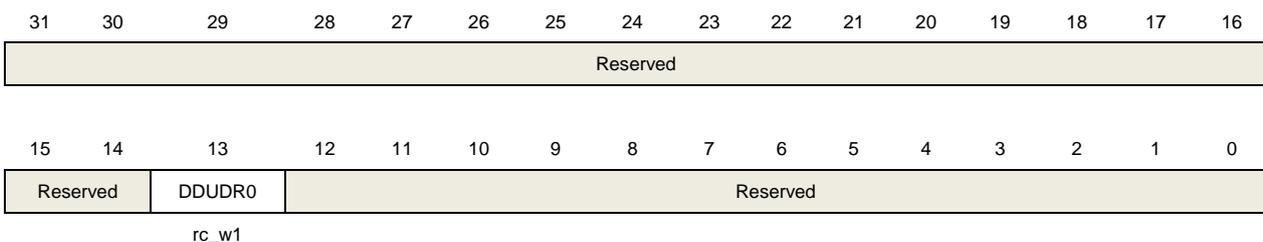
#### 11.4.14. Status register (DAC\_STAT)

**For GD32F150xx devices**

Address offset: 0x34

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)



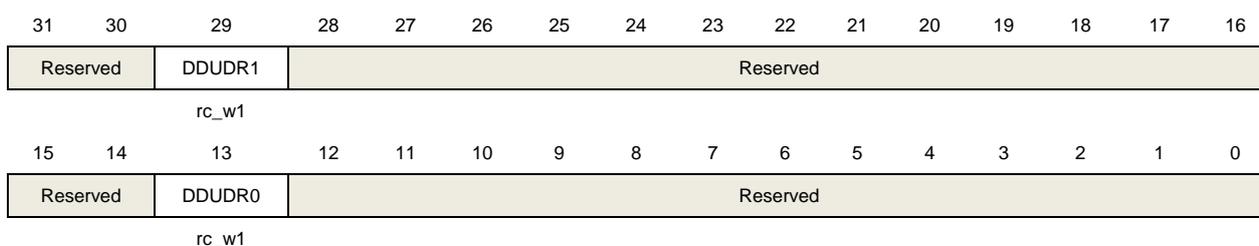
Bits	Fields	Descriptions
31:14	Reserved	Must be kept at reset value
13	DDUDR0	DAC0 DMA underrun flag This bit is set by hardware and cleared by software (by writing it to 1). 0: No DMA underrun error condition occurred 1: DMA underrun error condition occurred (the frequency of the current selected trigger that is driving DAC conversion is higher than the DMA service capability rate)
12:0	Reserved	Must be kept at reset value

### For GD32F190xx devices

Address offset: 0x34

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)



Bits	Fields	Descriptions
31:30	Reserved	Must be kept at reset value
29	DDUDR1	DAC1 DMA underrun flag This bit is set by hardware and cleared by software (by writing it to 1). 0: No DMA underrun error condition occurred 1: DMA underrun error condition occurred (the frequency of the current selected trigger that is driving DAC conversion is higher than the DMA service capability rate)
28:14	Reserved	Must be kept at reset value
13	DDUDR0	DAC0 DMA underrun flag This bit is set by hardware and cleared by software (by writing it to 1). 0: No DMA underrun error condition occurred 1: DMA underrun error condition occurred (the frequency of the current selected trigger that is driving DAC conversion is higher than the DMA service capability rate)
12:0	Reserved	Must be kept at reset value

## 12. Comparator (CMP)

### 12.1. Overview

The general purpose comparators, CMP0 and CMP1, can work either standalone (all terminal are available on I/Os) or together with the timers.

It could be used to wake up the MCU from low-power mode by an analog signal, provide a trigger source when an analog signal is in a certain condition, achieves some current control by working together with a PWM output of a timer and the DAC.

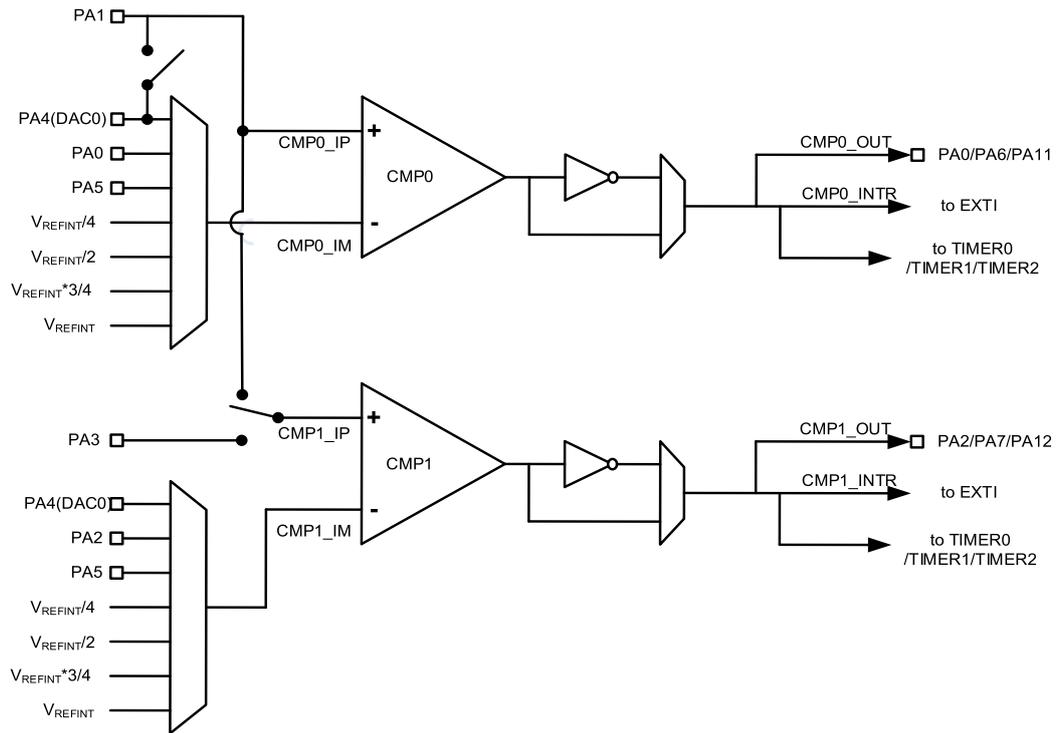
### 12.2. Characteristics

- Rail-to-rail comparators
- Configurable hysteresis
- Configurable speed and consumption
- Each comparator has configurable analog input source
  - DAC
  - 3 I/O pins
  - The whole or sub-multiple values of internal reference voltage
- Window comparator
- Outputs to I/O
- Outputs to timers for triggering
- Outputs to EXTI

### 12.3. Function overview

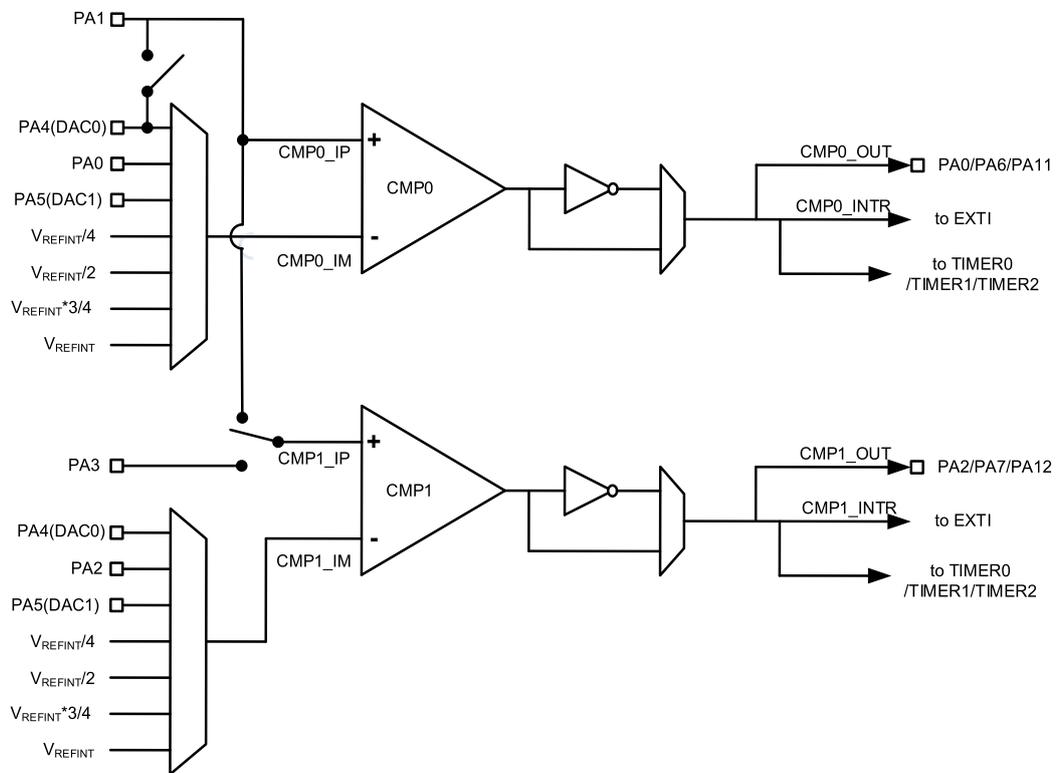
The block diagrams of CMP are shown as below.

**Figure 12-1. CMP block diagram of GD32F130xx and GD32F150xx devices**



**Note:**  $V_{REFINT}$  is 1.2V.

**Figure 12-2. CMP block diagram of GD32F170xx and GD32F190xx devices**



**Note:**  $V_{REFINT}$  is 1.2V.

### 12.3.1. CMP clock and reset

The CMP clock provided by the clock controller is synchronous with the PCLK. The CMP share common reset and clock enable bits with SYSCFG.

### 12.3.2. CMP inputs and outputs

These I/Os must be configured in analog mode in the GPIOs registers before they are selected as CMPs inputs.

Considering pin definitions in Datasheet, the CMP output must be connected to corresponding alternate I/Os.

A variety of timer inputs can be internally connected to the CMP output to realize the following functions:

- Input capture for timing measures
- Emergency shut-down of PWM signals, using BKIN
- Cycle-by-cycle current control, using OCPRE\_CLR inputs

In order to work even in Deep-sleep mode, the polarity selection logic and the output redirection to the port work independently from PCLK.

It is possible to have the CMP output simultaneously redirected internally and externally.

The CMP outputs are internally connected to the extended interrupts and events controller. Each CMP has its own EXTI line and can generate either interrupts or events. The same mechanism is used to exit from power saving modes.

### 12.3.3. CMP power mode

For a given application, there is a trade-off between the CMP power consumption versus propagation delay, which is adjusted by configuring bits CMPxM [1:0] in CMP\_CS register. The CMP works fastest with highest power consumption when  $CMPxM = 2'b00$ , while works slowest with lowest power consumption when  $CMPxM = 2'b11$

### 12.3.4. CMP hysteresis

In order to avoid spurious output transitions that caused by the noise signal, a programmable hysteresis is designed to force the hysteresis value using external components. This function can be shut down when you don't need it.

### **12.3.5. CMP register write protection**

The CMP control and status register (CMP\_CS) can be protected from writing by setting CMPxLK bit to 1. The CMP\_CS register, including the CMPxLK bit will be read-only, and can only be reset by the MCU reset.

This write protection function is useful in some applications, such as thermal protection and over-current protection.

## 12.4. Register definition

### 12.4.1. Control/status register (CMP\_CS)

For GD32F130xx and GD32F150xx devices

Address offset: 0x00

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CMP1LK	CMP1O	CMP1HST[1:0]		CMP1PL	CMP1OSEL[2:0]			WNDEN	CMP1MSEL[2:0]			CMP1M		Reserved	CMP1EN
rwo	r	rw/r		rw/r	rw/r			rw/r	rw/r			rw/r			rw/r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP0LK	CMP0O	CMP0HST[1:0]		CMP0PL	CMP0OSEL[2:0]			Reserved	CMP0MSEL[2:0]			CMP0M[1:0]		CMP0SW	CMP0EN
rwo	r	rw/r		rw/r	rw/r				rw/r			rw/r		rw/r	rw/r

Bits	Fields	Descriptions
31	CMP1LK	CMP1 lock This bit allows to have all control bits of CMP1 as read-only. This bit is write-once. It can only be cleared by a system reset once It is set by software. 0: CMP_CS[31:16] bits are read-write 1: CMP_CS[31:16] bits are read-only
30	CMP1O	CMP1 output This is a copy of CMP1 output state, which is read only. 0: Non-inverting input below inverting input and the output is low 1: Non-inverting input above inverting input and the output is high
29:28	CMP1HST[1:0]	CMP1 hysteresis These bits are used to control the hysteresis level. 00: No hysteresis 01: Low hysteresis 10: Medium hysteresis 11: High hysteresis
27	CMP1PL	Polarity of CMP1 output This bit is used to select the CMP1 output. 0: Output is not inverted 1: Output is inverted
26:24	CMP1OSEL[2:0]	CMP1 output selection These bits are used to select the destination of the CMP1 output. 000: No selection

		001: TIMER 0 break input
		010: TIMER 0 channel0 Input capture
		011: TIMER 0 OCPRE_CLR input
		100: TIMER1 channel3 input capture
		101: TIMER1 OCPRE_CLR input
		110: TIMER2 channel0 input capture
		111: TIMER2 OCPRE_CLR input
23	WNDEN	Window mode enable This bit is used to disconnect the CMP1_IP input of CMP1 from PA3 and connect it to CMP0's CMP0_IP input. 0: CMP1_IP is connected to PA3 1: CMP1_IP is connected to CMP0_IP
22:20	CMP1MSEL[2:0]	CMP1_IM input selection These bits are used to select the source connected to the CMP1_IM input of the CMP1. 000: $V_{REFINT}/4$ 001: $V_{REFINT}/2$ 010: $V_{REFINT} * 3/4$ 011: $V_{REFINT}$ 100: PA4 (DAC0) 101: PA5 110: PA2 111: Reserved
19:18	CMP1M[1:0]	CMP1 mode These bits are used to control the operating mode of the CMP1 adjust the speed/consumption. 00: High speed / full power 01: Medium speed / medium power 10: Low speed / low power 11: Very-low speed / ultra-low power
17	Reserved	Must be kept at reset value
16	CMP1EN	CMP1 enable 0: CMP1 disabled 1: CMP1 enabled
15	CMP0LK	CMP0 lock This bit allows to have all control bits of CMP0 as read-only. This bit is write-once. It can only be cleared by a system reset once It is set by software. 0: CMP_CS[15:0] bits are read-write 1: CMP_CS[15:0] bits are read-only
14	CMP0O	CMP0 output This is a copy of CMP0 output state, which is read only.

		0: Non-inverting input below inverting input and the output is low 1: Non-inverting input above inverting input and the output is high
13:12	CMP0HST[1:0]	CMP0 hysteresis These bits are used to control the hysteresis level. 00: No hysteresis 01: Low hysteresis 10: Medium hysteresis 11: High hysteresis
11	CMP0PL	Polarity of CMP0 output This bit is used to select the CMP0 output. 0 : Output is not inverted 1 : Output is inverted
10:8	CMP0OSEL[2:0]	Comparator 0 output selection These bits are used to select the destination of the CMP0 output. 000: no selection 001: TIMER 0 break input 010: TIMER 0 channel0 Input capture 011: TIMER 0 OCPRE_CLR input 100: TIMER1 channel3 input capture 101: TIMER1 OCPRE_CLR input 110: TIMER2 channel0 input capture 111: TIMER2 OCPRE_CLR input
7	Reserved	Must be kept at reset value
6:4	CMP0MSEL[2:0]	CMP0_IM input selection These bits are used to select the source connected to the CMP0_IM input of the CMP0. 000: $V_{REFINT} / 4$ 001: $V_{REFINT} / 2$ 010: $V_{REFINT} * 3/4$ 011: $V_{REFINT}$ 100: PA4 (DAC0) 101: PA5 110: PA0 111: Reserved
3:2	CMP0M[1:0]	CMP0 mode These bits are used to control the operating mode of the CMP0 adjust the speed/consumption. 00: High speed/ full power 01: Medium speed/ medium power 10: Low speed/ low power 11: Very-low speed/ ultra-low power

1	CMP0SW	CMP0 switch This bit is used to closes a switch between CMP0 non-inverting input on PA0 and PA4 (DAC0) I/O. 0: Switch open 1: Switch closed
0	CMP0EN	CMP0 enable 0: CMP0 disabled 1: CMP0 enabled

### For GD32F170xx and GD32F190xx devices

Address offset: 0x00

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CMP1LK	CMP1O	CMP1HST[1:0]		CMP1PL	CMP1OSEL[2:0]			WNDEN	CMP1MSEL[2:0]			CMP1M		Reserved	CMP1EN
rwo	r	rw/r		rw/r	rw/r			rw/r	rw/r			rw/r			rw/r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP0LK	CMP0O	CMP0HST[1:0]		CMP0PL	CMP0OSEL[2:0]			Reserved	CMP0MSEL[2:0]			CMP0M[1:0]		CMP0SW	CMP0EN
rwo	r	rw/r		rw/r	rw/r				rw/r			rw/r		rw/r	rw/r

Bits	Fields	Descriptions
31	CMP1LK	CMP1 lock This bit allows to have all control bits of CMP1 as read-only. This bit is write-once. It can only be cleared by a system reset once It is set by software. 0: CMP_CS[31:16] bits are read-write 1: CMP_CS[31:16] bits are read-only
30	CMP1O	CMP1 output This is a copy of CMP1 output state, which is read only. 0: Non-inverting input below inverting input and the output is low 1: Non-inverting input above inverting input and the output is high
29:28	CMP1HST[1:0]	CMP1 hysteresis These bits are used to control the hysteresis level. 00: No hysteresis 01: Low hysteresis 10: Medium hysteresis 11: High hysteresis
27	CMP1PL	Polarity of CMP1 output This bit is used to select the CMP1 output.

		0: Output is not inverted 1: Output is inverted
26:24	CMP1OSEL[2:0]	<p>CMP1 output selection</p> <p>These bits are used to select the destination of the CMP1 output.</p> <p>000: No selection 001: TIMER 0 break input 010: TIMER 0 channel0 Input capture 011: TIMER 0 OCPRE_CLR input 100: TIMER1 channel3 input capture 101: TIMER1 OCPRE_CLR input 110: TIMER2 channel0 input capture 111: TIMER2 OCPRE_CLR input</p>
23	WNDEN	<p>Window mode enable</p> <p>This bit is used to disconnect the CMP1_IP input of CMP1 from PA3 and connect it to CMP0's CMP0_IP input.</p> <p>0: CMP1_IP is connected to PA3 1: CMP1_IP is connected to CMP0_IP</p>
22:20	CMP1MSEL[2:0]	<p>CMP1_M input selection</p> <p>These bits are used to select the source connected to the CMP1_M input of the CMP1.</p> <p>000: <math>V_{REFINT}/4</math> 001: <math>V_{REFINT}/2</math> 010: <math>V_{REFINT} * 3/4</math> 011: <math>V_{REFINT}</math> 100: PA4 (DAC0) 101: PA5 (DAC1) 110: PA2 111: Reserved</p>
19:18	CMP1M[1:0]	<p>CMP1 mode</p> <p>These bits are used to control the operating mode of the CMP1 adjust the speed/consumption.</p> <p>00: High speed / full power 01: Medium speed / medium power 10: Low speed / low power 11: Very-low speed / ultra-low power</p>
17	Reserved	Must be kept at reset value
16	CMP1EN	<p>CMP1 enable</p> <p>0: CMP1 disabled 1: CMP1 enabled</p>
15	CMP0LK	<p>CMP0 lock</p> <p>This bit allows to have all control bits of CMP0 as read-only. This bit is write-once. It can</p>

		only be cleared by a system reset once It is set by software.
		0: CMP_CS[15:0] bits are read-write
		1: CMP_CS[15:0] bits are read-only
14	CMP0O	<p>CMP0 output</p> <p>This is a copy of CMP0 output state, which is read only.</p> <p>0: Non-inverting input below inverting input and the output is low</p> <p>1: Non-inverting input above inverting input and the output is high</p>
13:12	CMP0HST[1:0]	<p>CMP0 hysteresis</p> <p>These bits are used to control the hysteresis level.</p> <p>00: No hysteresis</p> <p>01: Low hysteresis</p> <p>10: Medium hysteresis</p> <p>11: High hysteresis</p>
11	CMP0PL	<p>Polarity of CMP0 output</p> <p>This bit is used to select the CMP0 output.</p> <p>0 : Output is not inverted</p> <p>1 : Output is inverted</p>
10:8	CMP0OSEL[2:0]	<p>CMP0 output selection</p> <p>These bits are used to select the destination of the CMP0 output.</p> <p>000: no selection</p> <p>001: TIMER 0 break input</p> <p>010: TIMER 0 channel0 Input capture</p> <p>011: TIMER 0 OCPRE_CLR input</p> <p>100: TIMER1 channel3 input capture</p> <p>101: TIMER1 OCPRE_CLR input</p> <p>110: TIMER2 channel0 input capture</p> <p>111: TIMER2 OCPRE_CLR input</p>
7	Reserved	Must be kept at reset value
6:4	CMP0MSEL[2:0]	<p>CMP0_M input selection</p> <p>These bits are used to select the source connected to the CMP0_M input of the CMP0.</p> <p>000: <math>V_{REFINT} / 4</math></p> <p>001: <math>V_{REFINT} / 2</math></p> <p>010: <math>V_{REFINT} * 3/4</math></p> <p>011: <math>V_{REFINT}</math></p> <p>100: PA4 (DAC0)</p> <p>101: PA5 (DAC1)</p> <p>110: PA0</p> <p>111: Reserved</p>
3:2	CMP0M[1:0]	<p>CMP0 mode</p> <p>These bits are used to control the operating mode of the CMP0 adjust the</p>

		speed/consumption. 00: High speed/ full power 01: Medium speed/ medium power 10: Low speed/ low power 11: Very-low speed/ ultra-low power
1	CMP0SW	CMP0 switch This bit is used to closes a switch between CMP0 non-inverting input on PA0 and PA4 (DAC0) I/O. 0: Switch open 1: Switch closed
0	CMP0EN	CMP0 enable 0: CMP0 disabled 1: CMP0 enabled

## 13. Watchdog timer (WDGT)

The watchdog timer (WDGT) is a hardware timing circuitry that can be used to detect system failures due to software malfunctions. There are two watchdog timer peripherals in the chip: free watchdog timer (FWDGT) and window watchdog timer (WWDGT). They offer a combination of a high safety level, flexibility of use and timing accuracy. Both watchdog timers are offered to resolve malfunctions of software.

The watchdog will generate a reset (or an interrupt in window watchdog timer) when the internal counter reaches a given value. The watchdog timer counter can be stopped while the processor is in the debug mode.

### 13.1. Free watchdog timer (FWDGT)

#### 13.1.1. Overview

The free watchdog timer (FWDGT) has free clock source (IRC40K). Thereupon the FWDGT can operate even if the main clock fails. It's suitable for the situation that requires an independent environment and lower timing accuracy.

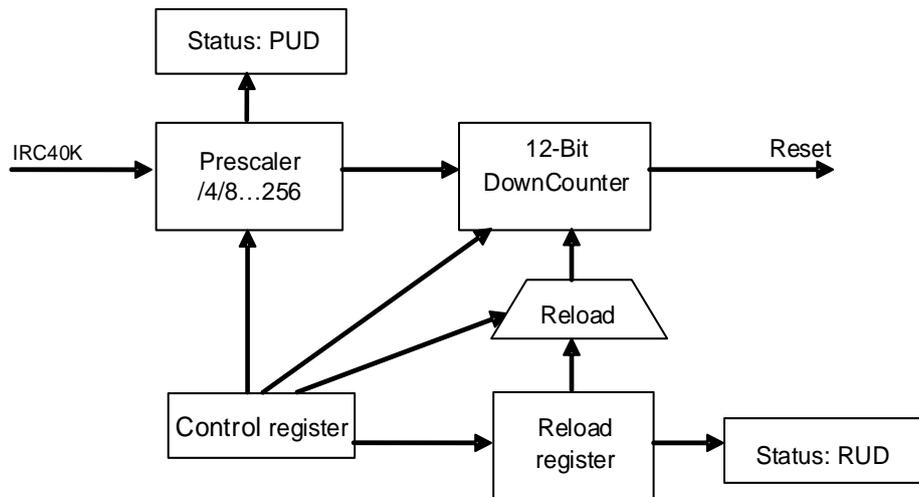
The free watchdog timer causes a reset when the internal down counter reaches 0. The register write protection function in free watchdog can be enabled to prevent it from changing the configuration unexpectedly.

#### 13.1.2. Characteristics

- Free-running 12-bit downcounter.
- Reset when the downcounter reaches 0, if the watchdog is enabled.
- Free clock source, FWDGT can operate even if the main clock fails such as in standby and Deep-sleep modes.
- Hardware free watchdog timer bit, automatically start the FWDGT at power on.
- FWDGT debug mode, the FWDGT can stop or continue to work in debug mode.

#### 13.1.3. Function overview

The free watchdog timer consists of an 8-stage prescaler and a 12-bit down-counter. Refer to the [Figure 13-1. Free watchdog timer block diagram](#) for the functional blocks of the free watchdog timer module.

**Figure 13-1. Free watchdog timer block diagram**


The free watchdog timer is enabled by writing the value 0xCCCC in the control register (FWDGT\_CTL), and the counter starts counting down. When the counter reaches the value 0x000, a reset is generated.

The counter can be reloaded by writing the value 0xAAAA to the FWDGT\_CTL register at any time. The reload value comes from the FWDGT\_RLD register. The software can prevent the watchdog reset by reloading the counter before the counter reaches the value 0x000.

By setting the appropriate window in the FWDGT\_WND register, the FWDGT can also work as a window watchdog timer. A reset will occur if the reload operation is performed while the counter is greater than the value stored in the window register (FWDGT\_WND). The default value of the FWDGT\_WND is 0x0000 0FFF, so if it is not updated, the window option is disabled. A reload operation is performed in order to reset the downcounter to the FWDGT\_RLD value and the prescaler counter to generate the next reload, as soon as the window value is changed.

The free watchdog timer can automatically start at power on when the hardware free watchdog timer bit in the device option bits is set. To avoid reset, the software should reload the counter before the counter reaches 0x000.

The FWDGT\_PSC register, the FWDGT\_RLD register and the FWDGT\_WND register are write protected. Before writing these registers, the software should write the value 0x5555 in the control register. These registers will be protected again by writing any other value in the control register. When an update of the prescaler (FWDGT\_PSC) or watchdog counter window value (FWDGT\_WND) or the reload value (FWDGT\_RLD) is on going, the status bit in FWDGT\_STAT register is set.

If the FWDGT\_HOLD bit in DBG module is cleared, the FWDGT continues to work even the Cortex™-M3 core halted (Debug mode). While the FWDGT stops in Debug mode if the FWDGT\_HOLD bit is set.

**Table 13-1. Min/max FWDGT timeout period at 40 kHz (IRC40K)**

Prescaler divider	PSC[2:0] bits	Min timeout (ms) RL[11:0]= 0x000	Max timeout (ms) RL[11:0]= 0xFFF
1/4	000	0.1	409.6
1/8	001	0.2	819.2
1/16	010	0.4	1638.4
1/32	011	0.8	3276.8
1/64	100	1.6	6553.6
1/128	101	3.2	13107.2
1/256	110 or 111	6.4	26214.4

The FWDGT timeout can be more accurately by calibrating the IRC40K.

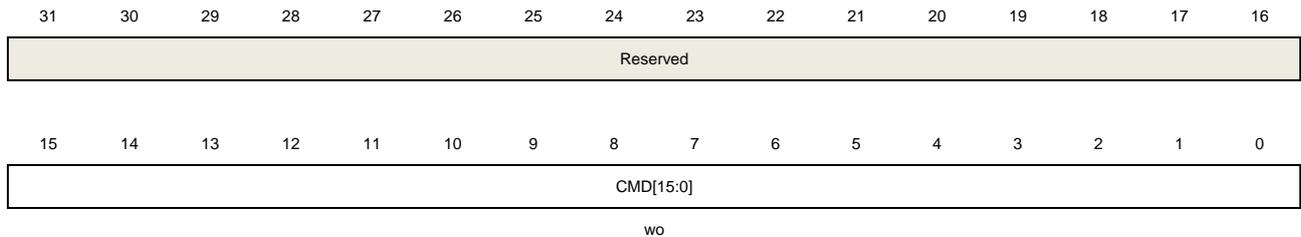
## 13.1.4. Register definition

### Control register (FWDGT\_CTL)

Address offset: 0x00

Reset value: 0x0000 0000

This register can be accessed by half-word (16-bit) or word (32-bit) access



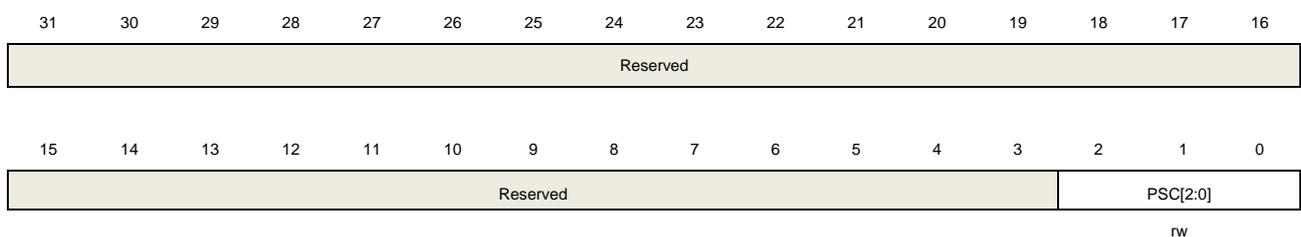
Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	CMD[15:0]	Write only. These bits have different functions when writing different values 0x5555: Disable the FWDGT_PSC, FWDGT_RLD and FWDGT_WND write protection 0xCCCC: Start the free watchdog timer counter. When the counter reduces to 0, the free watchdog timer generates a reset 0xAAAA: Reload the counter

### Prescaler register (FWDGT\_PSC)

Address offset: 0x04

Reset value: 0x0000 0000

This register can be accessed by half-word (16-bit) or word (32-bit) access



Bits	Fields	Descriptions
31:3	Reserved	Must be kept at reset value
2:0	PSC[2:0]	Free watchdog timer prescaler selection. Write 0x5555 in the FWDGT_CTL register before writing these bits. When a write operation to this register ongoing, the PUD bit in the FWDGT_STAT register is set and the value read from this register is invalid. 000: 1/4                      001: 1/8                      010: 1/16

011: 1/32                      100: 1/64                      101: 1/128  
 110: 1/256                     111: 1/256

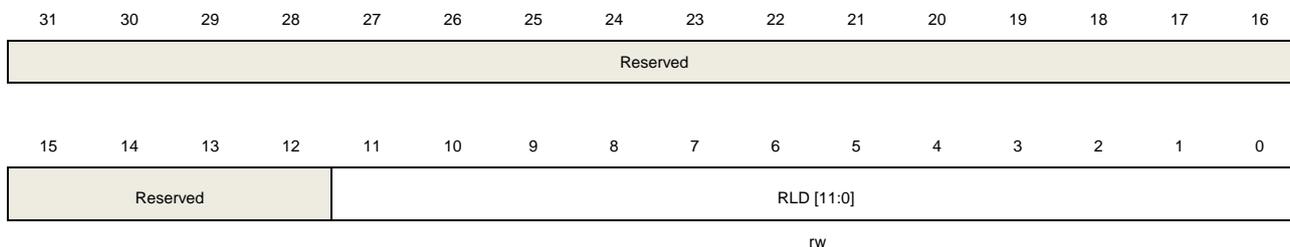
If several prescaler values are used by the application, it is mandatory to wait until PUD bit is reset before changing the prescaler value. However, after updating the prescaler value it is not necessary to wait until PUD is reset before continuing code execution, (for GD32F130xx and GD32F150xx, before entering low-power mode, it is necessary to wait until PUD is reset).

## Reload register (FWDGT\_RLD)

Address offset: 0x08

Reset value: 0x0000 0FFF

This register can be accessed by half-word (16-bit) or word (32-bit) access



Bits	Fields	Descriptions
31:12	Reserved	Must be kept at reset value
11:0	RLD[11:0]	Free watchdog timer counter reload value. Write 0xAAAA in the FWDGT_CTL register will reload the FWDGT conter. These bits are write-protected. Write 0X5555 in the FWDGT_CTL register before writing these bits. When a write operation to this register ongoing, the RUD bit in the FWDGT_STAT register is set and the value read from this register is invalid. If several reload values are used by the application, it is mandatory to wait until RUD bit is reset before changing the reload value. However, after updating the reload value it is not necessary to wait until RUD is reset before continuing code execution (for GD32F130xx and GD32F150xx, before entering low-power mode, it is necessary to wait until RUD is reset).

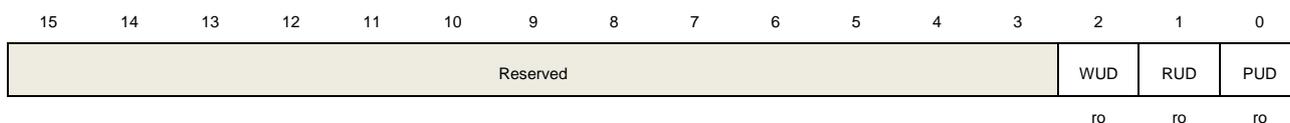
## Status register (FWDGT\_STAT)

Address offset: 0x0C

Reset value: 0x0000 0000

This register can be accessed by half-word (16-bit) or word (32-bit) access





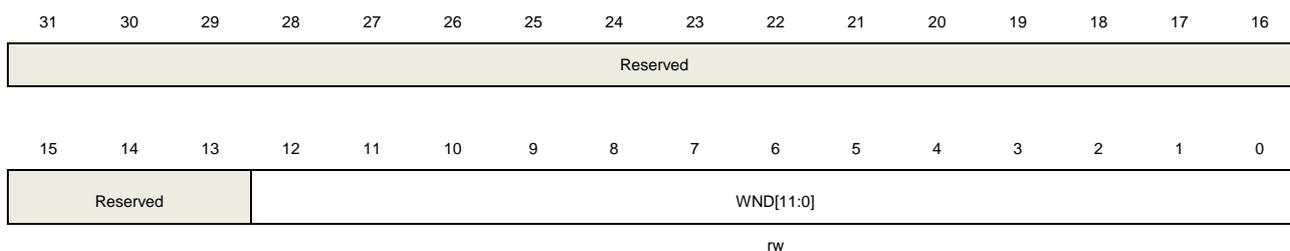
Bits	Fields	Descriptions
31:2	Reserved	Must be kept at reset value
2	WUD	Watchdog counter window value update When a write operation to FWDGT_WND register ongoing, this bit is set and the value read from FWDGT_WND register is invalid.
1	RUD	Free watchdog timer counter reload value update When a write operation to FWDGT_RLD register ongoing, this bit is set and the value read from FWDGT_RLD register is invalid.
0	PUD	Free watchdog timer prescaler value update When a write operation to FWDGT_PSC register ongoing, this bit is set and the value read from FWDGT_PSC register is invalid.

### Window register (FWDGT\_WND)

Address offset: 0x10

Reset value: 0x0000 0000

This register can be accessed by half-word (16-bit) or word (32-bit) access



Bits	Fields	Descriptions
31:12	Reserved	Must be kept at reset value
11:0	WND[11:0]	Watchdog counter window value. These bits are used to contain the high limit of the window value to be compared to the downcounter. A reset will occur if the reload operation is performed while the counter is greater than the value stored in this register. The WUD bit in the FWDGT_STAT register must be reset in order to be able to change the reload value.  These bits are write protected. Write 5555h in the FWDGT_CTL register before writing these bits.

If several window values are used by the application, it is mandatory to wait until WUD bit is reset before changing the window value. However, after updating the window value it is not necessary to wait until WUD is reset before continuing code execution except in case of low-power mode entry.

## 13.2. Window watchdog timer (WWDGT)

### 13.2.1. Overview

The window watchdog timer (WWDGT) is used to detect system failures due to software malfunctions. After the window watchdog timer starts, the value of downcounter reduce progressively. The watchdog timer causes a reset when the counter reached 0x3F (the CNT[6] bit becomes cleared). The watchdog timer also cause a reset if the counter is refreshed before the counter reached the window register value. So the software should refresh the counter in a limited window. The window watchdog timer generates an early wakeup status flag when the counter reaches 0x40 or refreshes before the counter reaches the window value. Interrupt occurs if it is enabled.

The window watchdog timer clock is prescaled from the APB1 clock. The window watchdog timer is suitable for the situation that requires an accurate timing.

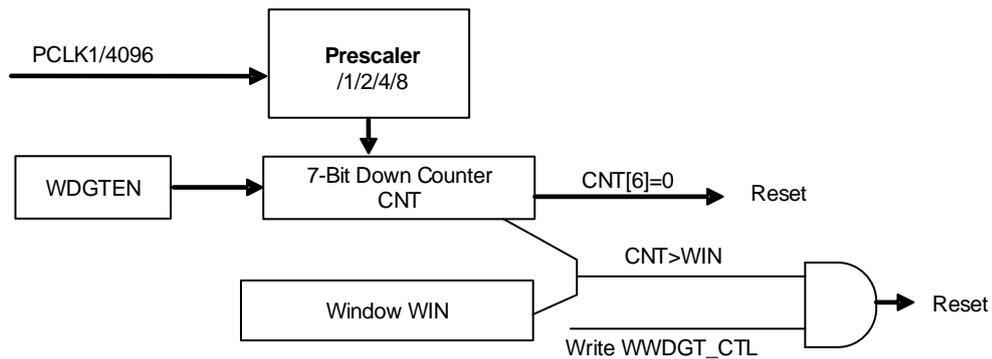
### 13.2.2. Characteristics

- Programmable free-running 7-bit downcounter.
- Generate reset in two conditions when WWDGT is enabled:
  - Reset when the counter reached 0x3F.
  - The counter is refreshed when the value of the counter is greater than the window register value.
- Early wakeup interrupt (EWI): if the watchdog is started and the interrupt is enabled, the interrupt occurs when the counter reaches 0x40 or refreshes before it reaches the window value.
- WWDGT debug mode, the WWDGT can stop or continue to work in debug mode.

### 13.2.3. Function overview

If the window watchdog timer is enable (set the WDG TEN bit in the WWDGT\_CTL), the watchdog timer cause a reset when the counter reached 0x3F (the CNT[6] bit becomes cleared), or when the counter is refreshed before the counter reached the window register value.

Figure 13-2. Window watchdog timer block diagram



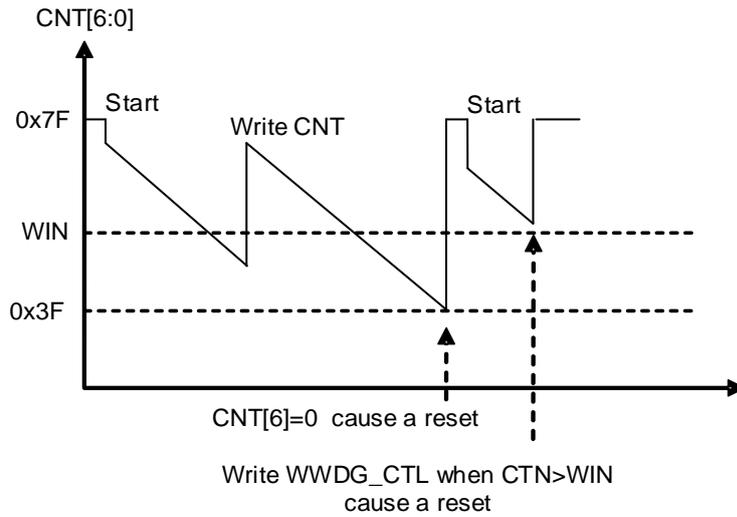
The watchdog is always disabled after power on reset. The software starts the watchdog by setting the WDG TEN bit in the WWDGT\_CTL register. Whenever window watchdog timer is enabled, the counter counts down all the time, the configured value of the counter should be greater than 0x3F, it implies that the CNT[6] bit should be set. The CNT[5:0] determine the maximum time interval of two reloading. The countdown speed depends on the APB1 clock and the prescaler (PSC[1:0] bits in the WWDGT\_CFG register).

The WIN[6:0] bits in the configuration register (WWDGT\_CFG) specifies the window value. The software can prevent the reset event by reloading the downcounter when counter value is less than the window value and greater than 0x3F, otherwise the watchdog causes a reset.

The early wakeup interrupt (EWI) is enabled by setting the EWIE bit in the WWDGT\_CFG register, and the interrupt is generated when the counter reaches 0x40 or the counter is refreshed before it reaches the window value. The software can do something such as communication or data logging in the interrupt service routine (ISR) in order to analyse the reason of software malfunctions or save the important data before resetting the device. Moreover the software can reload the counter in ISR to manage a software system check and so on. In this case, the WWDGT will never generate a WWDGT reset but can be used for other things.

The EWI interrupt is cleared by writing '0' to the EWIF bit in the WWDGT\_STAT register.

Figure 13-3. Window watchdog timer timing diagram



Calculate the WWDGT timeout by using the formula below.

$$t_{\text{WWDGT}} = t_{\text{PCLK1}} \times 4096 \times 2^{\text{PSC}} \times (\text{CNT}[5:0] + 1) \quad (\text{ms}) \quad (13-1)$$

where:

$t_{\text{WWDGT}}$ : WWDGT timeout

$t_{\text{PCLK1}}$ : APB1 clock period measured in ms

Refer to the [Table 13-2. Min-max timeout value at 36 MHz \(fPCLK1\)](#) for the minimum and maximum values of the  $t_{\text{WWDGT}}$ .

Table 13-2. Min-max timeout value at 36 MHz (fPCLK1)

Prescaler divider	PSC[1:0]	Min timeout value CNT[6:0] =0x40	Max timeout value CNT[6:0]=0x7F
1/1	00	113 $\mu$ s	7.28 ms
1/2	01	227 $\mu$ s	14.56 ms
1/4	10	455 $\mu$ s	29.12 ms
1/8	11	910 $\mu$ s	58.25 ms

If the WWDGT\_HOLD bit in DBG module is cleared, the WWDGT continues to work even the Cortex™-M3 core halted (Debug mode). While the WWDGT\_HOLD bit is set, the WWDGT stops in Debug mode.

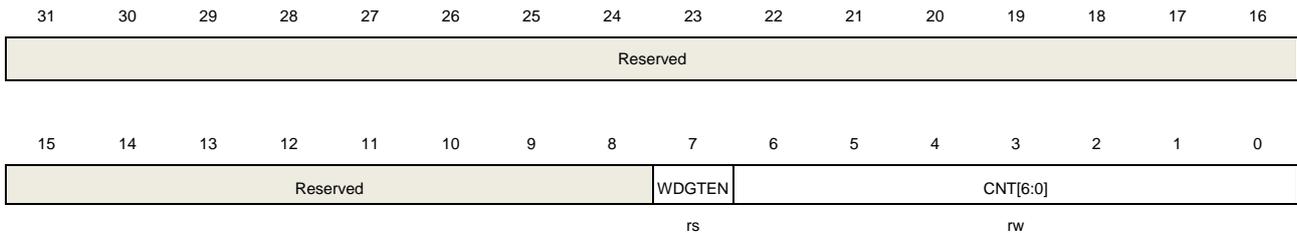
## 13.2.4. Register definition

### Control register (WWDGT\_CTL)

Address offset: 0x00

Reset value: 0x0000 007F

This register can be accessed by half-word (16-bit) or word (32-bit)



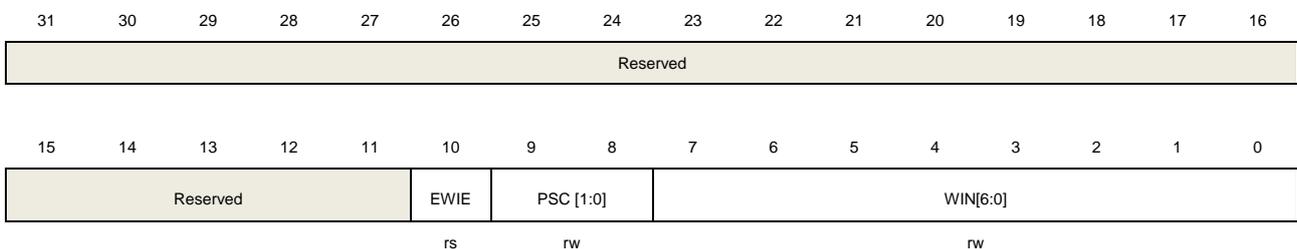
Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value.
7	WDGTEN	Start the window watchdog timer. Cleared by a hardware reset. Writing 0 has no effect. 0: Window watchdog timer disabled 1: Window watchdog timer enabled
6:0	CNT[6:0]	The value of the watchdog timer counter. A reset will occur when the value of this counter decreases from 0x40 to 0x3F. Writing this counter when the value of this counter is greater than the window value also cause a reset.

### Configuration register (WWDGT\_CFG)

Address offset: 0x04

Reset value: 0x0000 007F

This register can be accessed by half-word (16-bit) or word (32-bit)



Bits	Fields	Descriptions
31:10	Reserved	Must be kept at reset value.
9	EWIE	Early wakeup interrupt enable. An interrupt will occur when the counter reaches 0x40 or refreshes before it reaches the window value if the bit is set. It's could be cleared by a hardware reset or software clock reset (refer to 4.3.5. APB1 reset register ) . A write of 0 has

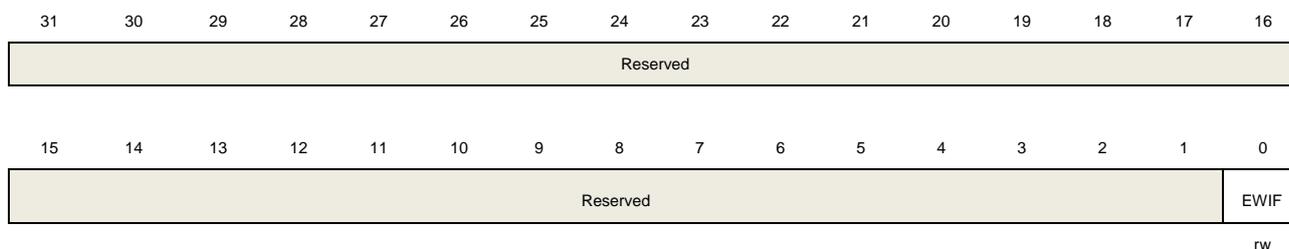
		no effect.
8:7	PSC[1:0]	Prescaler. The time base of the watchdog counter 00: PCLK1 / 4096 / 1 01: PCLK1 / 4096 / 2 10: PCLK1 / 4096 / 4 11: PCLK1 / 4096 / 8
6:0	WIN[6:0]	The Window value. A reset will occur if the watchdog counter (CNT bits in WWDGT_CTL) is written when the value of the watchdog counter is greater than the window value.

### Status register (WWDGT\_STAT)

Address offset: 0x08

Reset value: 0x0000 0000

This register can be accessed by half-word (16-bit) or word (32-bit)



Bits	Fields	Descriptions
31:1	Reserved	Must be kept at reset value.
0	EWIF	Early wakeup interrupt flag. When the counter has reached 0x40 or refreshes before it reaches the window value, this bit is set by hardware even the interrupt is not enabled (EWIE in WWDGT_CFG is cleared). This bit is cleared by writing 0. There is no effect when writing 1 to it.

## 14. Real-time clock(RTC)

### 14.1. Overview

The RTC provides a time which includes hour/minute/second/sub-second and a calendar including year/month/day/week day. The time and calendar are expressed in BCD code except sub-second. Sub-second is expressed in binary code. Hour adjustment for daylight saving time. Working in power saving mode and smart wakeup is software configurable. Support improving the calendar accuracy using extern accurate low frequency clock.

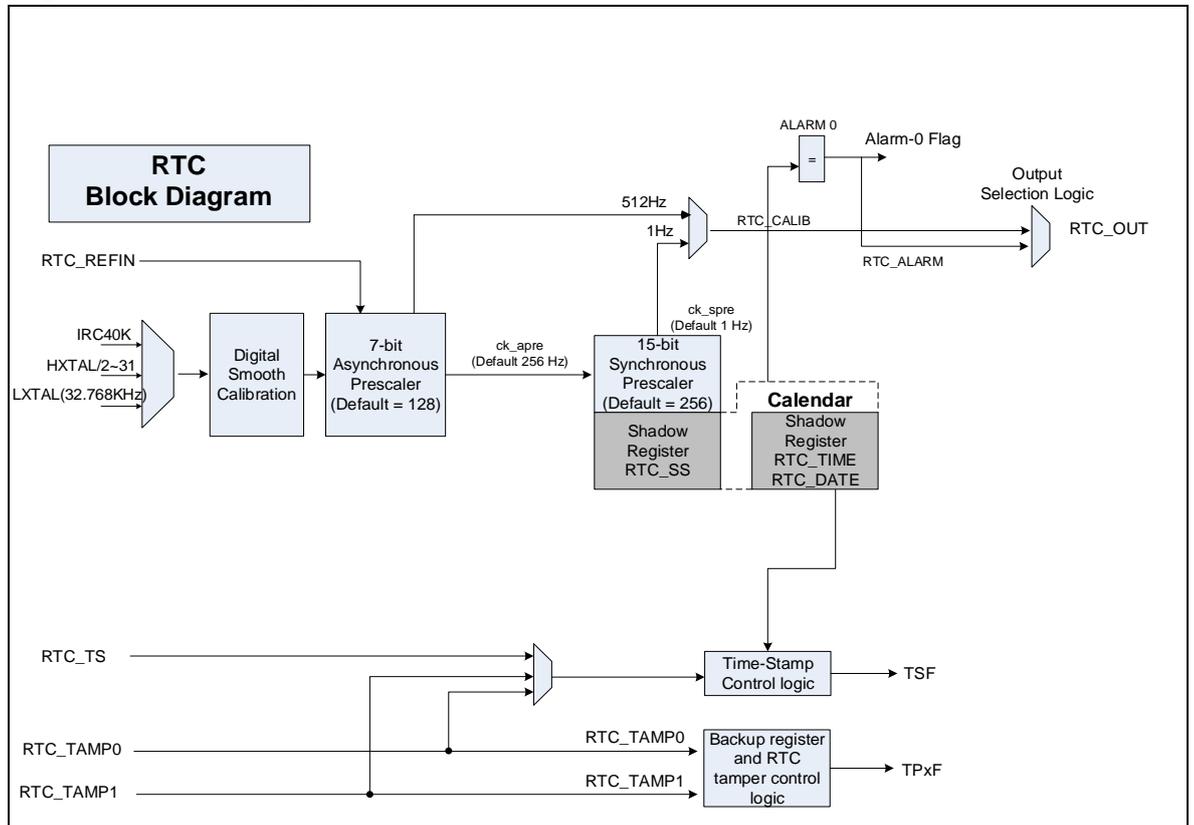
### 14.2. Characteristics

- Daylight saving compensation supported by software.
- External high-accurate low frequency (50Hz or 60Hz) clock used to achieve higher calendar accuracy performed by reference clock detection option function.
- Atomic clock adjustment (max adjustment accuracy is 0.95PPM) for calendar calibration performed by digital calibration function.
- Sub-second adjustment by shift function.
- Time-stamp function for saving event time.
- Two Tamper sources can be chosen and tamper type is configurable.
- Programmable calendar and one field maskable alarms.
- Maskable interrupt source:
  - Alarm 0
  - Time-stamp detection
  - Tamper detection
- Five 32-bit (20 bytes total) universal backup registers which can keep data under power saving mode. Backup register will be reset if tamper event detected.

## 14.3. Function overview

### 14.3.1. Block diagram

Figure 14-1. Block diagram of RTC



The RTC unit includes:

- Alarm event/interrupt
- Tamper event/interrupt
- 32-bit backup registers
- Optional RTC output function:
  - 512Hz (default prescale):RTC\_OUT
  - 1Hz(default prescale):RTC\_OUT
  - Alarm event(polarity is configurable):RTC\_OUT
- Optional RTC input function:
  - time stamp event detection: RTC\_TS
  - tamper 0 event detection: RTC\_TAMP0
  - tamper 1 event detection: RTC\_TAMP1

- reference clock input: RTC\_REFIN(50 or 60 Hz)

## 14.3.2. Clock source and prescalers

RTC unit has three independent clock sources: LXTAL, IRC40K and HXTAL with divided by 32.

In the RTC unit, there are two prescalers used for implementing the calendar and other functions. One prescaler is a 7-bit asynchronous prescaler and the other is a 15-bit synchronous prescaler. Asynchronous prescaler is mainly used for reducing power consumption. The asynchronous prescaler is recommended to set as high as possible if both prescalers are used.

The frequency formula of two prescalers is shown as below:

$$f_{ck\_apre} = \frac{f_{rtclk}}{FACTOR\_A + 1} \quad (14-1)$$

$$f_{ck\_spre} = \frac{f_{ck\_apre}}{FACTOR\_S + 1} = \frac{f_{rtclk}}{(FACTOR\_A + 1) * (FACTOR\_S + 1)} \quad (14-2)$$

The ck\_apre clock is used to driven the RTC\_SS down counter which stands for the time left to next second in binary format and when it reaches 0 it will automatically reload FACTOR\_S value. The ck\_spre clock is used to driven the calendar registers. Each clock will make second plus one.

## 14.3.3. Real-time clock and calendar

BPSHAD control bit decides the location when APB bus accesses the RTC calendar register RTC\_DATE, RTC\_TIME and RTC\_SS. By default, the BPSHAD is cleared, and APB bus accesses the shadow calendar registers. Shadow calendar registers is updated to real calendar register every two RTC clock and at the same time RSYNF bit is set once. This update mechanism is not performed in Deep-Sleep mode and Standby mode. When exiting these modes, software must clear RSYNF bit and wait it is asserted (the max wait time is 2 RTC clock) if software wants to read calendar register under BPSHAD=0 situation.

**Note:** When reading calendar registers (RTC\_SS, RTC\_TIME, RTC\_DATE) under BPSHAD=0, the frequency of the APB clock (f<sub>APB</sub>) must be at least 7 times the frequency of the RTC clock (f<sub>RTCLK</sub>).

System reset will reset the shadow calendar registers.

## 14.3.4. Configurable and field maskable alarm

RTC alarm function is divided into some fields and each has a maskable bit.

RTC alarm function can be enabled/disabled by ALRM0EN bit in RTC\_CTL. If all the alarm fields value match the corresponding calendar value when ALRM0EN=1, the ALRM0F flag will be set.

**Note:** FACTOR\_S in the RTC\_PSC register must be larger than 3 if MSKS bit reset in RTC\_ALRM0TD.

If a field is masked, the field is considered as matched in logic. If all the fields have been masked, the Alarm Flag will assert 3 RTC clock later after ALRM0EN is set.

### 14.3.5. RTC initialization and configuration

#### RTC register write protection

BKPWEN bit in the PMU\_CTL register is cleared in default, so writing to RTC registers needs setting BKPWEN bit ahead of time.

After power-on reset, most of RTC registers are write protected. Unlocking this protection is the first step before writing to them.

Following below steps will unlock the write protection:

1. Write '0xCA' into the RTC\_WPK register
2. Write '0x53' into the RTC\_WPK register

Writing a wrong value to RTC\_WPK will make write protection valid again. The state of write protection is not affected by system reset. Following registers are write protected but others are not:

RTC\_TIME, RTC\_DATE, RTC\_CTL, RTC\_STAT, RTC\_PSC, RTC\_ALRM0TD,  
RTC\_SHIFTCTL, RTC\_HRFC, RTC\_ALRM0SS

#### Calendar initialization and configuration

The prescaler and calendar value can be programmed by following steps:

1. Enter initialization mode (by setting INITM=1) and polling INITF bit until INITF=1.
2. Program both the asynchronous and synchronous prescaler factors in RTC\_PSC register.
3. Write the initial calendar values into the shadow calendar registers (RTC\_TIME and RTC\_DATE), and use the CS bit in the RTC\_CTL register to configure the time format (12 or 24 hours).
4. Exit the initialization mode (by setting INITM=0).

About 4 RTC clock cycles later, real calendar registers will load from shadow registers and calendar counter restarts.

**Note:** Reading calendar register (BPSHAD=0) after initialization, software should confirm the RSYNF bit is already set to 1.

YCM flag indicates whether the calendar has been initialized by checking the year field of calendar.

### Daylight saving Time

RTC unit supports daylight saving time adjust function through S1H, A1H and DSM bit.

S1H and A1H can subtract or add 1 hour to the calendar when the calendar is running. S1H and A1H operation can be tautologically set and DSM bit can be used to recording this adjustment operation. After setting the S1H/A1H, subtracting/adding 1 hour will perform when next second comes.

### Alarm function operation process

To avoid unexpected alarm assertion and metastable state, alarm function has an operation flow:

1. Disable Alarm (by resetting ALRM0EN in RTC\_CTL)
2. Set the Alarm registers needed(RTC\_ALARM0TD/RTC\_ALARM0SS)
3. Enable Alarm function (by setting ALRM0EN in the RTC\_CTL)

### 14.3.6. Calendar reading

#### Reading calendar registers under BPSHAD=0

When BPSHAD=0, calendar value is read from shadow registers. For the existence of synchronization mechanism, a basic request has to meet: the APB1 bus clock frequency must be equal to or greater than 7 times the RTC clock frequency. APB1 bus clock frequency lower than RTC clock frequency is not allowed in any case.

When APB1 bus clock frequency is not equal to or greater than 7 times the RTC clock frequency, the calendar reading flow should be obeyed:

1. reading calendar time register and date register twice
2. if the two values are equal, the value can be seen as the correct value
3. if the two values are not equal, a third reading should performed
4. the third value can be seen as the correct value

RSYNF is asserted once every 2 RTC clock and at this time point, the shadow registers will be updated to current time and date.

To ensure consistency of the 3 values (RTC\_SS, RTC\_TIME, and RTC\_DATE), below consistency mechanism is used in hardware:

1. reading RTC\_SS will lock the updating of RTC\_TIME and RTC\_DATE
2. reading RTC\_TIME will lock the updating of RTC\_DATE
3. reading RTC\_DATE will unlock updating of RTC\_TIME and RTC\_DATE

If the software wants to read calendar in a short time interval(smaller than 2 RTCCLK periods),

RSYNF must be cleared by software after the first calendar read, and then the software must wait until RSYNF is set again before next reading.

In below situations, software should wait RSYNF bit asserted before reading calendar registers(RTC\_SS, RTC\_TIME, RTC\_DATE):

1. after a system reset
2. after an initialization
3. after shift function

Especially that software must clear RSYNF bit and wait it asserted before reading calendar register after wakeup from power saving mode.

### **Reading calendar registers under BPSHAD=1**

When BPSHAD=1, RSYNF is cleared and maintains as 0 by hardware so reading calendar registers does not care about RSYNF bit. Current calendar value is read from real-time calendar counter directly. The benefit of this configuration is that software can get the real current time without any delay after wakeup from power saving mode (Deep-sleep /Standby Mode).

Because of no RSYNF bit periodic assertion, the results of the different calendar registers (RTC\_SS/RTC\_TIME/RTC\_DATE) might not be coherent with each other when clock ck\_apre edge occurs between two reading calendar registers.

In addition, if current calendar register is changing and at the same time the APB bus reading calendar register is also performing, the value of the calendar register read out might be not correct.

To ensure the correctness and consistency of the calendar value, software must perform reading operation as this: read all calendar registers continuously, if the last two values are the same, the data is coherent and correct.

### **14.3.7. Resetting the RTC**

There are two reset sources used in RTC unit: system reset and backup domain reset.

System reset will affect calendar shadow registers and some bits of the RTC\_STAT. When system reset is valid, the bits or registers mentioned before are reset to the default value.

Backup domain reset will affect the following registers and system reset will not affect them:

- RTC current real-time calendar registers
- RTC control register (RTC\_CTL)
- RTC prescaler register (RTC\_PSC)
- RTC high resolution frequency compensation register (RTC\_HRFC)

- RTC shift register (RTC\_SHIFTCTL)
- RTC timestamp registers (RTC\_SSTS/RTC\_TTS/RTC\_DTS)
- RTC tamper and alternate function configuration register (RTC\_TAMP)
- RTC backup registers (RTC\_BKPx)
- RTC Alarm registers (RTC\_ALRM0SS/RTC\_ALRM0TD)

The RTC unit will go on running when system reset occurs or enter power saving mode, but if backup domain reset occurs, RTC will stop counting and all registers will reset.

### 14.3.8. RTC shift function

When there is a remote clock with higher degree of precision and RTC 1Hz clock(ck\_spre) has an offset (in a fraction of a second) with the remote clock, RTC unit provides a function named shift function to remove this offset and thus make second precision higher.

RTC\_SS register indicates the fraction of a second in binary format and is down counting when RTC is running. Therefore by adding the SFS[14:0] value to the synchronous prescaler counter SSC[15:0] or by adding the SFS[14:0] value to the synchronous prescaler counter SSC[15:0] and at the same time set A1S bit can delay or advance the time when next second arrives.

The maximal RTC\_SS value depends on the FACTOR\_S value in RTC\_PSC. The higher is FACTOR\_S, the higher is adjust precision.

Because of the 1Hz clock(ck\_spre) is generated by FACTOR\_A and FACTOR\_S, the higher FACTOR\_S means the lower FACTOR\_A, then more power consuming.

**Note:** Before using shift function, the software must check the MSB of SSC in RTC\_SS(SSC[15]) and confirm it is 0.

After writing RTC\_SHIFTCTL register, the SOPF bit in RTC\_STAT will be set at once. When shift operation is complete, SOPF bit is cleared by hardware. System reset does not affect SOPF bit

Shift operation only works correctly when REFEN=0.

Software must not write to RTC\_SHIFTCTL if REFEN=1.

### 14.3.9. RTC reference clock detection

RTC reference clock detection is another way to increase the precision of RTC second. To enable this function, you should have an external clock source (50Hz or 60 Hz) which is more precise than LXTAL clock source.

After enabling this function (REFEN=1), each 1Hz clock (ck\_spre) edge is compared to the nearest RTC\_REFIN clock edge. In most cases, the two clock edges are aligned every time.

But when two clock edges are misaligned for the reason of LXTAL poor precision, the RTC reference clock detection function will shift the 1Hz clock edge a little to make next 1Hz clock edge aligned to reference clock edge.

When REFEN=1, a time window is applied at every second update time. Different detection state will use different window period.

7 ck\_apre window is used when detecting the first reference clock edge and 3 ck\_apre window is used for the edge aligned operation.

Whatever window used, the asynchronous prescaler counter will be forced to reload when the reference clock is detected in the window. When the two clock (ck\_spre and reference clock) edges are aligned, this reload operation has no effect for 1Hz clock. But when the two clock edge are not aligned, this reload operation will shift ck\_spre clock edge a bit to make the ck\_spre(1Hz) clock edge aligned to the reference clock edge.

When reference detection function is running while the external reference clock is removed (no reference clock edge found in 3 ck\_apre window), the calendar updating still can be performed by LXTAL clock only. If the reference clock is recovered later, detection function will use 7 ck\_apre window to identify the reference clock and use 3 ck\_apre window to adjust the 1Hz clock (ck\_spre) edge.

**Note:** Software must configure the FACTOR\_A=0x7F and FACTOR\_S=0xFF before enabling reference detection function (REFEN=1).

Reference detection function does not work in Standby Mode.

#### 14.3.10. RTC smooth digital calibration

RTC smooth calibration function is a way to calibrate the RTC frequency based on RTC clock in a configurable period time.

This calibration is equally executed in a period time and the cycle number of the RTC clock in the period time will be added or subtracted. The resolution of the calibration is about 0.954PPM with the range from -487.1PPM to +488.5PPM.

The calibration period time can be configured to the  $2^{20}/2^{19}/2^{18}$  RTC clock cycles which stands for 32/16/8 seconds if RTC input frequency is 32.768 KHz.

The High resolution frequency compensation register (RTC\_HRFC) specifies the number of RTCCLK clock cycles to be calibrated during the period time:

So using CMSK can mask clock cycles from 0 to 511 and thus the RTC frequency can be reduced by up to 487.1PPM.

To increase the RTC frequency the FREQI bit can be set. If FREQI bit is set, there will be 512 additional cycles to be added during period time which means every  $2^{11}/2^{10}/2^{09}$ (32/16/8 seconds) RTC clock insert one cycle.

So using FREQI can increase the RTC frequency by 488.5PPM.

The combined using of CMSK and FREQI can adjust the RTC cycles from -511 to +512 cycles in the period time which means the calibration range is -487.1PPM to +488.5PPM with a resolution of about 0.954PPM.

When calibration function is running, the output frequency of calibration is calculated by the following formula:

$$f_{cal} = f_{rtclk} \times \left( 1 + \frac{FREQI \times 512 - CMSK}{2^N + CMSK - FREQI \times 512} \right) \quad (14-3)$$

**Note:** N=20/19/18 for 32/16/8 seconds window period

## Calibration when FACTOR\_A < 3

When asynchronous prescaler value (FACTOR\_A) is set to less than 3, software should not set FREQI bit to 1 when using calibration function. FREQI setting will be ignored when FACTOR\_A < 3.

When the FACTOR\_A is less than 3, the FACTOR\_S value should be set to a value less than the nominal value. Assuming that RTC clock frequency is nominal 32.768 KHz, the corresponding FACTOR\_S should be set as following rule:

FACTOR\_A = 2: 2 less than nominal FACTOR\_S (8189 with 32.768 KHz)

FACTOR\_A = 1: 4 less than nominal FACTOR\_S (16379 with 32.768 KHz)

FACTOR\_A = 0: 8 less than nominal FACTOR\_S (32759 with 32.768 KHz)

When the FACTOR\_A is less than 3, CMSK is 0x100, the formula of calibration frequency is as follows:

$$f_{cal} = f_{rtclk} \times \left( 1 + \frac{256 - CMSK}{2^N + CMSK - 256} \right) \quad (14-4)$$

**Note:** N=20/19/18 for 32/16/8 seconds window period

## Verifying the RTC calibration

Calibration 1Hz output is provided to assist software to measure and verify the RTC precision.

Up to 2 RTC clock cycles measurement error may occur when measuring the RTC frequency over a limited measurement period. To eliminate this measurement error the measurement period should be the same as the calibration period.

- When the calibration period is 32 seconds (this is default configuration)

Using exactly 32s period to measure the accuracy of the calibration 1Hz output can guarantee the measure is within 0.477PPM (0.5 RTCCCLK cycles over 32s)

- When the calibration period is 16 seconds (by setting CWND16 bit)

In this configuration, CMSK[0] is fixed to 0 by hardware. Using exactly 16s period to measure the accuracy of the calibration 1Hz output can guarantee the measure is within 0.954PPM (0.5 RTCCCLK cycles over 16s)

- When the calibration period is 8 seconds (by setting CWND8 bit)

In this configuration, CMSK[1:0] is fixed to 0 by hardware. Using exactly 8s period to measure the accuracy of the calibration 1Hz output can guarantee the measure is within 1.907PPM (0.5 RTCCLK cycles over 8s)

### Re-calibration on-the-fly

When the INITF bit is 0, software can update the value of RTC\_HRFC using following steps:

- 1) Wait the SCPF=0
- 2) Write the new value into RTC\_HRFC register
- 3) After 3 ck\_apre clocks, the new calibration settings take effect

### 14.3.11. Time-stamp function

Time-stamp function is performed on RTC\_TS pin and is enabled by control bit TSEN.

When a time-stamp event occurs on RTC\_TS pin, the calendar value will be saved in time-stamp registers (RTC\_DTS/RTC\_TTS/RTC\_SSTS) and the time-stamp flag (TSF) is set to 1 by hardware. Time-stamp event can generate an interrupt if time-stamp interrupt enable (TSIE) is set.

Time-stamp registers only record the calendar at the first time time-stamp event occurs which means that time-stamp registers will not change when TSF=1.

To extend the time-stamp event source, one optional feature is provided: tamper function can also be considered as time-stamp function if TPTS is set.

**Note:** When the time-stamp event occurs, TSF is set 2 ck\_apre cycles delay because of synchronization mechanism.

### 14.3.12. Tamper detection

The RTC\_TAMPx pin input can be used for tamper event detection under edge detection mode or level detection mode with configurable filtering setting.

#### RTC backup registers(RTC\_BKPx)

The RTC backup registers are located in the V<sub>DD</sub> backup domain that remains powered-on by V<sub>BAT</sub> even if V<sub>DD</sub> power is switched off. The wake up action from Standby Mode or system reset are not affect these registers.

These registers are only reset by detected tamper event and backup domain reset.

#### Tamper detection function initialization

RTC tamper detection function can be independently enabled on tamper input pin by setting corresponding TPxEN bit. Tamper detection configuration is set before enable TPxEN bit. When the tamper event is detected, the corresponding flag (TPxF) will assert. Tamper event

can generate an interrupt if tamper interrupt enable (TPIE) is set. Any tamper event will reset all backup registers (RTC\_BKPx).

### Timestamp on tamper event

The TPTS bit can control whether the tamper detection function is used as time-stamp function. If the bit is set to 1, the TSF bit will be set when the tamper event detected as if “enable” the time-stamp function. Whatever the TPTS bit is, the TPxF will assert when tamper event detected.

### Edge detection mode on tamper input detection

When FLT bit is set to 0x0, the tamper detection is set to edge detection mode and TPxEG bit determines the rising edge or falling edge is the detecting edge. When tamper detection is under edge detection mode, the internal pull-up resistors on the tamper detection input pin are deactivated.

Because of detecting the tamper event will reset the backup registers (RTC\_BKPx), writing to the backup register should ensure that the tamper event reset and the writing operation will not occur at the same time, a recommend way to avoid this situation is disable the tamper detection before writing to the backup register and re-enable tamper detection after finish writing.

**Note:** Tamper detection is still running when V<sub>DD</sub> power is switched off if tamper is enabled.

### Level detection mode with configurable filtering on tamper input detection

When FLT bit is not reset to 0x0, the tamper detection is set to level detection mode and FLT bit determines the consecutive number of samples (2, 4 or 8) needed for valid level. When DISPU is set to 0x0(this is default), the internal pull-up resistance will pre-charge the tamper input pin before each sampling and thus larger capacitance is allowed to connect to the tamper input pin. The pre-charge duration is configured through PRCH bit. Higher capacitance needs long pre-charge time.

The time interval between each sampling is also configurable. Through adjusting the sampling frequency (FREQ), software can balance between the power consuming and tamper detection latency.

## 14.3.13. Calibration clock output

Calibration clock can be output on the RTC\_OUT if COEN bit is set to 1.

When the COS bit is set to 0(this is default) and asynchronous prescaler is set to 0x7F(FACTOR\_A), the frequency of RTC\_CALIB is  $f_{rtcclk}/64$ . When the RTCCLK is 32.768KHz, RTC\_CALIB output is corresponding to 512Hz.It's recommend to using rising edge of RTC\_CALIB output because there may be a light jitter on falling edge.

When the COS bit is set to 1, the RTC\_CALIB frequency is :

$$f_{rtc\_calib} = \frac{f_{rtcclk}}{(FACTOR\_A+1) \times (FACTOR\_S+1)} \quad (14-5)$$

When the RTCCLK is 32.768 KHz, RTC\_CALIB output is corresponding to 1Hz if prescaler are default values.

### 14.3.14. Alarm output

When OS control bits are not reset, RTC\_ALARM alternate function output is enabled. This function will directly output the content of alarm flag in RTC\_STAT.

The OPOL bit in RTC\_CTL can configure the polarity of the alarm output which means that the RTC\_ALARM output is the opposite of the corresponding flag bit or not.

### 14.3.15. RTC power saving mode management

**Table 14-1. RTC power saving mode management**

Mode	Active in Mode	Exit Mode
Sleep	Yes	RTC Interrupts
Deep-Sleep	Yes: if clock source is LXTAL or IRC40K	RTC Alarm/ Tamper Event/ Timestamp Event
Standby	Yes: if clock source is LXTAL or IRC40K	RTC Alarm/ Tamper Event/ Timestamp Event

### 14.3.16. RTC interrupts

All RTC interrupts are connected to the EXTI controller.

Below steps should be followed if you want to use the RTC alarm/tamper/timestamp:

- 1) Configure and enable the corresponding interrupt line to RTC alarm/tamper/timestamp event of EXTI and set the rising edge for triggering
- 2) Configure and enable the RTC alarm/tamper/timestamp global interrupt
- 3) Configure and enable the RTC alarm/tamper/timestamp function

**Table 14-2. RTC interrupts control**

Interrupt	Event flag	Control Bit	Exit Sleep	Exit Deep-sleep	Exit Standby
Alarm 0	ALRM0F	ALRM0IE	Y	Y(*)	Y(*)
Timestamp	TSF	TSIE	Y	Y(*)	Y(*)
Tamper 0	TPOF	TPIE	Y	Y(*)	Y(*)

---

Interrupt	Event flag	Control Bit	Exit Sleep	Exit Deep-sleep	Exit Standby
Tamper 1	TP1F	TPIE	Y	Y(*)	Y(*)

\* Only active when RTC clock source is LXTAL or IRC40K.

## 14.4. Register definition

### 14.4.1. Time register (RTC\_TIME)

Address offset: 0x00

System reset value: 0x0000 0000 when BPSHAD = 0.

Not affected when BPSHAD = 1.

This register is write protected and can only be written in initialization state

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									PM	HRT[1:0]		HRU[3:0]			
									rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	MNT[2:0]		MNU[3:0]			Reserved	SCT[2:0]		SCU[3:0]						
		rw	rw					rw	rw						

Bits	Fields	Descriptions
31:23	Reserved	Must be kept at reset value
22	PM	AM/PM mark 0: AM or 24-hour format 1: PM
21:20	HRT[1:0]	Hour tens in BCD code
19:16	HRU[3:0]	Hour units in BCD code
15	Reserved	Must be kept at reset value
14:12	MNT[2:0]	Minute tens in BCD code
11:8	MNU[3:0]	Minute units in BCD code
7	Reserved	Must be kept at reset value
6:4	SCT[2:0]	Second tens in BCD code
3:0	SCU[3:0]	Second units in BCD code

### 14.4.2. Date register (RTC\_DATE)

Address offset: 0x04

System reset value: 0x0000 2101 when BPSHAD = 0.

Not affected when BPSHAD = 1.

This register is write protected and can only be written in initialization state

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									YRT[3:0]			YRU[3:0]			
									rw			rw			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DOW[2:0]		MONT	MONU[2:0]			Reserved	DAYT		DAYU						
rw		rw	rw					rw	rw						

Bits	Fields	Descriptions
------	--------	--------------

31:24	Reserved	Must be kept at reset value
23:20	YRT[3:0]	Year tens in BCD code
19:16	YRU[3:0]	Year units in BCD code
15:13	DOW[2:0]	Days of the week 0x0: Reserved 0x1: Monday ... 0x7: Sunday
12	MONT	Month tens in BCD code
11:8	MONU[2:0]	Month units in BCD code
7:6	Reserved	Must be kept at reset value
5:4	DAYT[1:0]	Day tens in BCD code
3:0	DAYU[3:0]	Day units in BCD code

### 14.4.3. Control register (RTC\_CTL)

Address offset: 0x08

System reset: not affected

Backup domain reset value: 0x0000 0000

This register is writing protected

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								COEN	OS[1:0]		OPOL	COS	DSM	S1H	A1H
								rw	rw	rw	rw	rw	w	w	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSIE	Reserved	ALRMOIE	TSEN	Reserved	ALRMOEN	Reserved	CS	BPSHAD	REFEN	TSEG	Reserved				
rw		rw	rw		rw		rw	rw	rw	rw					

Bits	Fields	Descriptions
31:24	Reserved	Must be kept at reset value
23	COEN	Calibration output enable 0: Disable calibration output 1: Enable calibration output
22:21	OS[1:0]	Output selection This bit is used for selecting flag source to output 0x0: Disable output RTC_ALARM 0x1: Enable alarm0 flag output 0x2: Reserved 0x3: Reserved
20	OPOL	Output polarity This bit is used to invert output RTC_ALARM 0: Disable invert output RTC_ALARM 1: Enable invert output RTC_ALARM
19	COS	Calibration output selection Valid only when COEN=1 and prescalers are at default values 0: Calibration output is 512 Hz 1: Calibration output is 1Hz
18	DSM	Daylight saving mark This bit is flexible used by software. Often can be used to recording the daylight saving

		hour adjustment.
17	S1H	Subtract 1 hour(winter time change) One hour will be subtracted from current time if it is not 0 0: No effect 1: 1 hour will be subtracted at next second change time.
16	A1H	Add 1 hour(summer time change) One hour will be added from current time 0: No effect 1: 1 hour will be added at next second change time
15	TSIE	Time-stamp interrupt enable 0: Disable time-stamp interrupt 1: Enable time-stamp interrupt
14:13	Reserved	Must be kept at reset value
12	ALRM0IE	RTC alarm-0 interrupt enable 0: Disable alarm interrupt 1: Enable alarm interrupt
11	TSEN	Time-stamp function enable 0: Disable time-stamp function 1: Enable time-stamp function
10:9	Reserved	Must be kept at reset value
8	ALRM0EN	Alarm-0 function enable 0: Disable alarm function 1: Enable alarm function
7	Reserved	Must be kept at reset value
6	CS	Clock System 0: 24-hour format 1: 12-hour format <b>Note:</b> Can only be written in initialization state
5	BPSHAD	Shadow registers bypass control 0: Reading calendar from shadow registers 1: Reading calendar from current real-time calendar <b>Note:</b> If frequency of APB1 clock is less than seven times the frequency of RTCCLK, this bit must set to 1.
4	REFEN	Reference clock detection function enable 0: Disable reference clock detection function 1: Enable reference clock detection function <b>Note:</b> Can only be written in initialization state and FACTOR_S must be 0x00FF
3	TSEG	Valid event edge of time-stamp 0: rising edge is valid event edge for time-stamp event 1: falling edge is valid event edge for time-stamp event
2:0	Reserved	Must be kept at reset value

#### 14.4.4. Status register (RTC\_STAT)

Address offset: 0x0C

System reset: Only INITM, INITF and RSYNF bits are set to 0. Others are not affected

Backup domain reset value: 0x0000 0007

This register is writing protected except RTC\_STAT[14:8].

This register has to be accessed by word(32-bit)

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



0:Alarm registers programming is not allowed

1:Alarm registers programming is allowed

#### 14.4.5. Prescaler register (RTC\_PSC)

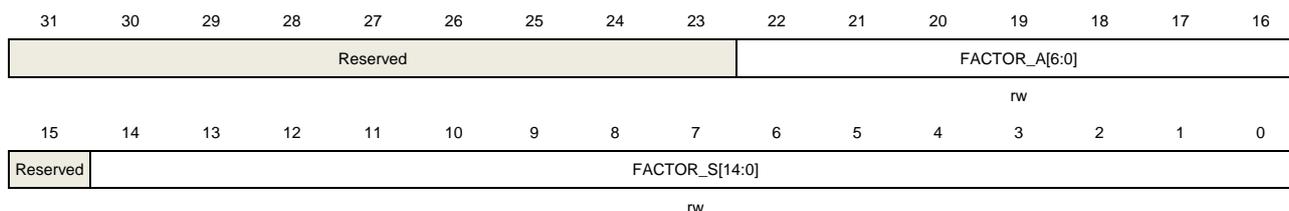
Address offset: 0x10

System reset: not effected

Backup domain reset value: 0x007F 00FF

This register is write protected and can only be written in initialization state

This register has to be accessed by word(32-bit)



Bits	Fields	Descriptions
31:23	Reserved	Must be kept at reset value
22:16	FACTOR_A[6:0]	Asynchronous prescaler factor ck_apre frequency = RTCCLK frequency/(FACTOR_A+1)
15	Reserved	Must be kept at reset value
14:0	FACTOR_S[14:0]	Synchronous prescaler factor ck_spre frequency = ck_apre frequency/(FACTOR_S+1)

#### 14.4.6. Alarm 0 time and date register (RTC\_ALRM0TD)

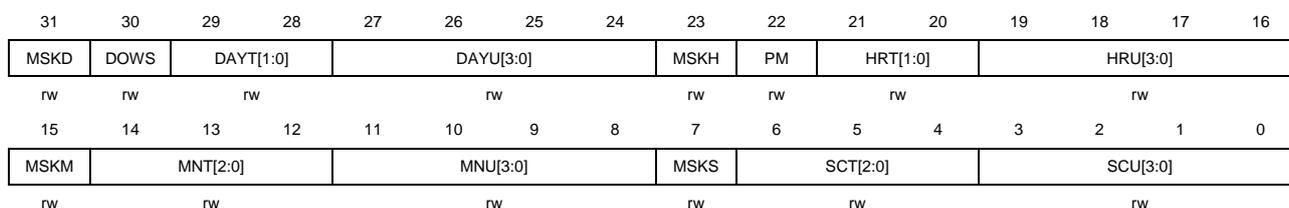
Address offset: 0x1C

System reset: not effect

Backup domain reset value: 0x0000 0000

This register is write protected and can only be written in initialization state

This register has to be accessed by word(32-bit)



Bits	Fields	Descriptions
31	MSKD	Alarm date mask bit 0:Not mask date/day field 1:Mask date/day field
30	DOWS	Day of the week selected 0:DAYU[3:0] indicates the date units 1: DAYU[3:0] indicates the week day and DAYT[1:0] has no means.
29:28	DAYT[1:0]	Date tens in BCD code
27:24	DAYU[3:0]	Date units or week day in BCD code
23	MSKH	Alarm hour mask bit

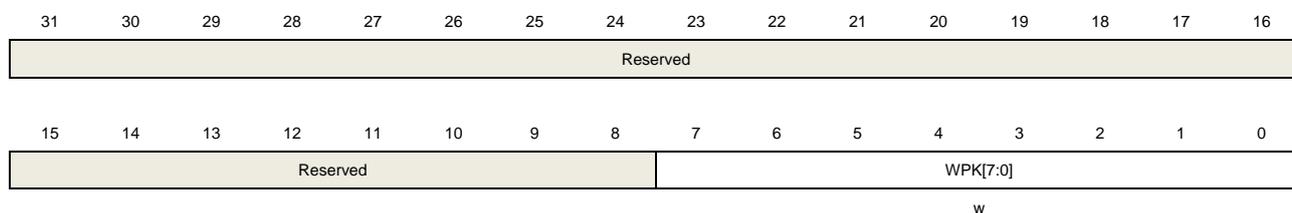
		0:Not mask hour field 1:Mask hour field
22	PM	AM/PM flag 0:AM or 24-hour format 1:PM
21:20	HRT[1:0]	Hour tens in BCD code
19:16	HRU[3:0]	Hour units in BCD code
15	MSKM	Alarm minutes mask bit 0:Not mask minutes field 1:Mask minutes field
14:12	MNT[2:0]	Minutes tens in BCD code
11:8	MNU[3:0]	Minutes units in BCD code
7	MSKS	Alarm second mask bit 0:Not mask second field 1:Mask second field
6:4	SCT[2:0]	Second tens in BCD code
3:0	SCU[3:0]	Second units in BCD code

#### 14.4.7. Write protection key register (RTC\_WPK)

Address offset: 0x24

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)



Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value
7:0	WPK[7:0]	Key for write protection

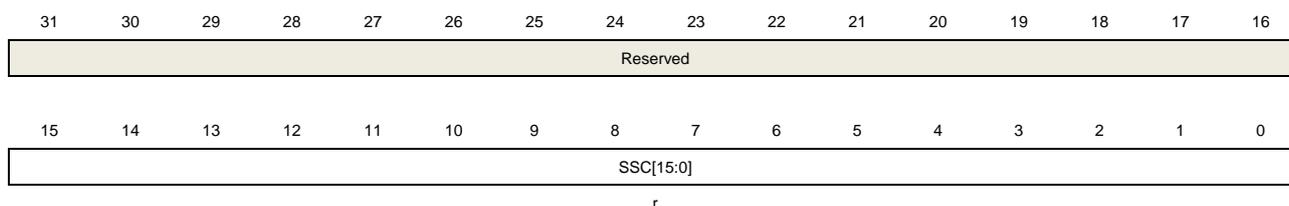
#### 14.4.8. Sub second register (RTC\_SS)

Address offset: 0x28

System reset value: 0x0000 0000 when BPSHAD = 0.

Not affected when BPSHAD = 1.

This register has to be accessed by word (32-bit)



Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	SSC[15:0]	Sub second value This value is the counter value of synchronous prescaler. Second fraction value is calculated by the below formula: Second fraction = ( FACTOR_S - SSC ) / ( FACTOR_S + 1 )

## 14.4.9. Shift function control register (RTC\_SHIFTCTL)

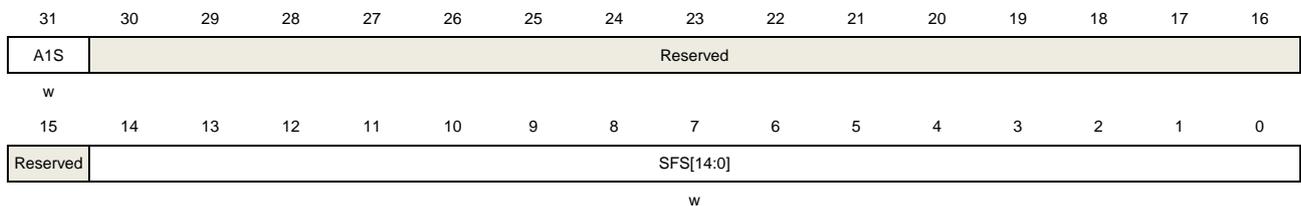
Address offset: 0x2C

System reset: not effect

Backup Reset value: 0x0000 0000

This register is writing protected and can only be wrote when SOPF=0

This register has to be accessed by word(32-bit)



Bits	Fields	Descriptions
31	A1S	One second add 0:Not add 1 second 1:Add 1 second to the clock/calendar. This bit is jointly used with SFS field to add a fraction of a second to the clock.
30:15	Reserved	Must be kept at reset value
14:0	SFS[14:0]	Subtract a fraction of a second The value of this bit will add to the counter of synchronous prescaler. When only using SFS, the clock will delay because the synchronous prescaler is a down counter: Delay (seconds) = SFS / ( FACTOR_S + 1 ) When jointly using A1S and SFS, the clock will advance: Advance (seconds) = ( 1 - ( SFS / ( FACTOR_S + 1 ) ) )

**Note:** Writing to this register will cause RSYNF bit to be cleared.

## 14.4.10. Time of time stamp register (RTC\_TTS)

Address offset: 0x30

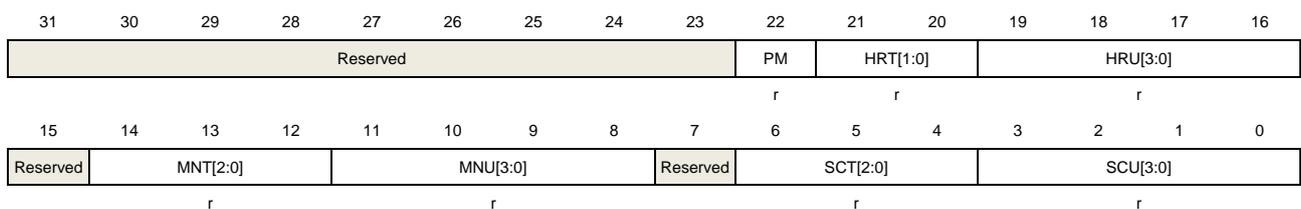
Backup domain reset value: 0x0000 0000

System reset: no effect

This register will record the calendar time when TSF is set to 1.

Reset TSF bit will also clear this register.

This register has to be accessed by word(32-bit)



Bits	Fields	Descriptions
------	--------	--------------

31:23	Reserved	Must be kept at reset value
22	PM	AM/PM mark 0:AM or 24-hour format 1:PM
21:20	HRT[1:0]	Hour tens in BCD code
19:16	HRU[3:0]	Hour units in BCD code
15	Reserved	Must be kept at reset value
14:12	MNT[2:0]	Minute tens in BCD code
11:8	MNU[3:0]	Minute units in BCD code
7	Reserved	Must be kept at reset value
6:4	SCT[2:0]	Second tens in BCD code
3:0	SCU[3:0]	Second units in BCD code

#### 14.4.11. Date of time stamp register (RTC\_DTS)

Address offset: 0x34

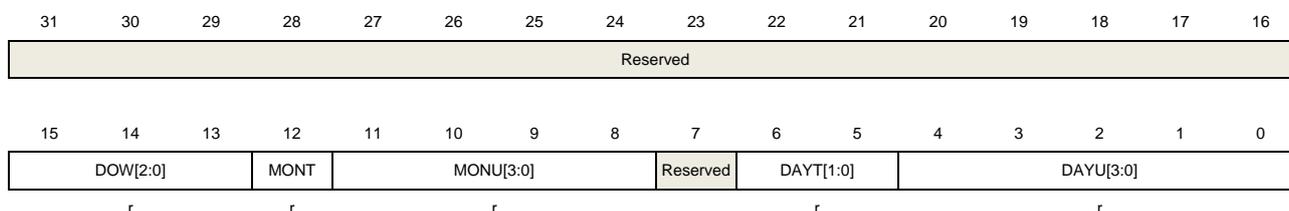
Backup domain reset value: 0x0000 0000

System reset: no effect

This register will record the calendar date when TSF is set to 1.

Reset TSF bit will also clear this register.

This register has to be accessed by word(32-bit)



Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:13	DOW[2:0]	Days of the week
12	MONT	Month tens in BCD code
11:8	MONU[3:0]	Month units in BCD code
7	Reserved	Must be kept at reset value
6:5	DAYT[1:0]	Day tens in BCD code
4:0	DAYU[3:0]	Day units in BCD code

#### 14.4.12. Sub second of time stamp register (RTC\_SSTS)

Address offset: 0x38

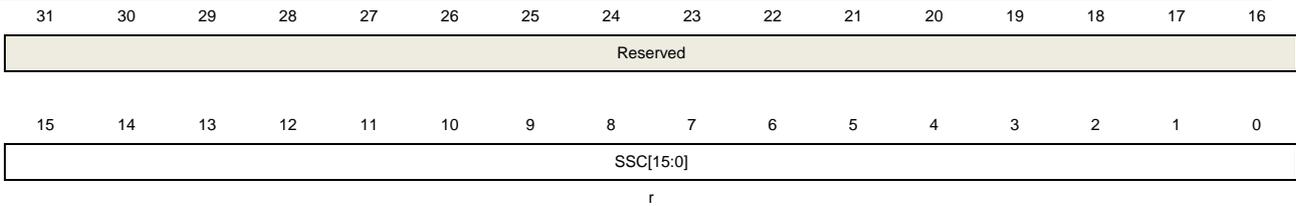
Backup domain reset: 0x0000 0000

System reset: no effect

This register will record the calendar date when TSF is set to 1.

Reset TSF bit will also clear this register.

This register has to be accessed by word(32-bit)

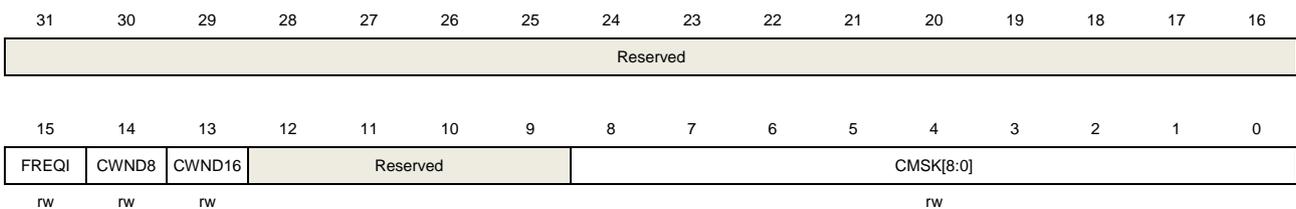


Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	SSC[15:0]	Sub second value This value is the counter value of synchronous prescaler when TSF is set to 1.

#### 14.4.13. High resolution frequency compensation register (RTC\_HRFC)

Address offset: 0x3C  
Backup domain reset: 0x0000 0000  
System Reset: no effect  
This register is write protected.

This register has to be accessed by word(32-bit)



Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15	FREQI	Increase RTC frequency by 488.5PPM 0: No effect 1: One RTCCLK pulse is inserted every 2 <sup>11</sup> pulses. This bit should be used in conjunction with CMSK bit. If the input clock frequency is 32.768KHz, the number of RTCCLK pulses added during 32s calibration window is (512 * FREQI) - CMSK
14	CWND8	Frequency compensation window 8 second selected 0:No effect 1:Calibration window is 8 second <b>Note:</b> When CWND8=1, CMSK[1:0] are stuck at "00".
13	CWND16	Frequency compensation window 16 second selected 0:No effect 1:Calibration window is 16 second <b>Note:</b> When CWND16=1, CMSK[0] are stuck at "0".
12:9	Reserved	Must be kept at reset value
8:0	CMSK[8:0]	Calibration mask number The number of mask pulse out of 2 <sup>20</sup> RTCCLK pulse. This feature will decrease the frequency of calendar with a resolution of 0.9537 PPM.

#### 14.4.14. Tamper register (RTC\_TAMP)

Address offset: 0x40  
Backup domain reset: 0x0000 0000

System reset: no effect

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								PC15MDE	PC15VAL	PC14MDE	PC14VAL	PC13MDE	PC13VAL	Reserved	
								rw	rw	rw	rw	rw	rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DISPU	PRCH[1:0]		FLT[1:0]		FREQ[2:0]		TPTS	Reserved			TP1EG	TP1EN	TPIE	TP0EG	TP0EN
rw	rw		rw		rw		rw				rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:24	Reserved	Must be kept at reset value
23	PC15MDE	PC15 Mode 0:No effect 1:Force PC15 to push-pull output if LXTAL is disable
22	PC15VAL	PC15 Value Only valid when LXTAL is disabled and PC15MDE=1,PC15 output this bit data.
21	PC14MDE	PC14 Mode 0:No effect 1:Force PC14 to push-pull output if LXTAL is disable
20	PC14VAL	PC14 Value Only valid when LXTAL is disabled and PC14MDE=1,PC14 output this bit data.
19	PC13MDE	PC13 Mode 0:No effect 1:Force PC13 to push-pull output if all RTC alternate functions are disabled.
18	PC13VAL	PC13 value or alarm output type value When PC13 is used to output alarm: 0:PC13 is in open-drain output type 1:PC13 is in push-pull output type When all RTC alternate functions are disabled and PC13MDE=1: 0:PC13 output 0 1:PC13 output 1
17:16	Reserved	Must be kept at reset value
15	DISPU	RTC_TAMPx pull up disable bit 0:Enable inner pull-up before sampling for pre-charge RTC_TAMPx pin 1:Disable pre-charge duration
14:13	PRCH[1:0]	Pre-charge duration time of RTC_TAMPx This setting determines the pre-charge time before each sampling. 0x0:1 RTC clock 0x1:2 RTC clock 0x2:4 RTC clock 0x3:8 RTC clock
12:11	FLT[1:0]	RTC_TAMPx filter count setting This bit determines the tamper sampling type and the number of consecutive sample. 0x0: Detecting tamper event using edge mode. Pre-charge duration is disabled automatically 0x1: Detecting tamper event using level mode.2 consecutive valid level samples will make an effective tamper event 0x2:Detecting tamper event using level mode.4 consecutive valid level samples will make an effective tamper event 0x3:Detecting tamper event using level mode.8 consecutive valid level samples will make an effective tamper event
10:8	FREQ[2:0]	Sampling frequency of tamper event detection 0x0: Sample once every 32768 RTCCLK(1Hz if RTCCLK=32.768KHz)

		0x1: Sample once every 16384 RTCCLK(2Hz if RTCCLK=32.768KHz)
		0x2: Sample once every 8192 RTCCLK(4Hz if RTCCLK=32.768KHz)
		0x3: Sample once every 4096 RTCCLK(8Hz if RTCCLK=32.768KHz)
		0x4: Sample once every 2048 RTCCLK(16Hz if RTCCLK=32.768KHz)
		0x5: Sample once every 1024 RTCCLK(32Hz if RTCCLK=32.768KHz)
		0x6: Sample once every 512 RTCCLK(64Hz if RTCCLK=32.768KHz)
		0x7: Sample once every 256 RTCCLK(128Hz if RTCCLK=32.768KHz)
7	TPTS	Make tamper function used for timestamp function 0:No effect 1:TSF is set when tamper event detected even TSEN=0
6:5	Reserved	Must be kept at reset value
4	TP1EG	Tamper 1 event trigger edge If tamper detection is in edge mode(FLT =0): 0: Rising edge triggers a tamper detection event 1: Falling edge triggers a tamper detection event If tamper detection is in level mode(FLT !=0): 0: Low level triggers a tamper detection event 1: High level triggers a tamper detection event
3	TP1EN	Tamper 1 detection enable 0:Disable tamper 1 detection function 1:Enable tamper 1 detection function
2	TPIE	Tamper detection interrupt enable 0:Disable tamper interrupt 1:Enable tamper interrupt
1	TPOEG	Tamper 0 event trigger edge If tamper detection is in edge mode(FLT =0): 0: Rising edge triggers a tamper detection event 1: Falling edge triggers a tamper detection event If tamper detection is in level mode(FLT !=0): 0: Low level triggers a tamper detection event 1: High level triggers a tamper detection event
0	TPOEN	Tamper 0 detection enable 0:Disable tamper 0 detection function 1:Enable tamper 0 detection function

**Note:** It's strongly recommended that reset the TPxEN before change the tamper configuration.

#### 14.4.15. Alarm 0 sub second register (RTC\_ALRM0SS)

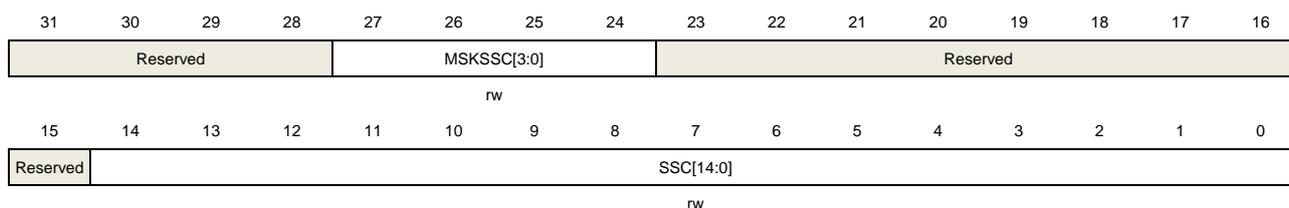
Address offset: 0x44

Backup domain reset: 0x0000 0000

System reset: no effect

This register is write protected and can only be wrote when ALRM0EN=0 or INITM=1

This register has to be accessed by word(32-bit)



Bits	Fields	Descriptions
31:28	Reserved	Must be kept at reset value
27:24	MSKSSC[3:0]	Mask control bit of SSC 0x0: Mask alarm sub second setting. The alarm asserts at every second time point if all

the rest alarm fields are matched.

- 0x1: SSC[0] is to be compared and all others are ignored
- 0x2: SSC[1:0] is to be compared and all others are ignored
- 0x3: SSC[2:0] is to be compared and all others are ignored
- 0x4: SSC[3:0] is to be compared and all others are ignored
- 0x5: SSC[4:0] is to be compared and all others are ignored
- 0x6: SSC[5:0] is to be compared and all others are ignored
- 0x7: SSC[6:0] is to be compared and all others are ignored
- 0x8: SSC[7:0] is to be compared and all others are ignored
- 0x9: SSC[8:0] is to be compared and all others are ignored
- 0xA: SSC[9:0] is to be compared and all others are ignored
- 0xB: SSC[10:0] is to be compared and all others are ignored
- 0xC: SSC[11:0] is to be compared and all others are ignored
- 0xD: SSC[12:0] is to be compared and all others are ignored
- 0xE: SSC[13:0] is to be compared and all others are ignored
- 0xF: SSC[14:0] is to be compared and all others are ignored

**Note:** The bit 15 of synchronous counter (SSC[15] in RTC\_SS) is never compared.

23:15	Reserved	Must be kept at reset value
14:0	SSC[14:0]	Alarm sub second value This value is the alarm sub second value which is to be compared with synchronous prescaler counter SSC. Bit number is controlled by MSKSSC bits.

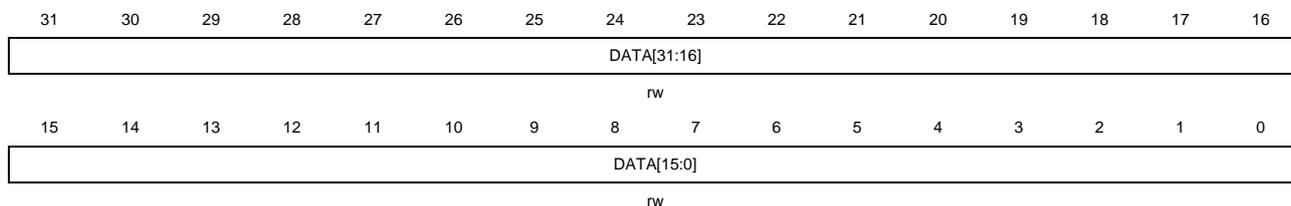
#### 14.4.16. Backup registers (RTC\_BKPx) (x=0..4)

Address offset: 0x50~0x60

Backup domain reset: 0x0000 0000

System reset: no effect

This register has to be accessed by word(32-bit)



Bits	Fields	Descriptions
31:0	DATA[31:0]	Data These registers can be wrote or read by software. The content remains valid even in power saving mode because they can powered-on by V <sub>BAT</sub> . Tamper detection flag TPxF assertion will reset these registers. Also when the FMC readout protection disables will reset these registers.

## 15. Timer (TIMERx)

**Table 15-1. Timers (TIMERx) are divided into six sorts**

TIMER	TIMER0	TIMER1/2	TIMER13	TIMER14	TIMER15/16	TIMER5
TYPE	Advanced	General-L0	General-L2	General-L3	General-L4	Basic
Prescaler	16-bit	16-bit	16-bit	16-bit	16-bit	16-bit
Counter	16-bit	32-bit(TIMER1) 16-bit(TIMER2)	16-bit	16-bit	16-bit	16-bit
Count mode	UP,DOWN, Center-aligned	UP,DOWN, Center-aligned	UP ONLY	UP ONLY	UP ONLY	UP ONLY
Repetition	•	×	×	•	•	×
CH Capture/ Compare	4	4	1	2	1	0
Complementary & Dead-time	•	×	×	•	•	×
Break	•	×	×	•	•	×
Single Pulse	•	•	×	•	•	•
Quadrature Decoder	•	•	×	×	×	×
Slave Controller	•	•	×	•	×	×
Inter connection	• <sup>(1)</sup>	• <sup>(2)</sup>	×	• <sup>(3)</sup>	×	TRGO TO DAC
DMA	•	•	×	•	•	• <sup>(4)</sup>
Debug Mode	•	•	•	•	•	•

(1) TIMER0 IT10: TIMER14\_TRGO IT11: TIMER1\_TRGO IT12: TIMER2\_TRGO IT13: 0

(2) TIMER1 IT10: TIMER0\_TRGO IT11: TIMER14\_TRGO IT12: TIMER2\_TRGO IT13: 0  
 TIMER2 IT10: TIMER0\_TRGO IT11: TIMER1\_TRGO IT12: TIMER14\_TRGO IT13: 0

(3) TIMER14 IT10: TIMER1\_TRGO IT11: TIMER2\_TRGO IT12: 0 IT13: 0

(4) Only update events will generate DMA request. Note that TIMER5 do not have DMA configuration registers.

## 15.1. Advanced timer (TIMERx,x=0)

### 15.1.1. Overview

The advanced timer module (TIMER0) is a four-channel timer that supports both input capture and output compare. They can generate PWM signals to control motor or be used for power management applications. The advanced timer has a 16-bit counter that can be used as an unsigned counter.

In addition, the advanced timers can be programmed and be used for counting, their external events can be used to drive other timers.

Timer also includes a dead-time Insertion module which issuitable for motor control applications.

Timers are completely independent with each other, but they may be synchronized to provide a larger timer with their counters incrementing in unison.

### 15.1.2. Characteristics

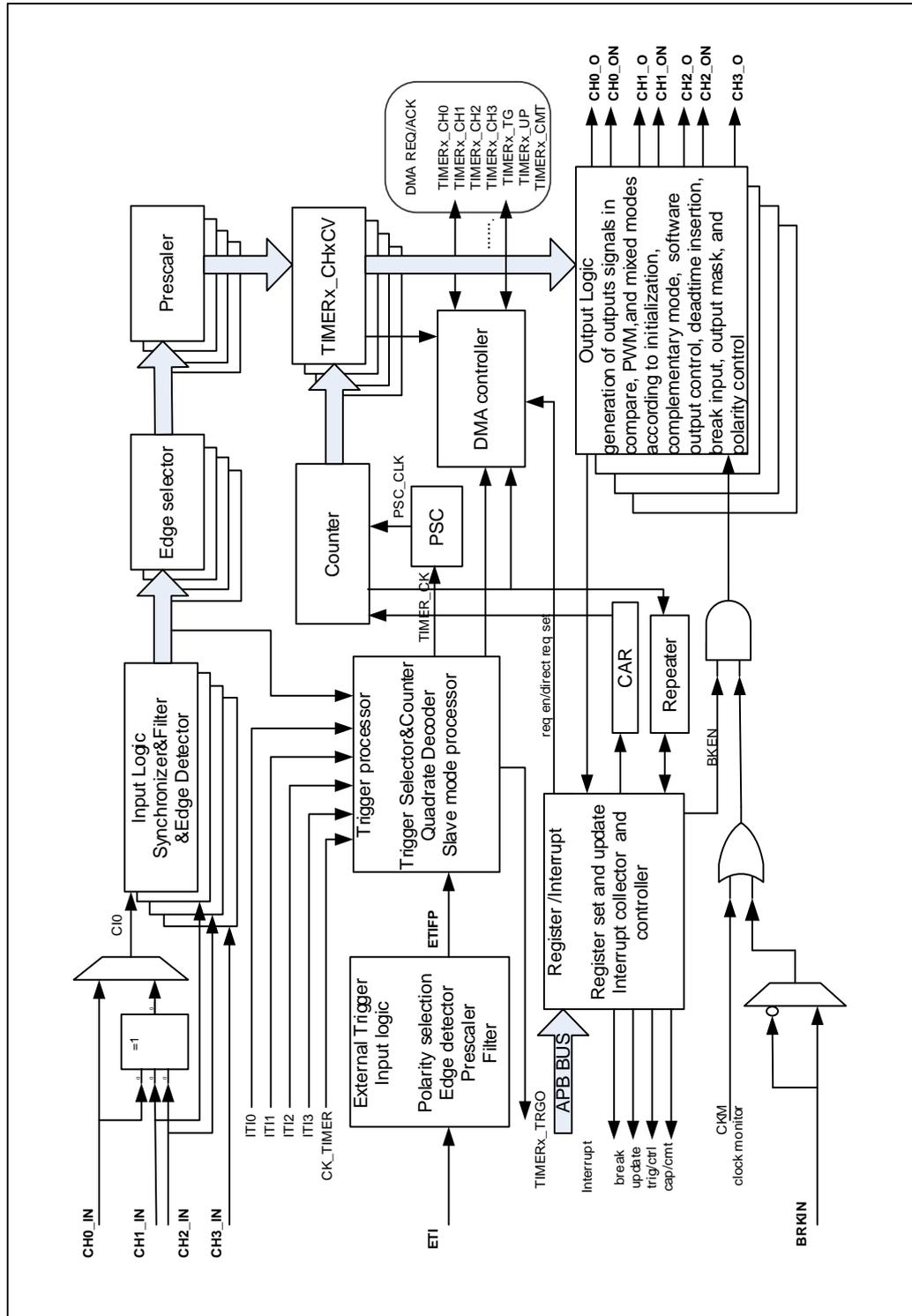
- Total channel num: 4.
- Counter width: 16 bit.
- Source of counter clock is selectable:  
internal clock, internal trigger, external input, external trigger.
- Multiple counter modes: count up, count down, count up/down.
- Quadrature Decoder: used to track motion and determine both rotation direction and position.
- Hall sensor: for 3-phase motor control.
- Programmable prescaler: 16 bit.The factor can be changed on the go.
- Each channel is user-configurable:  
input capture mode, output compare mode, programmable PWM mode, single pulse mode
- Programmable dead time insertion.
- Auto reload function.
- Programmable counter repetition function.
- Break input.
- Interrupt output or DMA request on: update, trigger event, compare/capture event, and break input.
- Daisy chaining of timer modules allows a single timer to initiate multiple timers.
- Timer synchronization allows selected timers to start counting on the same clock cycle.
- Timer Master/Slave mode controller.

### 15.1.3. Block diagram

[Figure 15-1. Advanced timer block diagram](#) provides details of the internal configuration of

the advanced timer.

Figure 15-1. Advanced timer block diagram



### 15.1.4. Function overview

#### Clock selection

The advanced timer has the capability of being clocked by either the TIMER\_CK or an alternate clock source controlled by SMC (TIMERx\_SMCFG bit [2:0]).

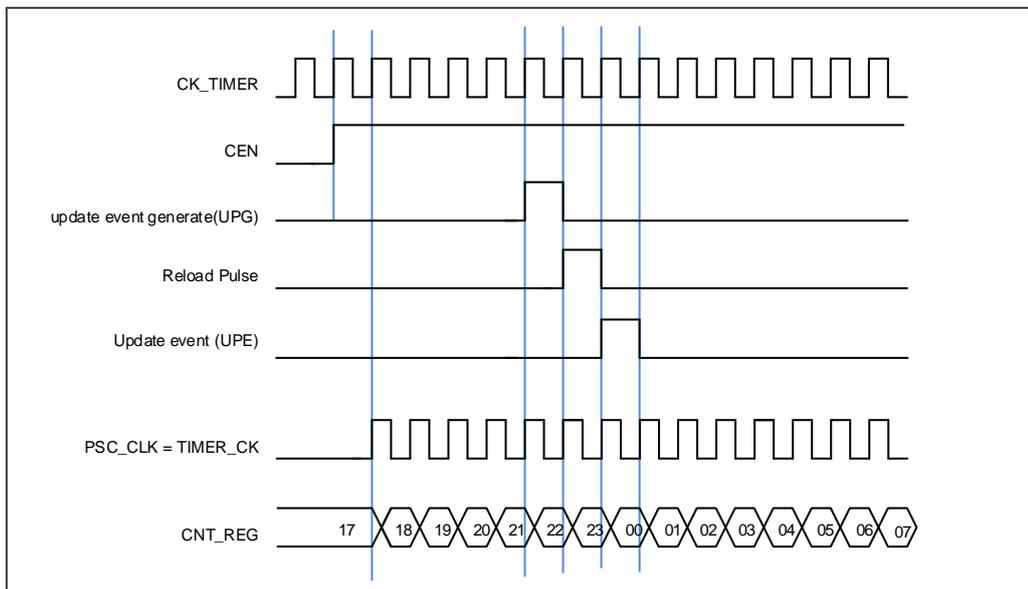
- SMC [2:0] == 3'b000. Internal clock CK\_TIMER is selected as timer clock source which is from module RCU.

The default clock source is the CK\_TIMER for driving the counter prescaler when the slave mode is disabled (SMC [2:0] == 3'b000). When the CEN is set, the CK\_TIMER will be divided by PSC value to generate PSC\_CLK.

In this mode, the TIMER\_CK, which drives counter's prescaler to count, is equal to CK\_TIMER which is from RCU.

If the slave mode controller is enabled by setting SMC [2:0] in the TIMERx\_SMCFG register to an available value including 0x1, 0x2, 0x3 and 0x7, the prescaler is clocked by other clock sources selected by the TRGS [2:0] in the TIMERx\_SMCFG register, details as follows. When the slave mode selection bits SMC [2:0] are set to 0x4, 0x5 or 0x6, the internal clock TIMER\_CK is the counter prescaler driving clock source.

**Figure 15-2. Normal mode, internal clock divided by 1**



- SMC [2:0] == 3'b111 (external clock mode 0). External input pin is selected as timer clock source

The TIMER\_CK, which drives counter's prescaler to count, can be triggered by the event of rising or falling edge on the external pin TIMERx\_CH0/TIMERx\_CH1. This mode can be selected by setting SMC [2:0] to 0x7 and the TRGS [2:0] to 0x4, 0x5 or 0x6.

And, the counter prescaler can also be driven by rising edge on the internal trigger input pin ITIO/1/2/3. This mode can be selected by setting SMC [2:0] to 0x7 and the TRGS [2:0] to 0x0, 0x1, 0x2 or 0x3.

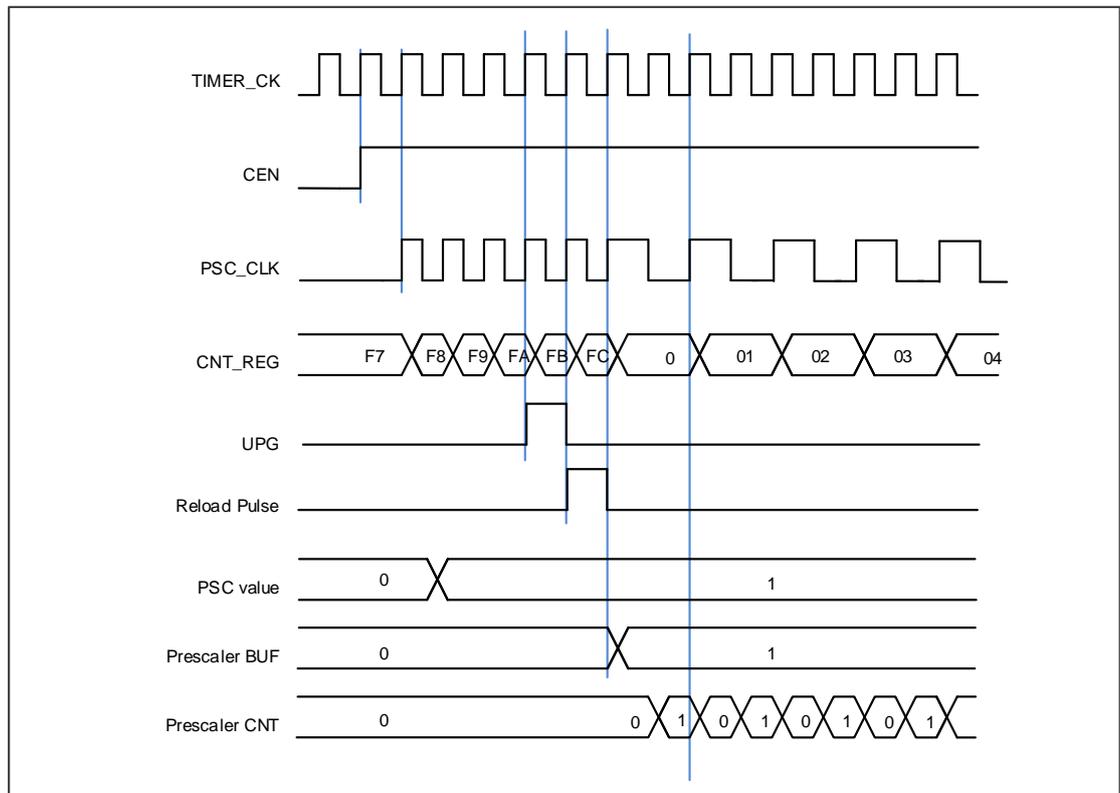
- SMC1== 1'b1 (external clock mode 1). External input is selected as timer clock source (ETI)

The TIMER\_CK, which drives counter's prescaler to count, can be triggered by the event of rising or falling edge on the external pin ETI. This mode can be selected by setting the SMC1 bit in the TIMEx\_SMCFG register to 1. The other way to select the ETI signal as the clock source is to set the SMC [2:0] to 0x7 and the TRGS [2:0] to 0x7 respectively. Note that the ETI signal is derived from the ETI pin sampled by a digital filter. When the ETI signal is selected as clock source, the trigger controller including the edge detection circuitry will generate a clock pulse on each ETI signal rising edge to clock the counter prescaler.

## Prescaler

The prescaler can divide the timer clock (TIMER\_CK) to a counter clock (PSC\_CLK) by any factor between 1 and 65536. It is controlled by prescaler register (TIMEx\_PSC) which can be changed on the go but is taken into account at the next update event.

**Figure 15-3. Counter timing diagram with prescaler division change from 1 to 2**



### Up counting mode

In this mode, the counter counts up continuously from 0 to the counter-reload value, which is defined in the `TIMERx_CAR` register, in a count-up direction. Once the counter reaches the counter reload value, the counter restarts from 0. If the repetition counter is set, the update events will be generated after  $(\text{TIMERx\_CREP}+1)$  times of overflow. Otherwise the update event is generated each time when overflows. The counting direction bit `DIR` in the `TIMERx_CTL0` register should be set to 0 for the up counting mode.

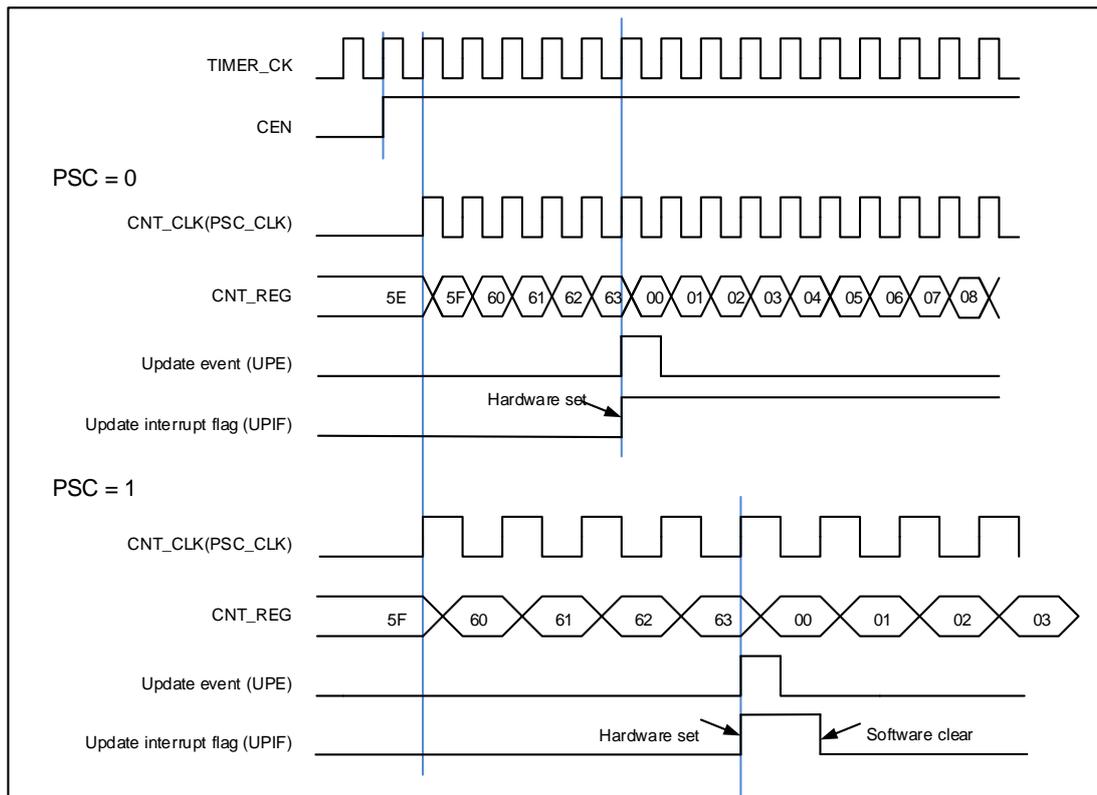
Whenever, if the update event software trigger is enabled by setting the `UPG` bit in the `TIMERx_SWEVG` register, the counter value will be initialized to 0 and generates an update event.

If set the `UPDIS` bit in `TIMERx_CTL0` register, the update event is disabled.

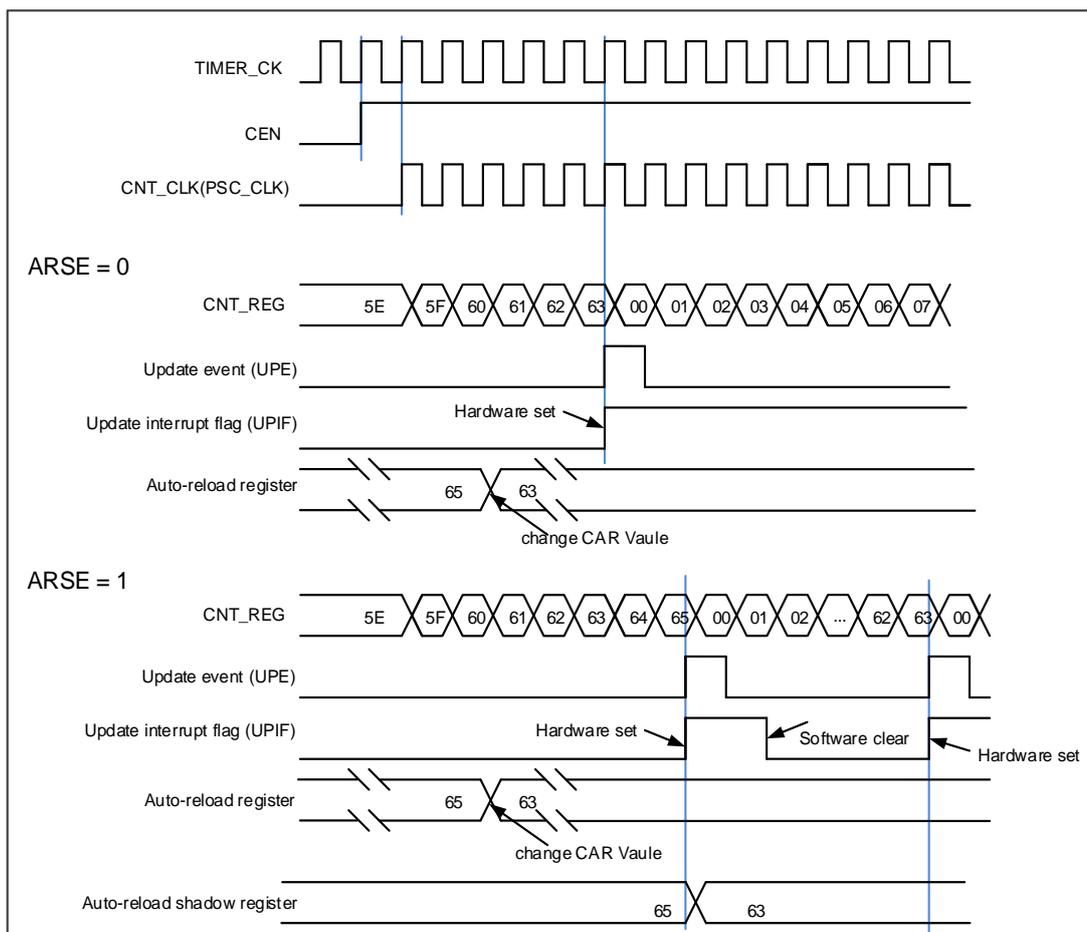
When an update event occurs, all the registers (repetition counter, auto reload register, prescaler register) are updated.

[Figure 15-4. Up-counter timechart, PSC=0/1](#) show some examples of the counter behavior for different clock prescaler factor when `TIMERx_CAR=0x63`.

**Figure 15-4. Up-counter timechart, PSC=0/1**



**Figure 15-5. Up-counter timechart, change TIMERx\_CAR on the go**



**Down counting mode**

In this mode, the counter counts down continuously from the counter-reload value, which is defined in the TIMERx\_CAR register, to 0 in a count-down direction. Once the counter reaches to 0, the counter restarts to count again from the counter-reload value. If the repetition counter is set, the update event will be generated after (TIMERx\_CREP+1) times of underflow. Otherwise the update event is generated each time when underflows. The counting direction bit DIR in the TIMERx\_CTL0 register should be set to 1 for the down-counting mode.

When the update event is set by the UPG bit in the TIMERx\_SWEVG register, the counter value will be initialized to the counter-reload value and generates an update event.

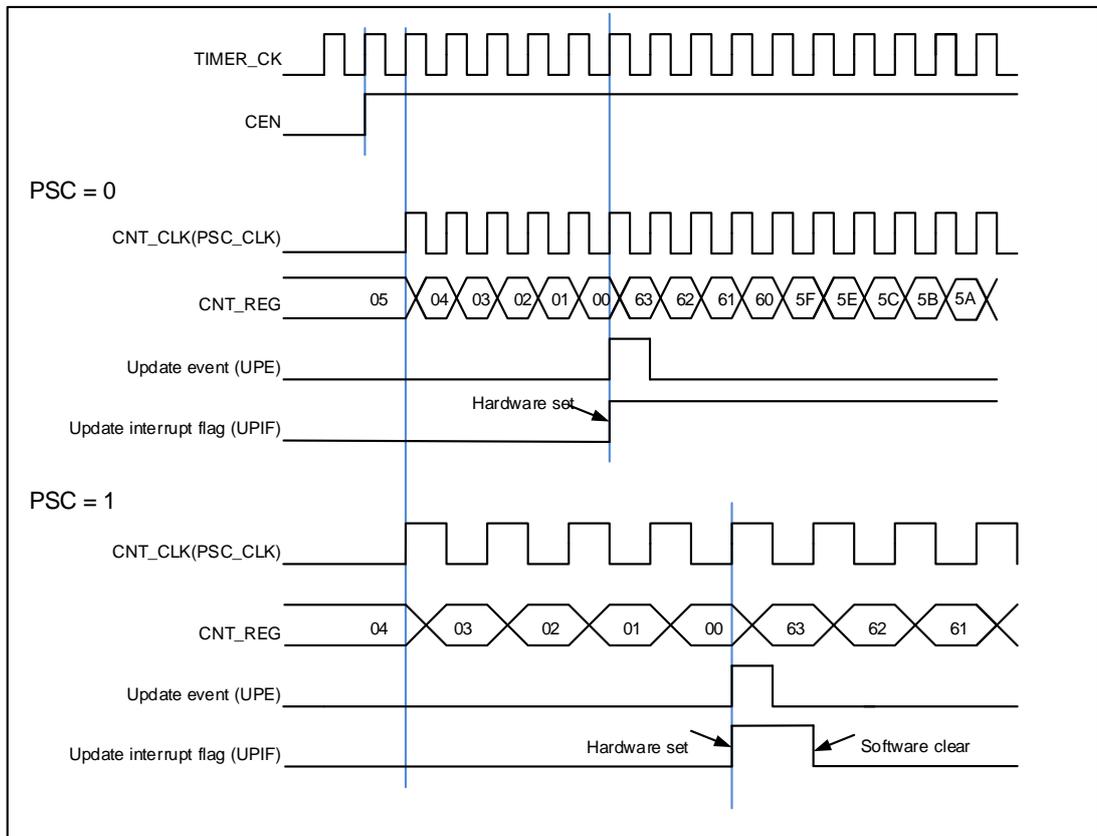
If set the UPDIS bit in TIMERx\_CTL0 register, the update event is disabled.

When an update event occurs, all the registers (repetition counter, auto reload register, prescaler register) are updated.

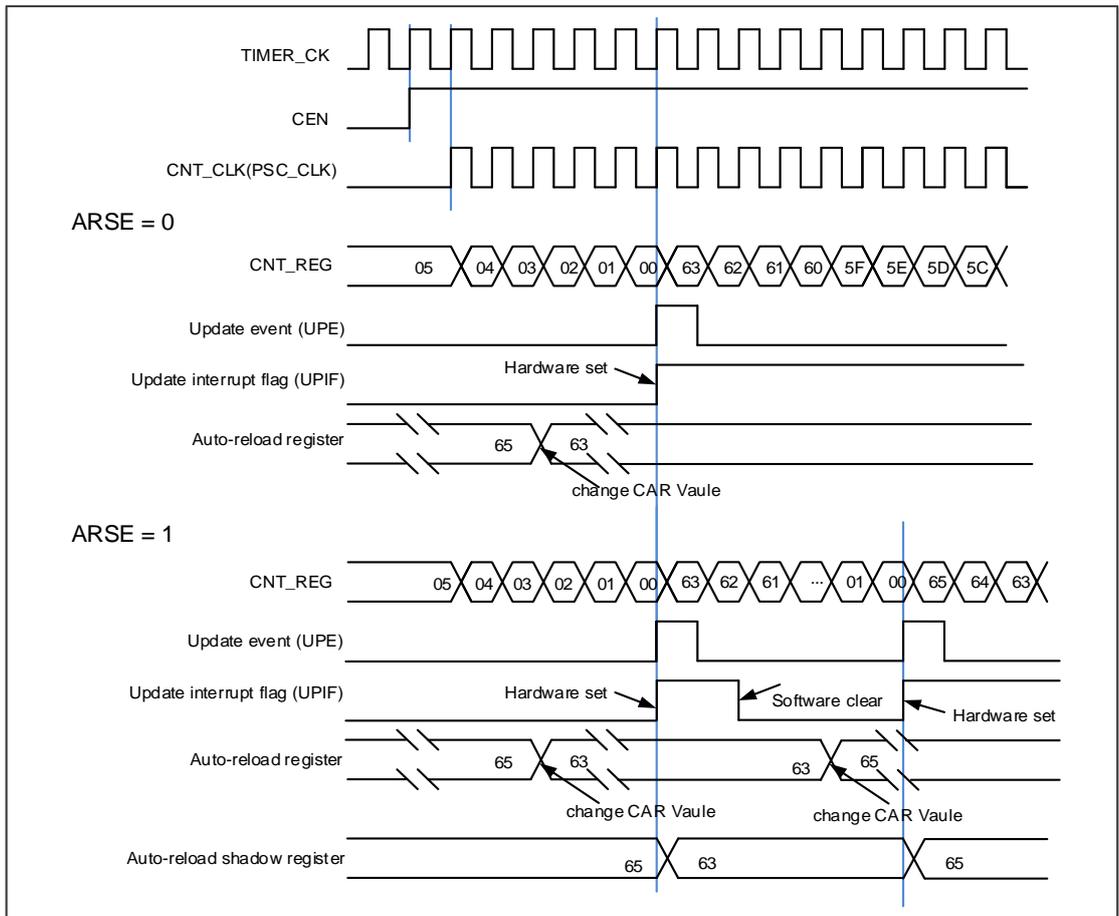
**Figure 15-6. Down-counter timechart, PSC=0/1** show some examples of the counter

behavior in different clock frequencies when  $TIMERx\_CAR=0x63$ .

**Figure 15-6. Down-counter timechart, PSC=0/1**



**Figure 15-7. Down-counter timechart, change TIMERx\_CAR on the go**



### Center-aligned counting mode

In the center-aligned counting mode, the counter counts up from 0 to the counter-reload value and then counts down to 0 alternatively. The TIMER module generates an overflow event when the counter counts to the counter-reload value subtract 1 in the up-counting direction and generates an underflow event when the counter counts to 1 in the down-counting direction. The counting direction bit DIR in the TIMERx\_CTL0 register is read-only and indicates the counting direction when in the center-aligned mode. The counting direction is updated by hardware automatically.

Setting the UPG bit in the TIMERx\_SWEVG register will initialize the counter value to 0 and generates an update event irrespective of whether the counter is counting up or down in the center-align counting mode.

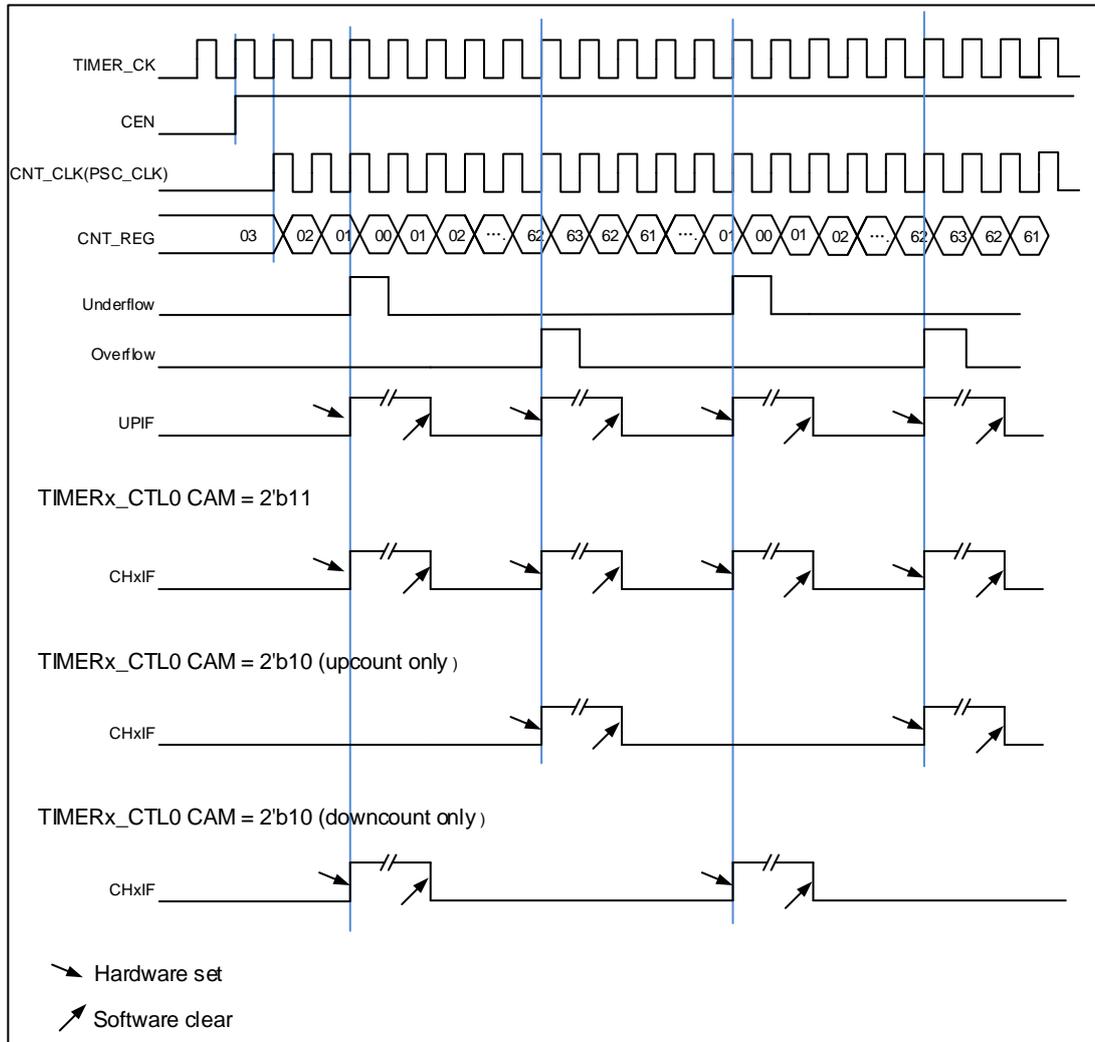
The UPIF bit in the TIMERx\_INTF register can be set to 1 either when an underflow event or an overflow event occurs. While the CHxIF bit is associated with the value of CAM in TIMERx\_CTL0. The details refer to [Figure 15-8. Center-aligned counter timechart](#)

If set the UPDIS bit in the TIMERx\_CTL0 register, the update event is disabled.

When an update event occurs, all the registers (repetition counter, auto-reload register, prescaler register) are updated.

**Figure 15-8. Center-aligned counter timechart** show some examples of the counter behavior when  $TIMERx\_CAR=0x63$ .  $TIMERx\_PSC=0x0$

**Figure 15-8. Center-aligned counter timechart**



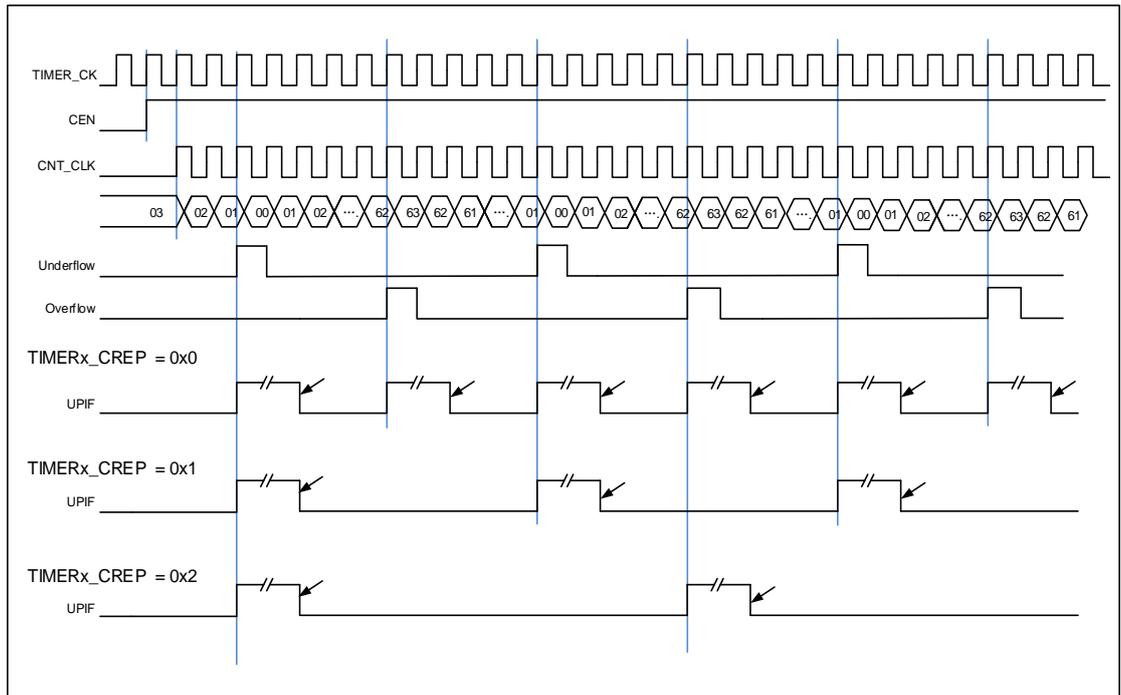
### Counter repetition

Counter Repetition is used to generate update event or updates the timer registers only after a given number (N+1) of cycles of the counter, where N is CREP in  $TIMERx\_CREP$  register. The repetition counter is decremented at each counter overflow in up-counting mode, at each counter underflow in down-counting mode or at each counter overflow and at each counter underflow in center-aligned mode.

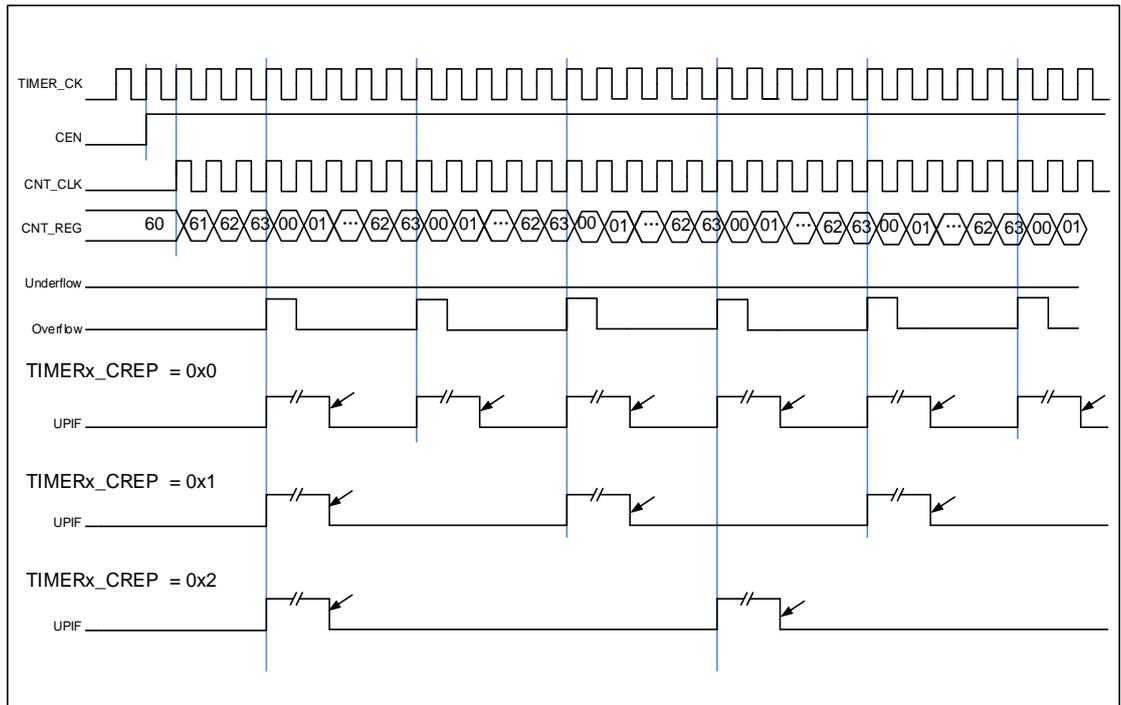
Setting the UPG bit in the  $TIMERx\_SWEVG$  register will reload the content of CREP in  $TIMERx\_CREP$  register and generate an update event.

For odd values of CREP in center-aligned mode, the update event occurs either on the overflow or on the underflow depending on when the CREP register was written and when the counter was started. The update event generated at overflow when the CREP was written before starting the counter, and generated at underflow when the CREP was written after starting the counter.

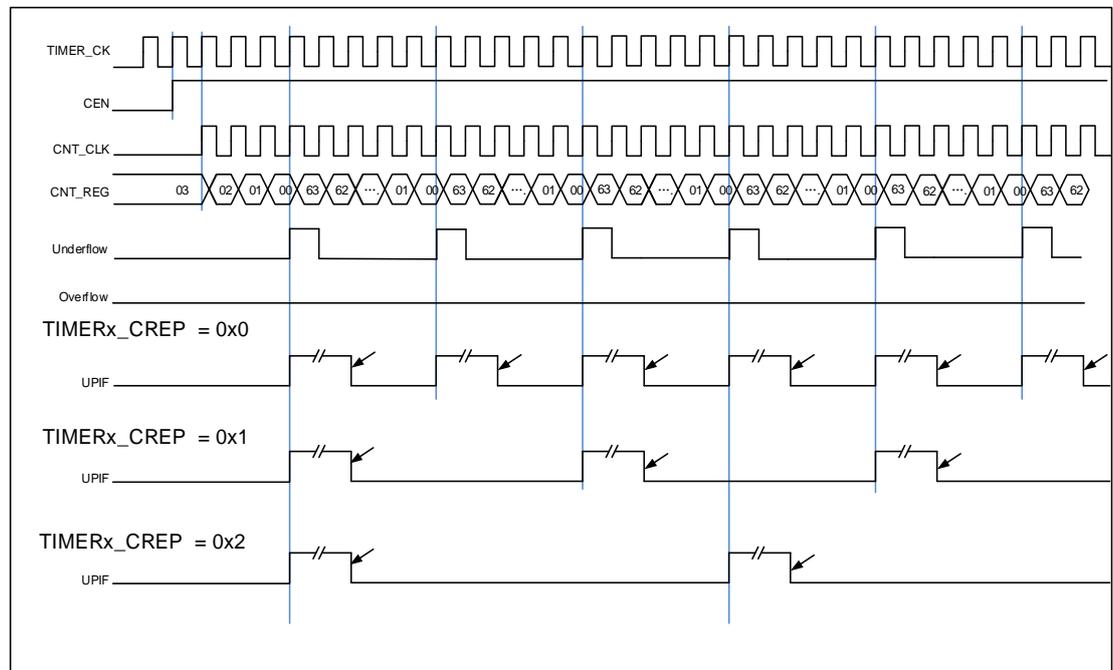
**Figure 15-9. Repetition timecart for center-aligned counter**



**Figure 15-10. Repetition timechart for up-counter**



**Figure 15-11. Repetition timechart for down-counter**



### Capture/compare channels

The advanced timer has four independent channels which can be used as capture inputs or compare match outputs. Each channel is built around a channel capture compare register



**Step2:** Edge selection. (CHxP/CHxNP in TIMERx\_CHCTL2)  
Rising or falling edge, choose one by CHxP/CHxNP.

**Step3:** Capture source selection. (CHxMS in TIMERx\_CHCTL0)  
As soon as you select one input capture source by CHxMS, you have set the channel to input mode (CHxMS!=0x0) and TIMERx\_CHxCV cannot be written any more.

**Step4:** Interrupt enable. (CHxIE and CHxDEN in TIMERx\_DMAINTEN)  
Enable the related interrupt enable; you can get the interrupt and DMA request.

**Step5:** Capture enables. (CHxEN in TIMERx\_CHCTL2)

**Result:** when you wanted input signal is got, TIMERx\_CHxCV will be set by counter's value. And CHxIF is asserted. If the CHxIF is high, the CHxOF will be asserted also. The interrupt and DMA request will be asserted based on the configuration of CHxIE and CHxDEN in TIMERx\_DMAINTEN

**Direct generation:** if you want to generate a DMA request or Interrupt, you can set CHxG by software directly.

The input capture mode can be also used for pulse width measurement from signals on the TIMERx\_CHx pins. For example, PWM signal connect to CI0 input. Select channel 0 capture signals to CI0 by setting CH0MS to 2'b01 in the channel control register (TIMERx\_CHCTL0) and set capture on rising edge. Select channel 1 capture signal to CI0 by setting CH1MS to 2'b10 in the channel control register (TIMERx\_CHCTL0) and set capture on falling edge. The counter set to restart mode and restart on channel 0 rising edge. Then the TIMERx\_CH0CV can measure the PWM period and the TIMERx\_CH1CV can measure the PWM duty.

## ■ Output compare mode

In output compare mode, the TIMERx can generate timed pulses with programmable position, polarity, duration and frequency. When the counter matches the value in the CHxVAL register of an output compare channel, the channel (n) output can be set, cleared, or toggled based on CHxCOMCTL. When the counter reaches the value in the CHxVAL register, the CHxIF bit is set and the channel (n) interrupt is generated if CHxIE = 1. And the DMA request will be assert, if CHxDEN=1.

So the process can be divided to several steps as below:

**Step1:** Clock Configuration. Such as clock source, clock prescaler and so on.

**Step2:** Compare mode configuration.

- \* Set the shadow enable mode by CHxCOMSEN
- \* Set the output mode (Set/Clear/Toggle) by CHxCOMCTL.
- \* Select the active high polarity by CHxP/CHxNP
- \* Enable the output by CHxEN

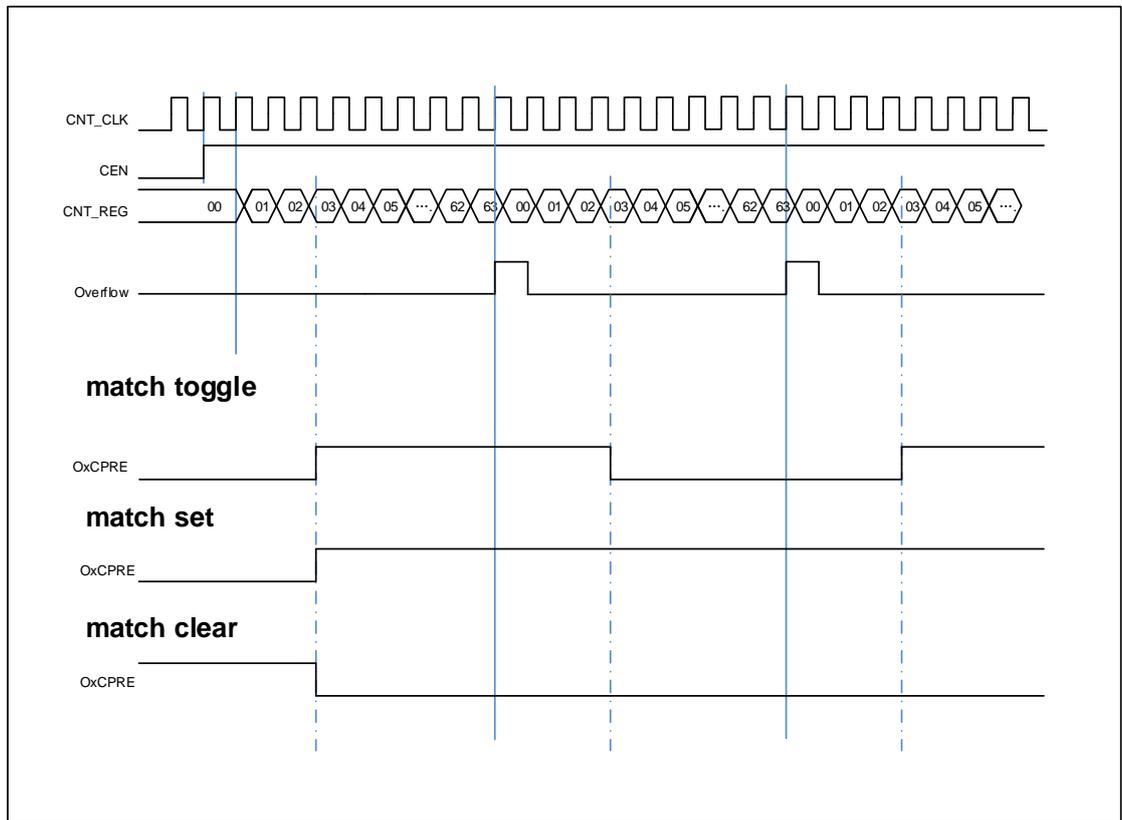
**Step3:** Interrupt/DMA-request enables configuration by CHxIE/ CHxDEN

**Step4:** Compare output timing configuration by TIMERx\_CAR and TIMERx\_CHxCV  
About the CHxVAL; you can change it on the go to meet the waveform you expected.

**Step5:** Start the counter by CEN.

The timechart below show the three compare modes toggle/set/clear. CAR=0x63, CHxVAL=0x3

**Figure 15-13. Output-compare under three modes**



**PWM mode**

In the output PWM mode (by setting the CHxCOMCTL bits to 3'b110 (PWM mode0) or to 3'b111(PWM mode1), the channel can generate PWM waveform according to the TIMERx\_CAR registers and TIMERx\_CHxCV registers.

Based on the counter mode, we can also divide PWM into EAPWM (Edge aligned PWM) and CAPWM (Centre aligned PWM).

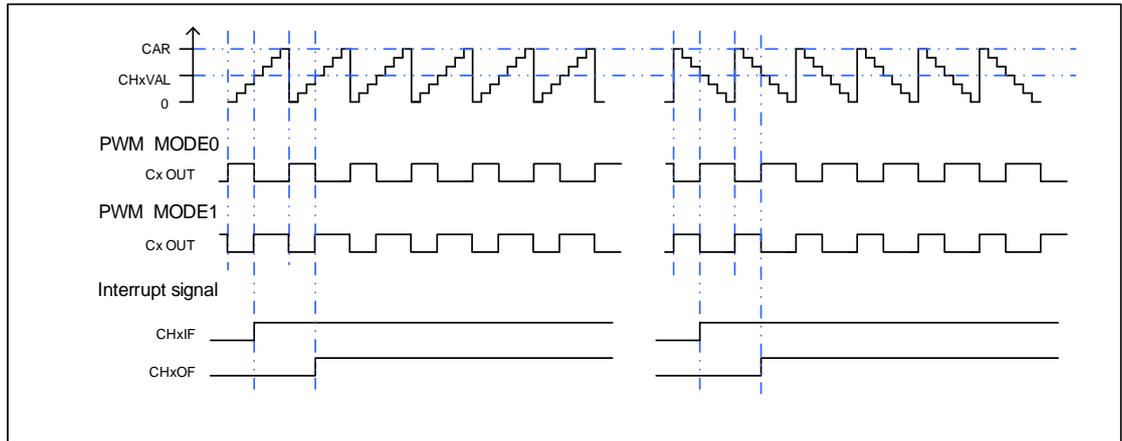
The EAPWM period is determined by TIMERx\_CAR and duty cycle is determined by TIMERx\_CHxCV. [Figure 15-14. EAPWM timechart](#) shows the EAPWM output and interrupts waveform.

The CAPWM period is determined by 2\*TIMERx\_CAR, and duty cycle is by 2\*TIMERx\_CHxCV. [Figure 15-15. CAPWM timechart](#) shows the CAPWM output and interrupts waveform.

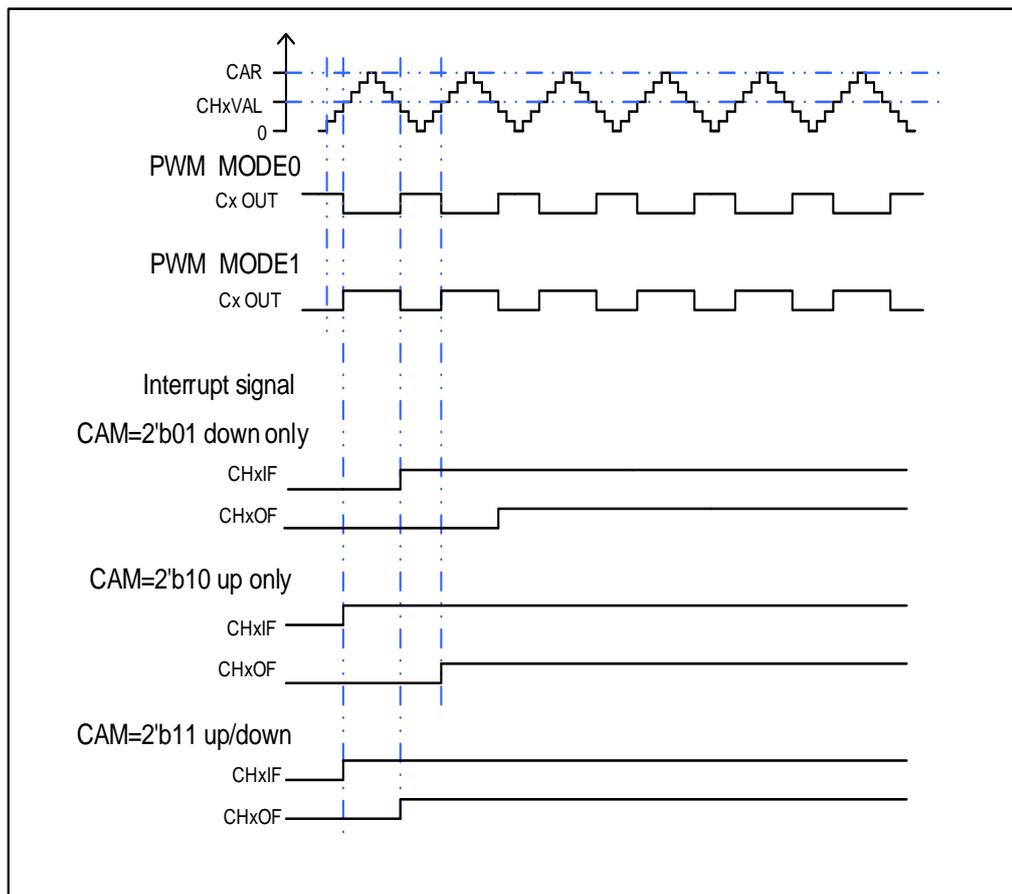
If TIMERx\_CHxCV is greater than TIMERx\_CAR, the output will be always active under PWM mode0 (CHxCOMCTL==3'b110).

And if `TIMERx_CHxCV` is equal to zero, the output will be always inactive under PWM mode0 (`CHxCOMCTL==3'b110`).

**Figure 15-14. EAPWM timechart**



**Figure 15-15. CAPWM timechart**



## Channel output reference signal

When the TIMERx is used in the compare match output mode, the OxCPRE signal (Channel x Output prepare signal) is defined by setting the CHxCOMCTL field. The OxCPRE signal has several types of output function. These include, keeping the original level by setting the CHxCOMCTL field to 0x00, set to 1 by setting the CHxCOMCTL field to 0x01, set to 0 by setting the CHxCOMCTL field to 0x02 or signal toggle by setting the CHxCOMCTL field to 0x03 when the counter value matches the content of the TIMERx\_CHxCV register.

The PWM mode 0 and PWM mode 1 outputs are also another kind of OxCPRE output which is setup by setting the CHxCOMCTL field to 0x06/0x07. In these modes, the OxCPRE signal level is changed according to the counting direction and the relationship between the counter value and the TIMERx\_CHxCV content. With regard to a more detail description refer to the relative bit definition.

Another special function of the OxCPRE signal is a forced output which can be achieved by setting the CHxCOMCTL field to 0x04/0x05. Here the output can be forced to an inactive/active level irrespective of the comparison condition between the counter and the TIMERx\_CHxCV values.

The OxCPRE signal can be forced to 0 when the ETIFE signal is derived from the external ETI pin and when it is set to a high level by setting the CHxCOMCEN bit to 1 in the TIMERx\_CHCTL0 register. The OxCPRE signal will not return to its active level until the next update event occurs.

## Outputs complementary

Function of complementary is for a pair of CHx\_O and CHx\_ON. Those two output signals cannot be active at the same time. The TIMERx has 4 channels, but only the first three channels have this function. The complementary signals CHx\_O and CHx\_ON are controlled by a group of parameters: the CHxEN and CHxNEN bits in the TIMERx\_CHCTL2 register and the POEN, ROS, IOS, ISOx and ISOxN bits in the TIMERx\_CCHP and TIMERx\_CTL1 registers. The outputs polarity is determined by CHxP and CHxNP bits in the TIMERx\_CHCTL2 register.

**Table 15-2. Complementary outputs controlled by parameters**

Complementary Parameters					Output Status	
POEN	ROS	IOS	CHxEN	CHxNEN	CHx_O	CHx_ON
0	0/1	0	0	0	CHx_O / CHx_ON = LOW CHx_O / CHx_ON output disable.	
				1	CHx_O = CHxP CHx_ON = CHxNP	
			1	0	CHx_O/CHx_ON output disable.	
				1	If clock is enable:	

Complementary Parameters					Output Status	
POEN	ROS	IOS	CHxEN	CHxNEN	CHx_O	CHx_ON
					CHx_O = ISOx CHx_ON = ISOxN	
		1	0	0	CHx_O = CHxP CHx_ON = CHxNP CHx_O/CHx_ON output disable.	
				1	CHx_O = CHxP CHx_ON = CHxNP	
			1	0	CHx_O/CHx_ON output enable.	
				1	If clock is enable: CHx_O = ISOx CHx_ON = ISOxN	
1	0	0/1	0	0	CHx_O/CHx_ON = LOW CHx_O/CHx_ON output disable.	
				1	CHx_O = LOW CHx_O output disable.	CHx_ON=OxCPRE $\oplus$ CHxNP CHx_ON output enable
			1	0	CHx_O=OxCPRE $\oplus$ CHxP CHx_O output enable	CHx_ON = LOW CHx_ON output disable.
				1	CHx_O=OxCPRE $\oplus$ CHxP CHx_O output enable	CHx_ON=OxCPRE $\oplus$ CHxNP CHx_ON output enable
	1		0	0	CHx_O = CHxP CHx_O output disable.	CHx_ON = CHxNP CHx_ON output disable.
				1	CHx_O = CHxP CHx_O output enable	CHx_ON=OxCPRE $\oplus$ CHxNP CHx_ON output enable
			1	0	CHx_O=OxCPRE $\oplus$ CHxP CHx_O output enable	CHx_ON = CHxNP CHx_ON output enable.
				1	CHx_O=OxCPRE $\oplus$ CHxP CHx_O output enable	CHx_ON=OxCPRE $\oplus$ CHxNP CHx_ON output enable.

## Dead time insertion

The dead time insertion is enabled when both CHxEN and CHxNEN are 1'b1, and set POEN is also necessary. The field named DTCFG defines the dead time delay that can be used for all channels expect for channel 3. The detail about the delay time, refer to the register TIMERx\_CCHP.

The dead time delay insertion ensures that no two complementary signals drive the active state at the same time.

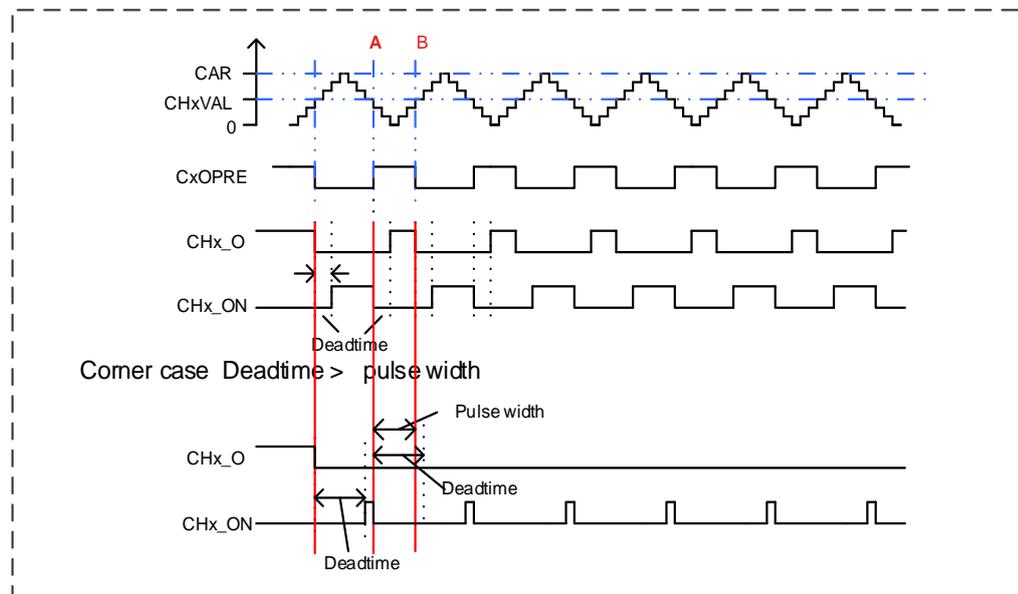
When the channel (x) match (TIMERx counter = CHxVAL) occurs, OxCPRE will be toggled because under PWM0 mode. At point A in the [Figure 15-16. Complementary output with dead-time insertion.](#) CHx\_O signal remains at the low value until the end of the deadtime delay, while CHx\_ON will be cleared at once. Similarly, At point B when counter match (counter = CHxVAL) occurs again, OxCPRE is cleared, CHx\_O signal will be cleared at once, while CHx\_ON signal remains at the low value until the end of the dead time delay.

Sometimes, we can see corner cases about the dead time insertion. For example:

The dead time delay is greater than or equal to the CHx\_O duty cycle, then the CHx\_O signal is always the inactive value. (as show in the [Figure 15-16. Complementary output with dead-time insertion.](#) )

The dead time delay is greater than or equal to the CHx\_ON duty cycle, then the CHx\_ON signal is always the inactive value.

**Figure 15-16. Complementary output with dead-time insertion.**



## Break function

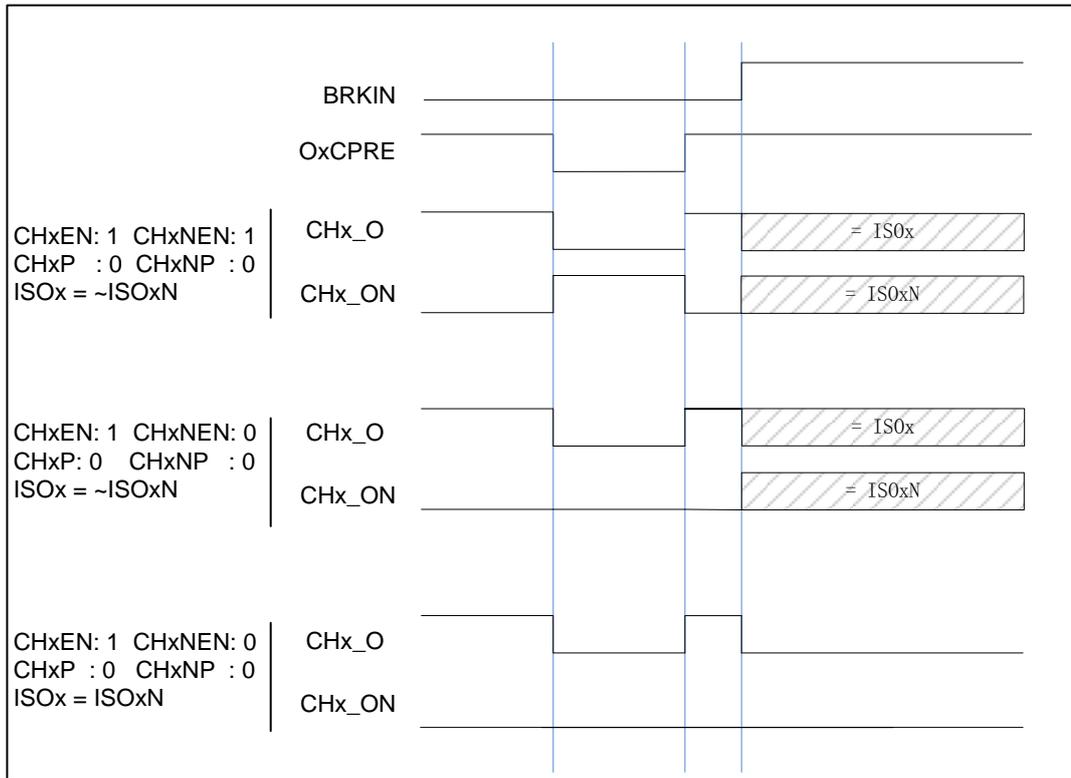
In this function, the output CHx\_O and CHx\_ON are controlled by the POEN, IOS and ROS bits in the TIMERx\_CCHP register, ISOx and ISOxN bits in the TIMERx\_CTL1 register and cannot be set both to active level when break occurs. The break sources are input break pin and HXTAL stuck event by Clock Monitor (CKM) in RCU. The break function enabled by setting the BRKEN bit in the TIMERx\_CCHP register. The break input polarity is setting by the BRKP bit in TIMERx\_CCHP.

When a break occurs, the POEN bit is cleared asynchronously, the output CHx\_O and CHx\_ON are driven with the level programmed in the ISOx bit and ISOxN in the TIMERx\_CTL1 register as soon as POEN is 0. If IOS is 0 then the timer releases the enable output else the enable output remains high. The complementary outputs are first put in reset

state, and then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the ISOx and ISOxN bits after a dead-time.

When a break occurs, the BRKIF bit in the TIMEx\_INTF register is set. If BRKIE is 1, an interrupt generated.

**Figure 15-17. Output behavior in response to a break(The break high active)**



## Quadrature decoder

The quadrature decoder function uses two quadrature inputs CI0 and CI1 derived from the TIMEx\_CH0 and TIMEx\_CH1 pins respectively to interact to generate the counter value. The DIR bit is modified by hardware automatically during each input source transition. The input source can be either CI0 only, CI1 only or both CI0 and CI1, the selection mode by setting the SMC [2:0] to 0x01, 0x02 or 0x03. The mechanism for changing the counter direction is shown in the following table. The quadrature decoder can be regarded as an external clock with a directional selection. This means that the counter counts continuously in the interval between 0 and the counter-reload value. Therefore, users must configure the TIMEx\_CAR register before the counter starts to count.

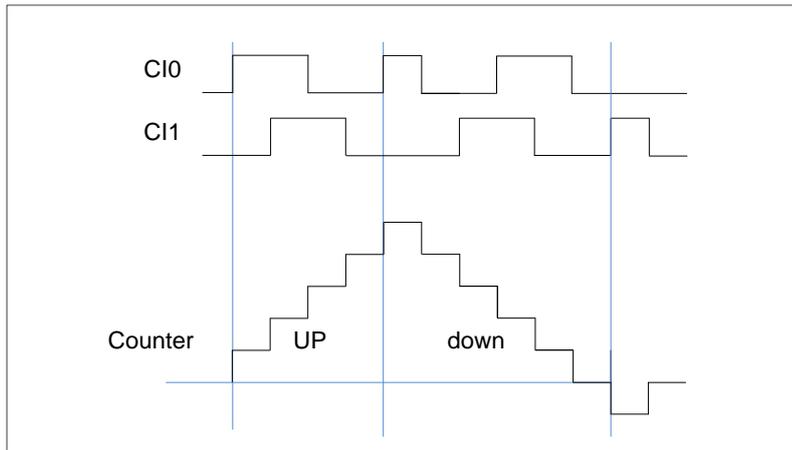
**Table 15-3. Counting direction versus encoder signals**

Counting mode	Level	CI0FE0		CI1FE1	
		Rising	Falling	Rising	Falling
CI0 only counting	CI1FE1=High	Down	Up	-	-
	CI1FE1=Low	Up	Down	-	-

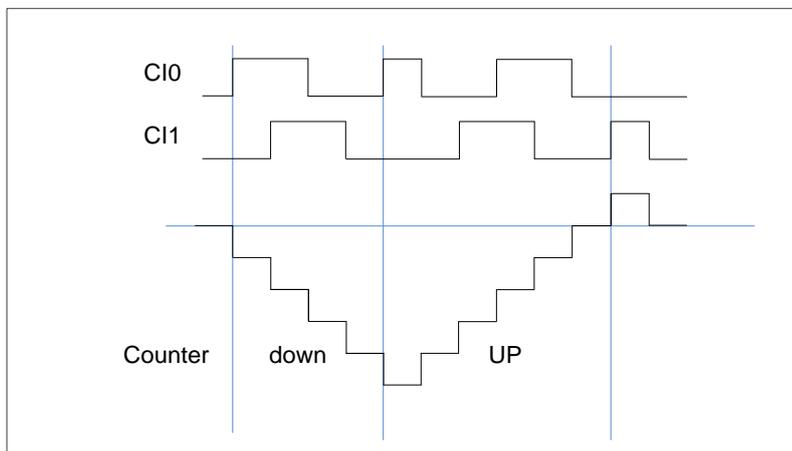
C11 only counting	CI0FE0=High	-	-	Up	Down
	CI0FE0=Low	-	-	Down	Up
C10 and C11 counting	CI1FE1=High	Down	Up	X	X
	CI1FE1=Low	Up	Down	X	X
	CI0FE0=High	X	X	Up	Down
	CI0FE0=Low	X	X	Down	Up

**Note:** "-" means "no counting"; "X" means impossible.

**Figure 15-18. Example of counter operation in encoder interface mode**



**Figure 15-19. Example of encoder interface mode with CI0FE0 polarity inverted**



## Hall sensor function

Hall sensor is generally used to control BLDC Motor; advanced timer can support this function.

**Figure 15-20. Hall sensor is used to BLDC motor** show how to connect. And we can see we need two timers. First TIMER\_in(Advanced/General0 TIMER) should accept three Rotor

Position signals from Motor.

Each of the 3 sensors provides a pulse that applied to an input capture pin, can then be analyzed and both speed and position can be deduced.

By the internal connection such as TRGO-ITIx, TIMER\_in and TIMER\_out can be connected. TIMER\_out will generate PWM signal to control BLDC motor's speed based on the ITRx. Then, the feedback circuit is finished, also you change configuration to fit your request.

About the TIMER\_in, it need have input XOR function, so you can choose from Advanced/GeneralLO TIMER.

And TIMER\_out need have functions of complementary and Dead-time, so only advanced timer can be chosen. Else, based on the timers' internal connection relationship, pair's timers can be selected. For example:

TIMER\_in (TIMER0) -> TIMER\_out (TIMER7 ITI0)

TIMER\_in (TIMER1) -> TIMER\_out (TIMER0 ITI1)

And so on.

After getting appropriate timers combination, and wire connection, we need to configure timers. Some key settings include:

- Enable XOR by setting TI0S, then, each of input signal change will make the CI0 toggle. CH0VAL will record the value of counter at that moment.
- Enable ITIx connected to commutation function directly by setting CCUC and CCSE.
- Configuration PWM parameter based on your request.

**Figure 15-20. Hall sensor is used to BLDC motor**

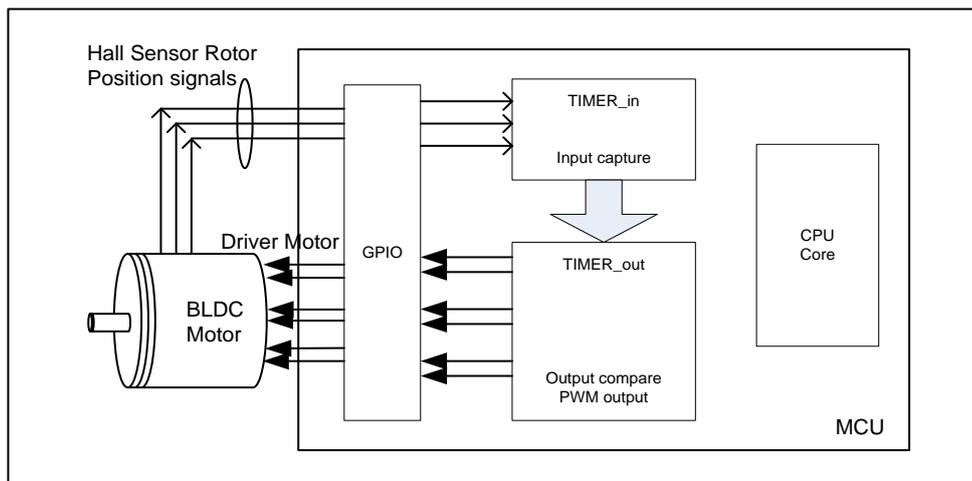
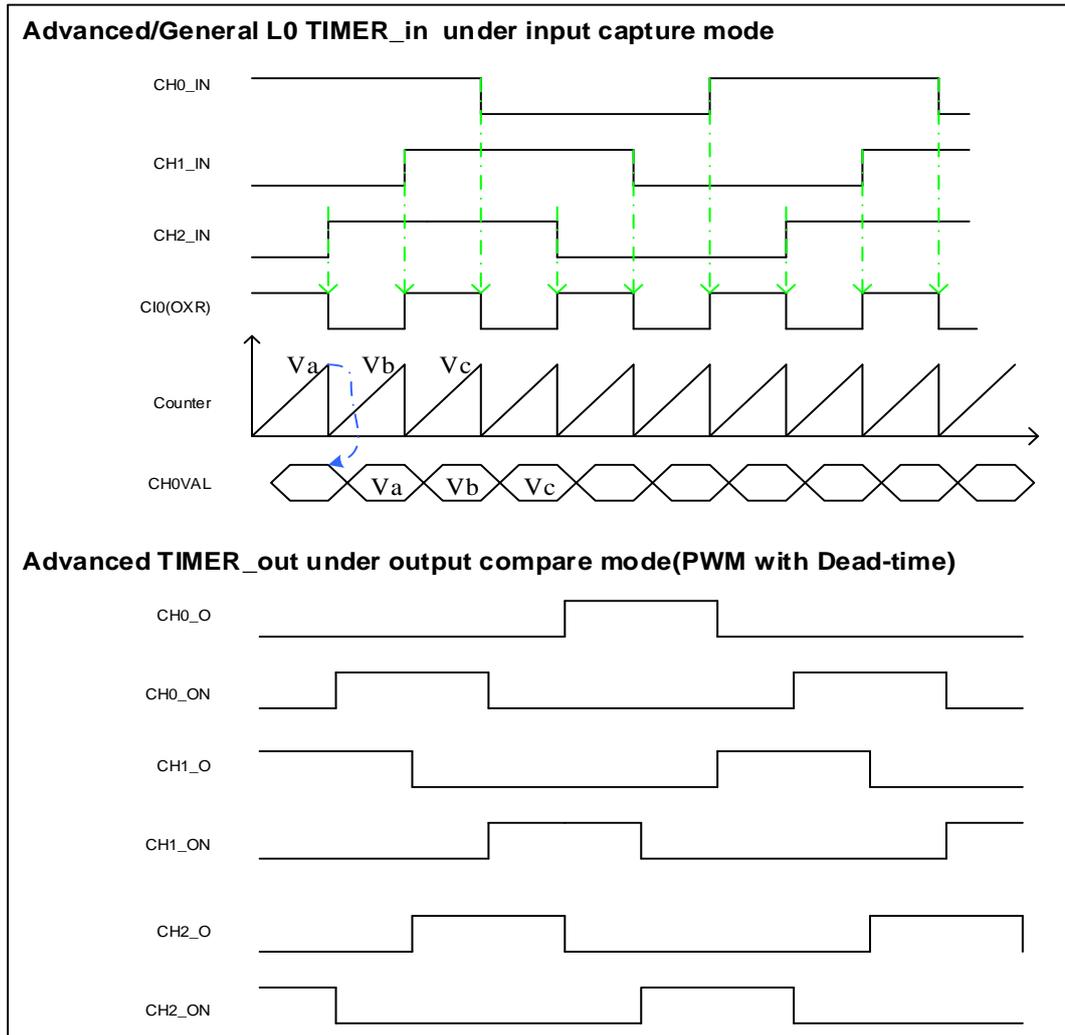


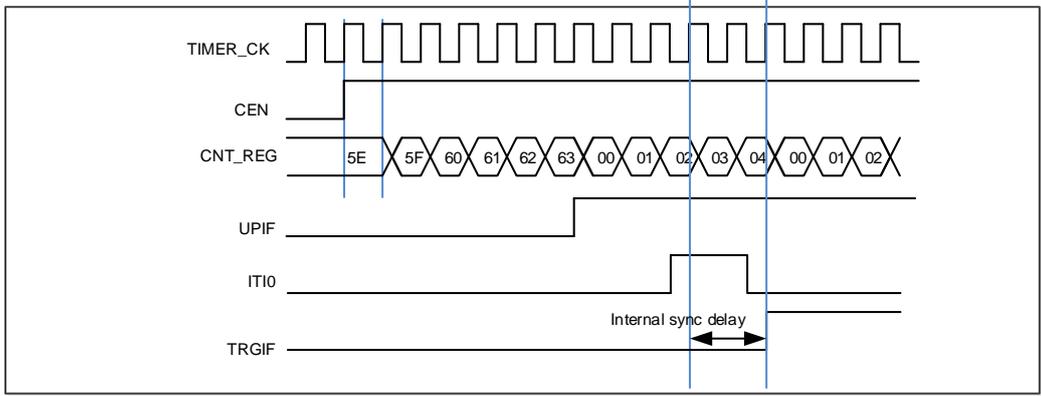
Figure 15-21. Hall sensor timing between two timers

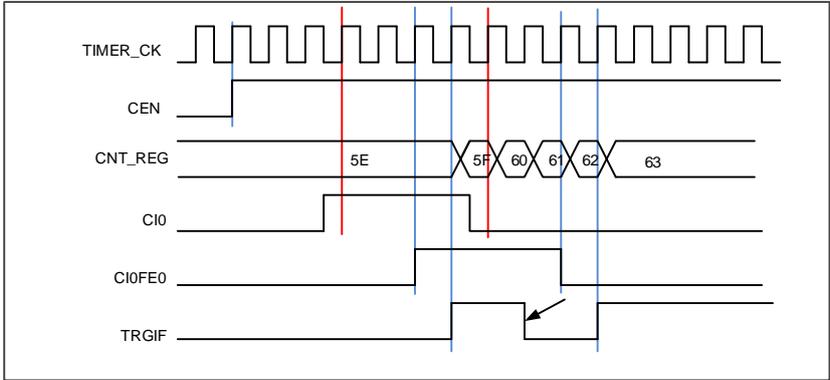
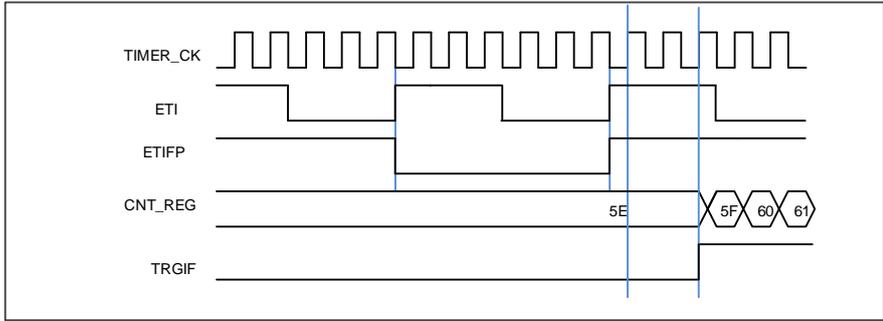


### Slave controller

The TIMERx can be synchronized with a trigger in several modes including the restart mode, the pause mode and the event mode which is selected by the SMC [2:0] in the TIMERx\_SMCFG register. The trigger input of these modes can be selected by the TRGS [2:0] in the TIMERx\_SMCFG register.

**Table 15-4. Slave mode example table**

	Mode Selection	Source Selection	Polarity Selection	Filter and Prescaler
LIST	SMC[2:0]  3'b100 (restart mode)  3'b101 (pause mode)  3'b110 (event mode)	TRGS[2:0] 000: ITI0 001: ITI1 010: ITI2 011: ITI3 100: CI0F_ED 101: CI0FE0 110: CI1FE1 111: ETIFP	If you choose the CI0FE0 or CI1FE1, configure the CHxP and CHxNP for the polarity selection and inversion.  If you choose the ETIF, configure the ETP for polarity selection and inversion.	For the ITIx no filter and prescaler can be used.  For the CIx, configure Filter by CHxCAPFLT, no prescaler can be used.  For the ETIF, configure Filter by ETFC and Prescaler by ETPSC.
Exam1	Restart mode  The counter can be clear and restart when a rising trigger input.	TRGS[2:0]=3'b000  ITI0 is the selection.	-  For ITI0, no polarity selector can be used.	-  For the ITI0, no filter and prescaler can be used.
<p><b>Figure 15-22. Restart mode</b></p> 				
Exam2	Pause mode  The counter can be paused when the trigger input is low.	TRGS[2:0]=3'b101  CI0FE0 is the selection.	TI0S=0.(Non-xor) [CH0NP==0, CH0P==0] no inverted. Capture will be sensitive to the rising edge only.	Filter is bypass in this example.

	Mode Selection	Source Selection	Polarity Selection	Filter and Prescaler
	<p><b>Figure 15-23. Pause mode</b></p> 			
Exam3	<p>Event mode The counter will start to count when a rising trigger input.</p>	<p>TRGS[2:0]=3 'b111 ETIF is the selection.</p>	<p>ETP = 0 no polarity change.</p>	<p>ETPSC = 1, divided by 2. ETFC = 0 , no filter</p>
	<p><b>Figure 15-24. Event mode</b></p> 			

## Single pulse mode

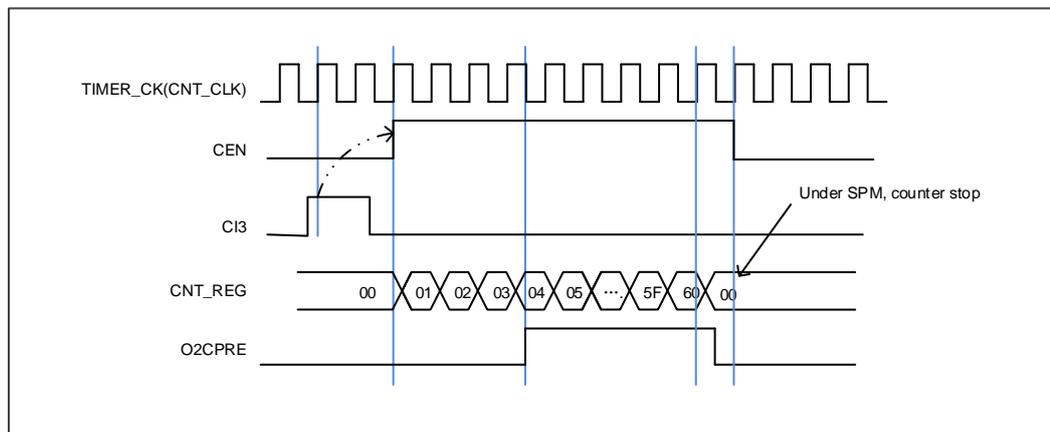
Single pulse mode is opposite to the repetitive mode, which can be enabled by setting SPM in `TIMERx_CTL0`. When you set SPM, the counter will be clear and stop when the next update event automatically. In order to get pulse waveform, you can set the `TIMERx` to PWM mode or compare by `CHxCOMCTL`.

Once the timer is set to operate in the single pulse mode, it is not necessary to set the timer enable bit `CEN` in the `TIMERx_CTL0` register to 1 to enable the counter. The trigger to generate a pulse can be sourced from the trigger signals edge or by setting the `CEN` bit to 1 using software. Setting the `CEN` bit to 1 or a trigger from the trigger signals edge can generate a pulse and then keep the `CEN` bit at a high state until the update event occurs or the `CEN` bit is written to 0 by software. If the `CEN` bit is cleared to 0 using software, the counter will be

stopped and its value held. If the CEN bit is automatically cleared to 0 by a hardware update event, the counter will be reinitialized.

In the single pulse mode, the trigger active edge which sets the CEN bit to 1 will enable the counter. However, there exist several clock delays to perform the comparison result between the counter value and the `TIMERx_CHxCV` value. In order to reduce the delay to a minimum value, the user can set the `CHxCOMFEN` bit in each `TIMERx_CHCTL0/1` register. After a trigger rising occurs in the single pulse mode, the `OxCPRE` signal will immediately be forced to the state which the `OxCPRE` signal will change to, as the compare match event occurs without taking the comparison result into account. The `CHxCOMFEN` bit is available only when the output channel is configured to operate in the PWM0 or PWM1 output mode and the trigger source is derived from the trigger signal.

**Figure 15-25. Single pulse mode `TIMERx_CHxCV = 0x04` `TIMERx_CAR=0x60`**

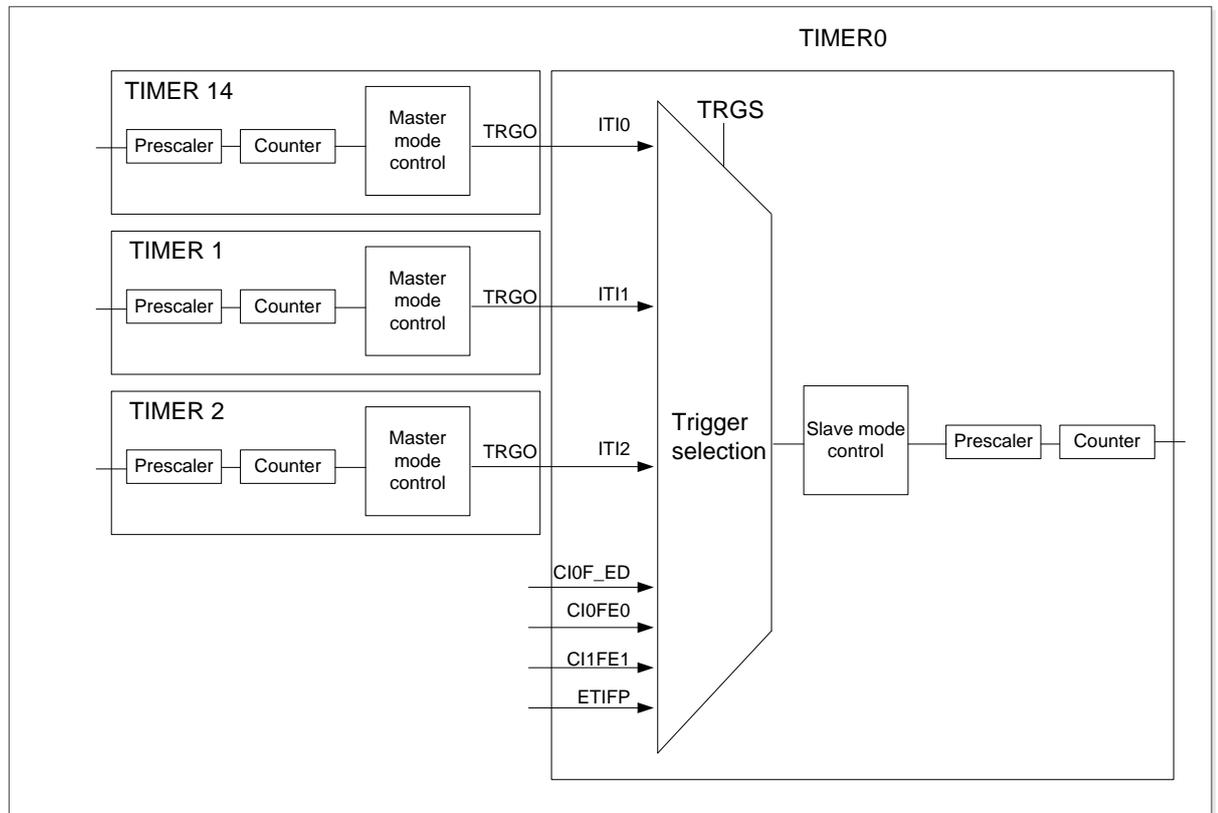


## Timers interconnection

The timers can be internally connected together for timer chaining or synchronization. This can be implemented by configuring one timer to operate in the master mode while configuring another timer to be in the slave mode. The following figures present several examples of trigger selection for the master and slave modes.

[Figure 15-26. \*TIMER0 Master/Slave mode timer example\*](#) shows the timer0 trigger selection when it is configured in slave mode.

**Figure 15-26. TIMER0 Master/Slave mode timer example**



Other interconnection examples:

■ **TIMER1 as prescaler for TIMER0**

We configure TIMER1 as a prescaler for TIMER0. Refer to [Figure 15-26. TIMER0 Master/Slave mode timer example](#) for connections. Do as follow:

1. Configure TIMER1 in master mode and select its Update Event (UPE) as trigger output (MMC=010 in the TIMER1\_CTL1 register). Then TIMER1 drives a periodic signal on each counter overflow.
2. Configure the TIMER1 period (TIMER1\_CAR registers).
3. Select the TIMER0 input trigger source from TIMER1 (TRGS=001 in the TIMER0\_SMCFG register).
4. Configure TIMER0 in external clock mode 1 (SMC=111 in TIMER0\_SMCFG register).
5. Start TIMER0 by writing '1 in the CEN bit (TIMER0\_CTL0 register).
6. Start TIMER1 by writing '1 in the CEN bit (TIMER1\_CTL0 register).

■ **Start TIMER0 with TIMER1's Enable/Update signal**

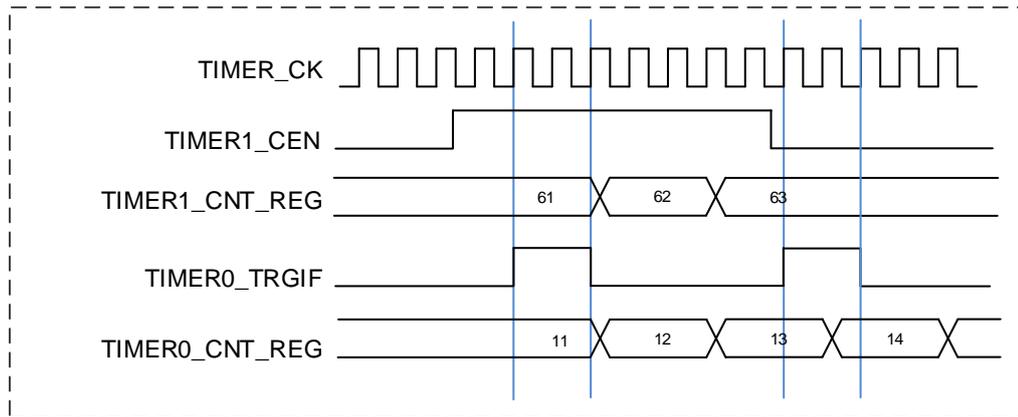
First, we enable TIMER0 with the enable out of TIMER1. Refer to [Figure 15-27. Triggering TIMER0 with Enable of TIMER1](#). TIMER0 starts counting from its current value on the divided internal clock after trigger by TIMER1 enable output.

When TIMER0 receives the trigger signal its CEN bit is automatically set and the counter counts until we disable TIMER0. Both counter clock frequencies are divided by 3 by the

prescaler compared to  $TIMER\_CK$  ( $f_{CNT\_CLK} = f_{TIMER\_CK} / 3$ ). Do as follow:

1. Configure TIMER1 master mode to send its enable signal as trigger output(MMC=001 in the TIMER1\_CTL1 register)
2. Configure TIMER0 to select the input trigger from TIMER1 (TRGS=001 in the TIMER0\_SMCFG register).
3. Configure TIMER0 in event mode (SMC=110 in TIMER0\_SMCFG register).
4. Start TIMER1 by writing 1 in the CEN bit (TIMER1\_CTL0 register).

**Figure 15-27. Triggering TIMER0 with Enable of TIMER1**

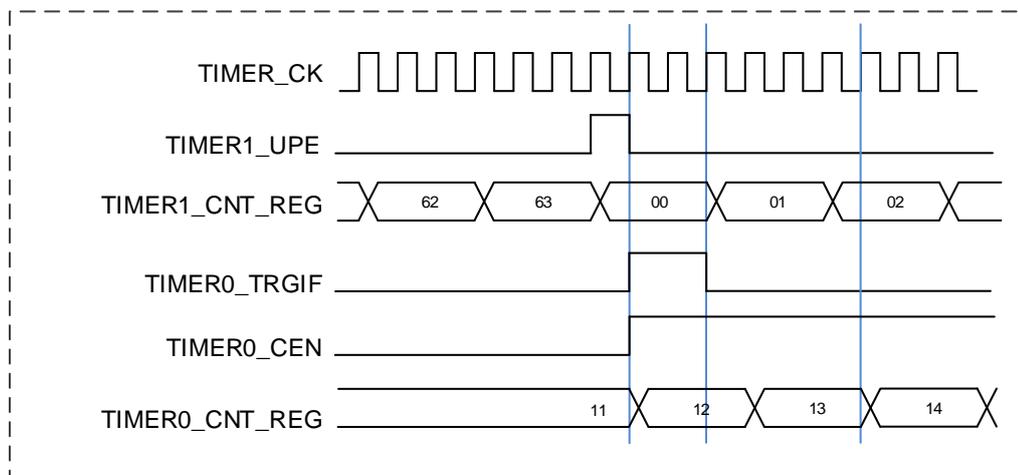


In this example, we also can use update Event as trigger source instead of enable signal.

Refer to [Figure 15-28. Triggering TIMER0 with update signal of TIMER1](#). Do as follow:

1. Configure TIMER1 in master mode and send its Update Event (UPE) as trigger output (MMC=010 in the TIMER1\_CTL1 register).
2. Configure the TIMER1 period (TIMER1\_CAR registers).
3. Configure TIMER0 to get the input trigger from TIMER1 (TRGS=001 in the TIMER0\_SMCFG register).
4. Configure TIMER0 in event mode (SMC=110 in TIMER0\_SMCFG register).
5. Start TIMER1 by writing '1 in the CEN bit (TIMER1\_CTL0 register).

**Figure 15-28. Triggering TIMER0 with update signal of TIMER1**

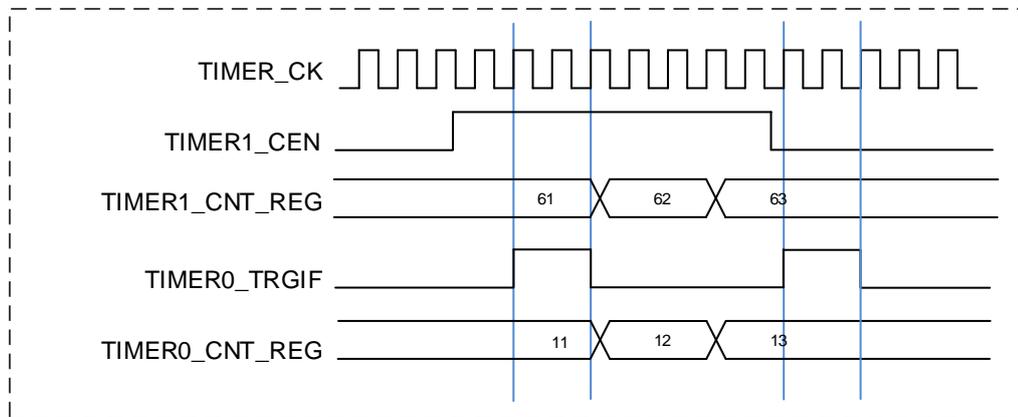


- Enable TIMER0 count with TIMER1's enable/O0CPRE signal

In this example, we control the enable of TIMER0 with the enable output of TIMER1. Refer to [Figure 15-29. Pause TIMER0 with enable of TIMER1](#) TIMER0 counts on the divided internal clock only when TIMER1 is enable. Both counter clock frequencies are divided by 3 by the prescaler compared to TIMER\_CK ( $f_{CNT\_CLK} = f_{TIMER\_CK} / 3$ ). Do as follow:

1. Configure TIMER1 input master mode and Output enable signal as trigger output (MMC=001 in the TIMER1\_CTL1 register).
2. Configure TIMER0 to get the input trigger from TIMER1 (TRGS=001 in the TIMER0\_SMCFG register).
3. Configure TIMER0 in pause mode (SMC=101 in TIMER0\_SMCFG register).
4. Enable TIMER0 by writing '1 in the CEN bit (TIMER0\_CTL0 register)
5. Start TIMER1 by writing '1 in the CEN bit (TIMER1\_CTL0 register).
6. Stop TIMER1 by writing '0 in the CEN bit (TIMER1\_CTL0 register).

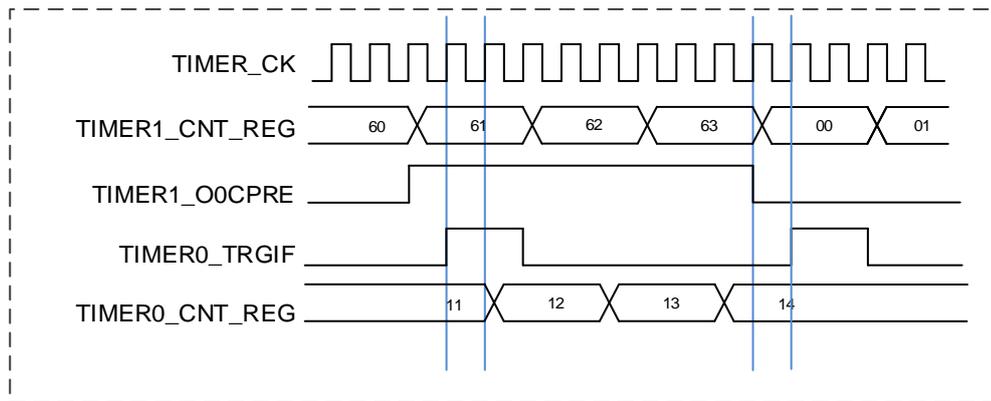
**Figure 15-29. Pause TIMER0 with enable of TIMER1**



In this example, we also can use O0CPRE as trigger source instead of enable signal output. Do as follow:

1. Configure TIMER1 in master mode and Output Compare 1 Reference (O0CPRE) signal as trigger output (MMS=100 in the TIMER1\_CTL1 register).
2. Configure the TIMER1 O0CPRE waveform (TIMER1\_CHCTL0 register).
3. Configure TIMER0 to get the input trigger from TIMER1 (TRGS=001 in the TIMER0\_SMCFG register).
4. Configure TIMER0 in pause mode (SMC=101 in TIMER0\_SMCFG register).
5. Enable TIMER0 by writing '1 in the CEN bit (TIMER0\_CTL0 register).
6. Start TIMER1 by writing '1 in the CEN bit (TIMER1\_CTL0 register).

**Figure 15-30. Pause TIMER0 with O0CPREof TIMER1**



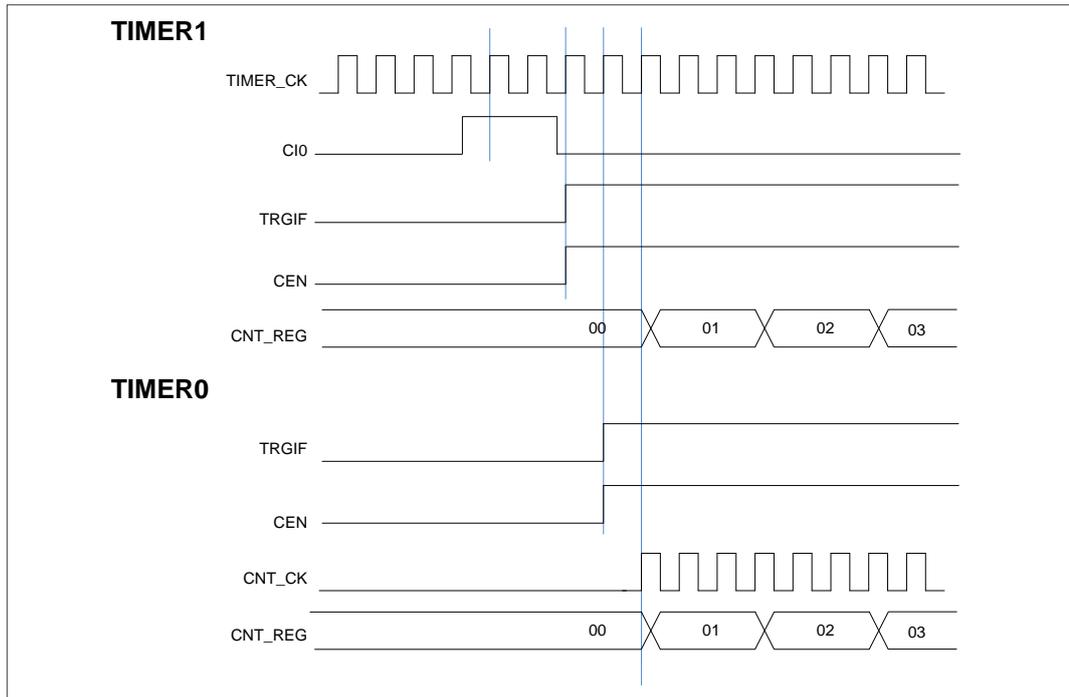
■ Using an external trigger to start 2 timers synchronously

We configure the start of TIMER0 is triggered by the enable of TIMER1, and TIMER1 is triggered by its CI0 input rises edge. To ensure 2 timers start synchronously, TIMER1 must be configured in Master/Slave mode. Do as follow:

1. Configure TIMER1 slave mode to get the input trigger from CI0 (TRGS=100 in the TIMER1\_SMCFG register).
2. Configure TIMER1 in event mode (SMC=110 in the TIMER1\_SMCFG register).
3. Configure the TIMER1 in Master/Slave mode by writing MSM=1 (TIMER1\_SMCFG register).
4. Configure TIMER0 to get the input trigger from TIMER1 (TRGS=001 in the TIMER0\_SMCFG register).
5. Configure TIMER0 in event mode (SMC=110 in the TIMER0\_SMCFG register).

When a rising edge occurs on TIMER1's CI0, two timer counters starts counting synchronously on the internal clock and both TRGIF flags are set.

Figure 15-31. Triggering TIMER0 and TIMER1 with TIMER1's CIO input



### Timer DMA mode

Timer's DMA mode is the function that configures timer's register by DMA module. The relative registers are `TIMERx_DMACFG` and `TIMERx_DMATB`. Of course, you have to enable a DMA request which will be asserted by some internal event. When the interrupt event was asserted, `TIMERx` will send a request to DMA, which is configured to M2P mode and PADDR is `TIMERx_DMATB`, then DMA will access the `TIMERx_DMATB`. In fact, register `TIMERx_DMATB` is only a buffer; timer will map the `TIMERx_DMATB` to an internal register, appointed by the field of `DMATA` in `TIMERx_DMACFG`. If the field of `DMATC` in `TIMERx_DMACFG` is 0(1 transfer), then the timer's DMA request is finished. While if `TIMERx_DMATC` is not 0, such as 3( 4 transfers), then timer will send 3 more requests to DMA, and DMA will access timer's registers `DMATA+0x4`, `DMATA+0x8`, `DMATA+0xc` at the next 3 accesses to `TIMERx_DMATB`. In one word, one time DMA internal interrupt event assert, `DMATC+1` times request will be send by `TIMERx`.

If one more time DMA request event coming, `TIMERx` will repeat the process as above.

### Timer debug mode

When the Cortex™-M3 halted, and the `TIMERx_HOLD` configuration bit in `DBG_CTL0` register set to 1, the `TIMERx` counter stops.

### 15.1.5. TIMERx registers(x=0)

#### Control register 0 (TIMERx\_CTL0)

Address offset: 0x00

Reset value: 0x0000

This register can be accessed by half-word(16-bit) or word(32-bit)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CKDIV[1:0]	ARSE	CAM[1:0]	DIR	SPM	UPS	UPDIS	CEN		
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
15:10	Reserved	Must be kept at reset value
9:8	CKDIV[1:0]	Clock division The CKDIV bits can be configured by software to specify division ratio between the timer clock (TIMER_CK) and the dead-time and sampling clock (DTS), which is used by the dead-time generators and the digital filters. 00: $f_{DTS}=f_{TIMER\_CK}$ 01: $f_{DTS}= f_{TIMER\_CK} /2$ 10: $f_{DTS}= f_{TIMER\_CK} /4$ 11: Reserved
7	ARSE	Auto-reload shadow enable 0: The shadow register for TIMERx_CAR register is disabled 1: The shadow register for TIMERx_CAR register is enabled
6:5	CAM[1:0]	Counter aligns mode selection 00: No center-aligned mode (edge-aligned mode). The direction of the counter is specified by the DIR bit. 01: Center-aligned and counting down assert mode. The counter counts under center-aligned and channel is configured in output mode (CHxMS=00 in TIMERx_CHCTL0 register). Only when the counter is counting down, compare interrupt flag of channels can be set. 10: Center-aligned and counting up assert mode. The counter counts under center-aligned and channel is configured in output mode (CHxMS=00 in TIMERx_CHCTL0 register). Only when the counter is counting up, compare interrupt flag of channels can be set. 11: Center-aligned and counting up/down assert mode. The counter counts under center-aligned and channel is configured in output mode (CHxMS=00 in TIMERx_CHCTL0 register). Both when the counter is counting up and counting down, compare interrupt flag of channels can be set. After the counter is enabled, cannot be switched from 0x00 to non 0x00.
4	DIR	Direction

		0: Count up 1: Count down This bit is read only when the timer is configured in center-aligned mode or encoder mode.
3	SPM	Single pulse mode. 0: Counter continues after update event. 1: The CEN is cleared by hardware and the counter stops at next update event.
2	UPS	Update source This bit is used to select the update event sources by software. 0: Any of the following events generate an update interrupt or DMA request: <ul style="list-style-type: none"> <li>- The UPG bit is set</li> <li>- The counter generates an overflow or underflow event</li> <li>- The slave mode controller generates an update event.</li> </ul> 1: Only counter overflow/underflow generates an update interrupt or DMA request.
1	UPDIS	Update disable. This bit is used to enable or disable the update event generation. 0: update event enable. The update event is generate and the buffered registers are loaded with their preloaded values when one of the following events occurs: <ul style="list-style-type: none"> <li>- The UPG bit is set</li> <li>- The counter generates an overflow or underflow event</li> <li>- The slave mode controller generates an update event.</li> </ul> 1: update event disable. The buffered registers keep their value, while the counter and the prescaler are reinitialized if the UG bit is set or if the slave mode controller generates a hardware reset event.
0	CEN	Counter enable 0: Counter disable 1: Counter enable The CEN bit must be set by software when timer works in external clock, pause mode and encoder mode. While in event mode, the hardware can set the CEN bit automatically.

## Control register 1 (TIMERx\_CTL1)

Address offset: 0x04

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	ISO3	ISO2N	ISO2	ISO1N	ISO1	ISO0N	ISO0	TI0S	MMC[2:0]			DMAS	CCUC	Reserved	CCSE
	rw	rw	rw	rw	rw	rw	rw	rw		rw		rw	rw		rw

Bits	Fields	Descriptions
15	Reserved	Must be kept at reset value
14	ISO3	Idle state of channel 3 output Refer to ISO0 bit
13	ISO2N	Idle state of channel 2 complementary output Refer to ISO0N bit
12	ISO2	Idle state of channel 2 output Refer to ISO0 bit
11	ISO1N	Idle state of channel 1 complementary output Refer to ISO0N bit
10	ISO1	Idle state of channel 1 output Refer to ISO0 bit
9	ISO0N	Idle state of channel 0 complementary output 0: When POEN bit is reset, CH0_ON is set low. 1: When POEN bit is reset, CH0_ON is set high This bit can be modified only when PROT [1:0] bits in TIMERx_CCHP register is 00.
8	ISO0	Idle state of channel 0 output 0: When POEN bit is reset, CH0_O is set low. 1: When POEN bit is reset, CH0_O is set high The CH0_O output changes after a dead-time if CH0_ON is implemented. This bit can be modified only when PROT [1:0] bits in TIMERx_CCHP register is 00.
7	TI0S	Channel 0 trigger input selection 0: The TIMERx_CH0 pin input is selected as channel 0 trigger input. 1: The result of combinational XOR of TIMERx_CH0, CH1 and CH2 pins is selected as channel 0 trigger input.
6:4	MMC[2:0]	Master mode control These bits control the selection of TRGO signal, which is sent in master mode to slave timers for synchronization function. 000: Reset. When the UPG bit in the TIMERx_SWEVG register is set or a reset is generated by the slave mode controller, a TRGO pulse occurs. And in the latter case, the signal on TRGO is delayed compared to the actual reset. 001: Enable. This mode is useful to start several timers at the same time or to control a window in which a slave timer is enabled. In this mode the master mode controller selects the counter enable signal as TRGO. The counter enable signal is set when CEN control bit is set or the trigger input in pause mode is high. There is a delay between the trigger input in pause mode and the TRGO output, except if the master-slave mode is selected. 010: Update. In this mode the master mode controller selects the update event as TRGO.

- 011: Capture/compare pulse. In this mode the master mode controller generates a TRGO pulse when a capture or a compare match occurred in channel0.
- 100: Compare. In this mode the master mode controller selects the O0CPRE signal is used as TRGO
- 101: Compare. In this mode the master mode controller selects the O1CPRE signal is used as TRGO
- 110: Compare. In this mode the master mode controller selects the O2CPRE signal is used as TRGO
- 111: Compare. In this mode the master mode controller selects the O3CPRE signal is used as TRGO
  
- 3        DMAS        DMA request source selection  
 0: DMA request of channel x is sent when capture/compare event occurs.  
 1: DMA request of channel x is sent when update event occurs.
  
- 2        CCUC        Commutation control shadow register update control  
 When the commutation control shadow enable (for CHxEN, CHxNEN and CHxCOMCTL bits) are set (CCSE=1), these shadow registers update are controlled as below:  
 0: The shadow registers update by when CMTG bit is set.  
 1: The shadow registers update by when CMTG bit is set or a rising edge of TRGI occurs.  
 When a channel does not have a complementary output, this bit has no effect.
  
- 1        Reserved        Must be kept at reset value.
  
- 0        CCSE        Commutation control shadow enable  
 0: The shadow registers for CHxEN, CHxNEN and CHxCOMCTL bits are disabled.  
 1: The shadow registers for CHxEN, CHxNEN and CHxCOMCTL bits are enabled.  
 After these bits have been written, they are updated based when commutation event coming.  
 When a channel does not have a complementary output, this bit has no effect.

### Slave mode configuration register (TIMERx\_SMCFG)

Address offset: 0x08

Reset value: 0x0000

This register can be accessed by half-word(16-bit) or word(32-bit)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	SMC1	ETPSC[1:0]			ETF3C[3:0]			MSM	TRGS[2:0]			OCRC	SMC[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
------	--------	--------------

15	ETP	<p>External trigger polarity</p> <p>This bit specifies the polarity of ETI signal</p> <p>0: ETI is active at high level or rising edge.</p> <p>1: ETI is active at low level or falling edge.</p>
14	SMC1	<p>Part of SMC for enable External clock mode1</p> <p>In external clock mode 1, the counter is clocked by any active edge on the ETIF signal.</p> <p>0: External clock mode 1 disabled</p> <p>1: External clock mode 1 enabled.</p> <p>Setting the SMC1 bit has the same effect as selecting external clock mode 0 with TRGI connected to ETIF (SMC=111 and TRGS =111).</p> <p>It is possible to simultaneously use external clock mode 1 with the reset mode, pause mode or event mode. But the TRGS bits must not be 111 in this case.</p> <p>The external clock input will be ETIF if external clock mode 1 and external clock mode 1 are enabled at the same time.</p> <p>Note: External clock mode 0 enable is in this register's SMC bit-filed.</p>
13:12	ETPSC[1:0]	<p>External trigger prescaler</p> <p>The frequency of external trigger signal ETI must not be at higher than 1/4 of TIMERx_CK frequency. When the external trigger signal is a fast clocks, the prescaler can be enabled to reduce ETI frequency.</p> <p>00: Prescaler disable</p> <p>01: ETI frequency will be divided by 2</p> <p>10: ETI frequency will be divided by 4</p> <p>11: ETI frequency will be divided by 8</p>
11:8	ETFC[3:0]	<p>External trigger filter control</p> <p>An event counter is used in the digital filter, in which a transition on the output occurs after N input events. This bit-field specifies the frequency used to sample ETI signal and the length of the digital filter applied to ETI.</p> <p>0000: Filter disable. fSAMP= fDTS, N=1.</p> <p>0001: fSAMP= fTIMER_CK, N=2.</p> <p>0010: fSAMP= fTIMER_CK, N=4.</p> <p>0011: fSAMP= fTIMER_CK, N=8.</p> <p>0100: fSAMP=fDTS/2, N=6.</p> <p>0101: fSAMP=fDTS/2, N=8.</p> <p>0110: fSAMP=fDTS/4, N=6.</p> <p>0111: fSAMP=fDTS/4, N=8.</p> <p>1000: fSAMP=fDTS/8, N=6.</p> <p>1001: fSAMP=fDTS/8, N=8.</p> <p>1010: fSAMP=fDTS/16, N=5.</p> <p>1011: fSAMP=fDTS/16, N=6.</p> <p>1100: fSAMP=fDTS/16, N=8.</p> <p>1101: fSAMP=fDTS/32, N=5.</p>

		1110: fSAMP=fDTS/32, N=6.
		1111: fSAMP=fDTS/32, N=8.
7	MSM	<p>Master-slave mode</p> <p>This bit can be used to synchronize selected timers to begin counting at the same time. The TRGI is used as the start event, and through TRGO, timers are connected together.</p> <p>0: Master-slave mode disable 1: Master-slave mode enable</p>
6:4	TRGS[2:0]	<p>Trigger selection</p> <p>This bit-field specifies which signal is selected as the trigger input, which is used to synchronize the counter.</p> <p>000: Internal trigger input 0 (ITIO) TIMER14 001: Internal trigger input 1 (IT1) TIMER1 010: Internal trigger input 2 (IT2) TIMER2 011: Reserved 100: CI0 edge flag (CI0F_ED) 101: channel 0 input filtered output (CI0FE0) 110: channel 1 input filtered output (CI1FE1) 111: external trigger input filter output (ETIFP)</p> <p>These bits must not be changed when slave mode is enabled.</p>
3	OCRC	<p>OCPRE clear source selection</p> <p>0: OCPRE_CLR_INT is connected to the OCPRE_CLR input 1: OCPRE_CLR_INT is connected to ETIF</p>
2:0	SMC[2:0]	<p>Slave mode control</p> <p>000: Disable mode. The slave mode is disabled; The prescaler is clocked directly by the internal clock (TIMER_CK) when CEN bit is set high.</p> <p>001: Quadrature decoder mode 0. The counter counts on CI1FE1 edge, while the direction depends on CI0FE0 level.</p> <p>010: Quadrature decoder mode 1. The counter counts on CI0FE0 edge, while the direction depends on CI1FE1 level.</p> <p>011: Quadrature decoder mode 2. The counter counts on both CI0FE0 and CI1FE1 edge, while the direction depends on each other.</p> <p>100: Restart Mode. The counter is reinitialized and the shadow registers are updated on the rising edge of the selected trigger input.</p> <p>101: Pause Mode. The trigger input enables the counter clock when it is high and disables the counter when it is low.</p> <p>110: Event Mode. A rising edge of the trigger input enables the counter. The counter cannot be disabled by the slave mode controller.</p> <p>111: External Clock Mode 0. The counter counts on the rising edges of the selected trigger.</p> <p>Because CI0F_ED outputs 1 pulse for each transition on CI0F, and the pause mode checks the level of the trigger signal, when CI0F_ED is selected as the trigger input,</p>

the pause mode must not be used.

## DMA and interrupt enable register (TIMERx\_DMAINTEN)

Address offset: 0x0C

Reset value: 0x0000

This register can be accessed by half-word(16-bit) or word(32-bit)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	TRGDEN	CMTDEN	CH3DEN	CH2DEN	CH1DEN	CH0DEN	UPDEN	BRKIE	TRGIE	CMTIE	CH3IE	CH2IE	CH1IE	CH0IE	UPIE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
15	Reserved	Must be kept at reset value.
14	TRGDEN	Trigger DMA request enable 0: disabled 1: enabled
13	CMTDEN	Commutation DMA request enable 0: disabled 1: enabled
12	CH3DEN	Channel 3 capture/compare DMA request enable 0: disabled 1: enabled
11	CH2DEN	Channel 2 capture/compare DMA request enable 0: disabled 1: enabled
10	CH1DEN	Channel 1 capture/compare DMA request enable 0: disabled 1: enabled
9	CH0DEN	Channel 0 capture/compare DMA request enable 0: disabled 1: enabled
8	UPDEN	Update DMA request enable 0: disabled 1: enabled
7	BRKIE	Break interrupt enable 0: disabled 1: enabled
6	TRGIE	Trigger interrupt enable

		0: disabled 1: enabled
5	CMTIE	commutation interrupt enable 0: disabled 1: enabled
4	CH3IE	Channel 3 capture/compare interrupt enable 0: disabled 1: enabled
3	CH2IE	Channel 2 capture/compare interrupt enable 0: disabled 1: enabled
2	CH1IE	Channel 1 capture/compare interrupt enable 0: disabled 1: enabled
1	CH0IE	Channel 0 capture/compare interrupt enable 0: disabled 1: enabled
0	UPIE	Update interrupt enable 0: disabled 1: enabled

## Interrupt flag register (TIMERx\_INTF)

Address offset: 0x10

Reset value: 0x0000

This register can be accessed by half-word(16-bit) or word(32-bit)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		CH3OF	CH2OF	CH1OF	CH0OF	Reserved.	BRKIF	TRGIF	CMTIF	CH3IF	CH2IF	CH1IF	CH0IF	UPIF	
		rc_w0	rc_w0	rc_w0	rc_w0	.	rc_w0								

Bits	Fields	Descriptions
15:13	Reserved	Must be kept at reset value.
12	CH3OF	Channel 3 over capture flag Refer to CH0OF description
11	CH2OF	Channel 2 over capture flag Refer to CH0OF description
10	CH1OF	Channel 1 over capture flag

		Refer to CH0OF description
9	CH0OF	<p>Channel 0 over capture flag</p> <p>When channel 0 is configured in input mode, this flag is set by hardware when a capture event occurs while CH0IF flag has already been set. This flag is cleared by software.</p> <p>0: No over capture interrupt occurred 1: Over capture interrupt occurred</p>
8	Reserved	Must be kept at reset value.
7	BRKIF	<p>Break interrupt flag</p> <p>This flag is set by hardware when the break input goes active, and cleared by software if the break input is not active.</p> <p>0: No active level break has been detected. 1: An active level has been detected.</p>
6	TRGIF	<p>Trigger interrupt flag</p> <p>This flag is set by hardware on trigger event and cleared by software. When the slave mode controller is enabled in all modes but pause mode, an active edge on trigger input generates a trigger event. When the slave mode controller is enabled in pause mode both edges on trigger input generates a trigger event.</p> <p>0: No trigger event occurred. 1: Trigger interrupt occurred.</p>
5	CMTIF	<p>Channel commutation interrupt flag</p> <p>This flag is set by hardware when channel's commutation event occurs, and cleared by software</p> <p>0: No channel commutation interrupt occurred 1: Channel commutation interrupt occurred</p>
4	CH3IF	<p>Channel 3 's capture/compare interrupt flag</p> <p>Refer to CH0IF description</p>
3	CH2IF	<p>Channel 2 's capture/compare interrupt flag</p> <p>Refer to CH0IF description</p>
2	CH1IF	<p>Channel 1 's capture/compare interrupt flag</p> <p>Refer to CH0IF description</p>
1	CH0IF	<p>Channel 0 's capture/compare interrupt flag</p> <p>This flag is set by hardware and cleared by software. When channel 0 is in input mode, this flag is set when a capture event occurs. When channel 0 is in output mode, this flag is set when a compare event occurs.</p> <p>0: No Channel 0 interrupt occurred 1: Channel 0 interrupt occurred</p>
0	UPIF	<p>Update interrupt flag</p> <p>This bit is set by hardware on an update event and cleared by software.</p>

0: No update interrupt occurred

1: Update interrupt occurred

### Software event generation register (TIMERx\_SWEVG)

Address offset: 0x14

Reset value: 0x0000

This register can be accessed by half-word(16-bit) or word(32-bit)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								BRKG	TRGG	CMTG	CH3G	CH2G	CH1G	CH0G	UPG
								w	w	w	w	w	w	w	w

Bits	Fields	Descriptions
15:8	Reserved	Must be kept at reset value.
7	BRKG	Break event generation This bit is set by software and cleared by hardware automatically. When this bit is set, the POEN bit is cleared and BRKIF flag is set, related interrupt or DMA transfer can occur if enabled. 0: No generate a break event 1: Generate a break event
6	TRGG	Trigger event generation This bit is set by software and cleared by hardware automatically. When this bit is set, the TRGIF flag in TIMERx_INTF register is set, related interrupt or DMA transfer can occur if enabled. 0: No generate a trigger event 1: Generate a trigger event
5	CMTG	Channel commutation event generation This bit is set by software and cleared by hardware automatically. When this bit is set, channel's capture/compare control registers (CHxEN, CHxNEN and CHxCOMCTL bits) are updated based on the value of CCSE (in the TIMERx_CTL1). 0: No affect 1: Generate channel's c/c control update event
4	CH3G	Channel 3's capture or compare event generation Refer to CH0G description
3	CH2G	Channel 2's capture or compare event generation Refer to CH0G description
2	CH1G	Channel 1's capture or compare event generation Refer to CH0G description
1	CH0G	Channel 0's capture or compare event generation

This bit is set by software in order to generate a capture or compare event in channel 0, it is automatically cleared by hardware. When this bit is set, the CH0IF flag is set, the corresponding interrupt or DMA request is sent if enabled. In addition, if channel 1 is configured in input mode, the current value of the counter is captured in TIMERx\_CH0CV register, and the CH0OF flag is set if the CH0IF flag was already high.

0: No generate a channel 1 capture or compare event

1: Generate a channel 1 capture or compare event

0 UPG

Update event generation

This bit can be set by software, and cleared by hardware automatically. When this bit is set, the counter is cleared if the center-aligned or up counting mode is selected, else (down counting) it takes the auto-reload value. The prescaler counter is cleared at the same time.

0: No generate an update event

1: Generate an update event

### Channel control register 0 (TIMERx\_CHCTL0)

Address offset: 0x18

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH1COM CEN	CH1COMCTL[2:0]			CH1COM SEN	CH1COM FEN	CH1MS[1:0]		CH0COM CEN	CH0COMCTL[2:0]			CH0COM SEN	CH0COM FEN	CH0MS[1:0]	
CH1CAPFLT[3:0]				CH1CAPPSC[1:0]			CH0CAPFLT[3:0]			CH0CAPPSC[1:0]					
rw				rw			rw			rw			rw		

#### Output compare mode:

Bits	Fields	Descriptions
15	CH1COMCEN	Channel 1 output compare clear enable Refer to CH0COMCEN description
14:12	CH1COMCTL[2:0]	Channel 1 compare output control Refer to CH0COMCTL description
11	CH1COMSEN	Channel 1 output compare shadow enable Refer to CH0COMSEN description
10	CH1COMFEN	Channel 1 output compare fast enable Refer to CH0COMSEN description
9:8	CH1MS[1:0]	Channel 1 mode selection This bit-field specifies the direction of the channel and the input signal selection. This bit-field is writable only when the channel is not active. (CH1EN bit in TIMERx_CHCTL2 register is reset).

		00: Channel 1 is configured as output
		01: Channel 1 is configured as input, IS1 is connected to CI1FE1
		10: Channel 1 is configured as input, IS1 is connected to CI0FE1
		11: Channel 1 is configured as input, IS1 is connected to ITS. This mode is working only if an internal trigger input is selected through TRGS bits in TIMERx_SMCFG register.
7	CH0COMCEN	<p>Channel 0 output compare clear enable.</p> <p>When this bit is set, the O0CPRE signal is cleared when High level is detected on ETIF input.</p> <p>0: Channel 0 output compare clear disable</p> <p>1: Channel 0 output compare clear enable</p>
6:4	CH0COMCTL[2:0]	<p>Channel 0 compare output control</p> <p>This bit-field controls the behavior of the output reference signal O0CPRE which drives CH0_O and CH0_ON. O0CPRE is active high, while CH0_O and CH0_ON active level depends on CH0P and CH0NP bits.</p> <p>000: Frozen. The O0CPRE signal keeps stable, independent of the comparison between the register TIMERx_CH0CV and the counter TIMERx_CNT.</p> <p>001: Set the channel output. O0CPRE signal is forced high when the counter matches the output compare register TIMERx_CH0CV.</p> <p>010: Clear the channel output. O0CPRE signal is forced low when the counter matches the output compare register TIMERx_CH0CV.</p> <p>011: Toggle on match. O0CPRE toggles when the counter matches the output compare register TIMERx_CH0CV.</p> <p>100: Force low. O0CPRE is forced low level.</p> <p>101: Force high. O0CPRE is forced high level.</p> <p>110: PWM mode0. When counting up, O0CPRE is active as long as the counter is smaller than TIMERx_CH0CV else inactive. When counting down, O0CPRE is inactive as long as the counter is larger than TIMERx_CH0CV else active.</p> <p>111: PWM mode1. When counting up, O0CPRE is inactive as long as the counter is smaller than TIMERx_CH0CV else active. When counting down, O0CPRE is active as long as the counter is larger than TIMERx_CH0CV else inactive.</p> <p>When configured in PWM mode, the O0CPRE level changes only when the output compare mode switches from “frozen” mode to “PWM” mode or when the result of the comparison changes.</p> <p>This bit cannot be modified when PROT [1:0] bit-field in TIMERx_CCHP register is 11 and CH0MS bit-field is 00(COMPARE MODE).</p>
3	CH0COMSEN	<p>Channel 0 compare output shadow enable</p> <p>When this bit is set, the shadow register of TIMERx_CH0CV register, which updates at each update event, will be enabled.</p> <p>0: Channel 0 output compare shadow disable</p> <p>1: Channel 0 output compare shadow enable</p> <p>The PWM mode can be used without validating the shadow register only in single</p>

		pulse mode (SPM bit in TIMERx_CTL0 register is set).
		This bit cannot be modified when PROT [1:0] bit-field in TIMERx_CCHP register is 11 and CH0MS bit-field is 00.
2	CH0COMFEN	<p>Channel 0 output compare fast enable</p> <p>When this bit is set, the effect of an event on the trigger in input on the capture/compare output will be accelerated if the channel is configured in PWM0 or PWM1 mode. The output channel will treat an active edge on the trigger input as a compare match, and CH0_O is set to the compare level independently from the result of the comparison.</p> <p>0: Channel 0 output quickly compare disable. The minimum delay from an edge on the trigger input to activate CH0_O output is 5 clock cycles.</p> <p>1: Channel 0 output quickly compare enable. The minimum delay from an edge on the trigger input to activate CH0_O output is 3 clock cycles.</p>
1:0	CH0MS[1:0]	<p>Channel 0 I/O mode selection</p> <p>This bit-field specifies the work mode of the channel and the input signal selection. This bit-field is writable only when the channel is not active. (CH0EN bit in TIMERx_CHCTL2 register is reset).</p> <p>00: Channel 0 is configured as output</p> <p>01: Channel 0 is configured as input, IS0 is connected to CI0FE0</p> <p>10: Channel 0 is configured as input, IS0 is connected to CI1FE0</p> <p>11: Channel 0 is configured as input, IS0 is connected to ITS, This mode is working only if an internal trigger input is selected through TRGS bits in TIMERx_SMCFG register.</p>

### Input capture mode:

Bits	Fields	Descriptions
15:12	CH1CAPFLT[3:0]	Channel 1 input capture filter control Refer to CH0CAPFLT description
11:10	CH1CAPPSC[1:0]	Channel 1 input capture prescaler Refer to CH0CAPPSC description
9:8	CH1MS[1:0]	Channel 1 mode selection Same as Output compare mode
7:4	CH0CAPFLT[3:0]	<p>Channel 0 input capture filter control</p> <p>An event counter is used in the digital filter, in which a transition on the output occurs after N input events. This bit-field specifies the frequency used to sample CI0 input signal and the length of the digital filter applied to CI0.</p> <p>0000: Filter disabled, <math>f_{SAMP}=f_{DTS}</math>, N=1</p> <p>0001: <math>f_{SAMP}=f_{TIMER\_CK}</math>, N=2</p> <p>0010: <math>f_{SAMP}=f_{TIMER\_CK}</math>, N=4</p> <p>0011: <math>f_{SAMP}=f_{TIMER\_CK}</math>, N=8</p> <p>0100: <math>f_{SAMP}=f_{DTS}/2</math>, N=6</p>

		0101: $f_{SAMP}=f_{DTS}/2$ , N=8
		0110: $f_{SAMP}=f_{DTS}/4$ , N=6
		0111: $f_{SAMP}=f_{DTS}/4$ , N=8
		1000: $f_{SAMP}=f_{DTS}/8$ , N=6
		1001: $f_{SAMP}=f_{DTS}/8$ , N=8
		1010: $f_{SAMP}=f_{DTS}/16$ , N=5
		1011: $f_{SAMP}=f_{DTS}/16$ , N=6
		1100: $f_{SAMP}=f_{DTS}/16$ , N=8
		1101: $f_{SAMP}=f_{DTS}/32$ , N=5
		1110: $f_{SAMP}=f_{DTS}/32$ , N=6
		1111: $f_{SAMP}=f_{DTS}/32$ , N=8
3:2	CH0CAPPSC[1:0]	Channel 0 input capture prescaler This bit-field specifies the factor of the prescaler on channel 0 input. The prescaler is reset when CH0EN bit in TIMERx_CHCTL2 register is clear. 00: Prescaler disable, capture is done on each channel input edge 01: Capture is done every 2 channel input edges 10: Capture is done every 4 channel input edges 11: Capture is done every 8 channel input edges
1:0	CH0MS[1:0]	Channel 0 mode selection Same as Output compare mode

### Channel control register 1 (TIMERx\_CHCTL1)

Address offset: 0x1C

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH3COM CEN	CH3COMCTL[2:0]		CH3COM SEN	CH3COM FEN	CH3MS[1:0]		CH2COM CEN	CH2COMCTL[2:0]		CH2COM SEN	CH2COM FEN	CH2MS[1:0]			
CH3CAPFLT[3:0]			CH3CAPPSC[1:0]					CH2CAPFLT[3:0]		CH2CAPPSC[1:0]					
rw			rw		rw			rw		rw		rw			

#### Output compare mode:

Bits	Fields	Descriptions
15	CH3COMCEN	Channel 3 output compare clear enable Refer to CH0COMCEN description
14:12	CH3COMCTL[2:0]	Channel 3 compare output control Refer to CH0COMCTL description
11	CH3COMSEN	Channel 3 output compare shadow enable Refer to CH0COMSEN description
10	CH3COMFEN	Channel 3 output compare fast enable

Refer to CH0COMSEN description

9:8	CH3MS[1:0]	<p>Channel 3 mode selection</p> <p>This bit-field specifies the direction of the channel and the input signal selection. This bit-field is writable only when the channel is not active. (CH3EN bit in TIMERx_CHCTL2 register is reset).</p> <p>00: Channel 3 is configured as output</p> <p>01: Channel 3 is configured as input, IS3 is connected to CI3FE3</p> <p>10: Channel 3 is configured as input, IS3 is connected to CI2FE3</p> <p>11: Channel 3 is configured as input, IS3 is connected to ITS, This mode is working only if an internal trigger input is selected through TRGS bits in TIMERx_SMCFG register.</p>
7	CH2COMCEN	<p>Channel 2 output compare clear enable.</p> <p>When this bit is set, the O2CPRE signal is cleared when High level is detected on ETIF input.</p> <p>0: Channel 2 output compare clear disable</p> <p>1: Channel 2 output compare clear enable</p>
6:4	CH2COMCTL[2:0]	<p>Channel 2 compare output control</p> <p>This bit-field controls the behavior of the output reference signal O2CPRE which drives CH2_O and CH2_ON. O2CPRE is active high, while CH2_O and CH2_ON active level depends on CH2P and CH2NP bits.</p> <p>000: Frozen. The O2CPRE signal keeps stable, independent of the comparison between the output compare register TIMERx_CH2CV and the counter TIMERx_CNT.</p> <p>001: Set high on match. O2CPRE signal is forced high when the counter matches the output compare register TIMERx_CH2CV.</p> <p>010: Set low on match. O2CPRE signal is forced low when the counter matches the output compare register TIMERx_CH2CV.</p> <p>011: Toggle on match. O2CPRE toggles when the counter matches the output compare register TIMERx_CH2CV.</p> <p>100: Force low. O2CPRE is forced low level.</p> <p>101: Force high. O2CPRE is forced high level.</p> <p>110: PWM mode0. When counting up, O0CPRE is active as long as the counter is smaller than TIMERx_CH0CV else inactive. When counting down, O0CPRE is inactive as long as the counter is larger than TIMERx_CH0CV else active.</p> <p>111: PWM mode1. When counting up, O0CPRE is inactive as long as the counter is smaller than TIMERx_CH0CV else active. When counting down, O0CPRE is active as long as the counter is larger than TIMERx_CH0CV else inactive.</p> <p>When configured in PWM mode, the O2CPRE level changes only when the output compare mode switches from “frozen” mode to “PWM” mode or when the result of the comparison changes.</p> <p>This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 and CH2MS bit-filed is 00(COMPARE MODE).</p>

3	CH2COMSEN	<p>Channel 2 compare output shadow enable</p> <p>When this bit is set, the shadow register of TIMERx_CH2CV register, which updates at each update event will be enabled.</p> <p>0: Channel 2 output compare shadow disable 1: Channel 2 output compare shadow enable</p> <p>The PWM mode can be used without validating the shadow register only in single pulse mode (SPM bit in TIMERx_CTL0 register is set).</p> <p>This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 and CH0MS bit-filed is 00.</p>
2	CH2COMFEN	<p>Channel 2 output compare fast enable</p> <p>When this bit is set, the effect of an event on the trigger in input on the capture/compare output will be accelerated if the channel is configured in PWM1 or PWM2 mode. The output channel will treat an active edge on the trigger input as a compare match, and CH2_O is set to the compare level independently from the result of the comparison.</p> <p>0: Channel 2 output quickly compare disable. The minimum delay from an edge on the trigger input to activate CH2_O output is 5 clock cycles. 1: Channel 2 output quickly compare enable. The minimum delay from an edge on the trigger input to activate CH2_O output is 3 clock cycles.</p>
1:0	CH2MS[1:0]	<p>Channel 2 I/O mode selection</p> <p>This bit-field specifies the work mode of the channel and the input signal selection. This bit-field is writable only when the channel is not active. (CH2EN bit in TIMERx_CHCTL2 register is reset).).</p> <p>00: Channel 2 is configured as output 01: Channel 2 is configured as input, IS2 is connected to CI2FE2 10: Channel 2 is configured as input, IS2 is connected to CI3FE2 11: Channel 2 is configured as input, IS2 is connected to ITS. This mode is working only if an internal trigger input is selected through TRGS bits in TIMERx_SMCFG register.</p>

**Input capture mode:**

Bits	Fields	Descriptions
15:12	CH3CAPFLT[3:0]	Channel 3 input capture filter control Refer to CH0CAPFLT description
11:10	CH3CAPPSC[1:0]	Channel 3 input capture prescaler Refer to CH0CAPPSC description
9:8	CH3MS[1:0]	Channel 3 mode selection Same as Output compare mode
7:4	CH2CAPFLT[3:0]	Channel 2 input capture filter control An event counter is used in the digital filter, in which a transition on the output occurs after N input events. This bit-field specifies the frequency used to sample CI2 input

signal and the length of the digital filter applied to CI2.

0000: Filter disable,  $f_{SAMP}=f_{DTS}$ ,  $N=1$

0001:  $f_{SAMP}=f_{TIMER\_CK}$ ,  $N=2$

0010:  $f_{SAMP}=f_{TIMER\_CK}$ ,  $N=4$

0011:  $f_{SAMP}=f_{TIMER\_CK}$ ,  $N=8$

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

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

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

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

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

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

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

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

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

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

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

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

- 3:2 CH2CAPPSC[1:0] Channel 2 input capture prescaler  
 This bit-field specifies the factor of the prescaler on channel 2 input. The prescaler is reset when CH2EN bit in TIMERx\_CHCTL2 register is clear.  
 00: Prescaler disable, capture is done on each channel input edge  
 01: Capture is done every 2 channel input edges  
 10: Capture is done every 4 channel input edges  
 11: Capture is done every 8 channel input edges
- 1:0 CH2MS[1:0] Channel 2 mode selection  
 Same as Output compare mode

### Channel control register 2 (TIMERx\_CHCTL2)

Address offset: 0x20

Reset value: 0x0000

This register can be accessed by half-word(16-bit) or word(32-bit)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	CH3P	CH3EN	CH2NP	CH2NEN	CH2P	CH2EN	CH1NP	CH1NEN	CH1P	CH1EN	CH0NP	CH0NEN	CH0P	CH0EN	
	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	

Bits	Fields	Descriptions
15:14	Reserved	Must be kept at reset value
13	CH3P	Channel 3 capture/compare function polarity Refer to CH0P description
12	CH3EN	Channel 3 capture/compare function enable

		Refer to CH0EN description
11	CH2NP	Channel 2 complementary output polarity Refer to CH0NP description
10	CH2NEN	Channel 2 complementary output enable Refer to CH0NEN description
9	CH2P	Channel 2 capture/compare function polarity Refer to CH0P description
8	CH2EN	Channel 2 capture/compare function enable Refer to CH0EN description
7	CH1NP	Channel 1 complementary output polarity Refer to CH0NP description
6	CH1NEN	Channel 1 complementary output enable Refer to CH0NEN description
5	CH1P	Channel 1 capture/compare function polarity Refer to CH0P description
4	CH1EN	Channel 1 capture/compare function enable Refer to CH0EN description
3	CH0NP	Channel 0 complementary output polarity When channel 0 is configured in output mode, this bit specifies the complementary output signal polarity. 0: Channel 0 active high 1: Channel 0 active low When channel 0 is configured in input mode, In conjunction with CH0P, this bit is used to define the polarity of CI0. This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 or 10.
2	CH0NEN	Channel 0 complementary output enable When channel 0 is configured in output mode, setting this bit enables the complementary output in channel0. 0: Channel 0 complementary output disabled 1: Channel 0 complementary output enabled
1	CH0P	Channel 0 capture/compare function polarity When channel 0 is configured in output mode, this bit specifies the output signal polarity. 0: Channel 0 active high 1: Channel 0 active low When channel 0 is configured in input mode, this bit specifies the CI0 signal polarity. [CH0NP, CH0P] will select the active trigger or capture polarity for CI0FE0 or CI1FE0.

[CH0NP==0, CH0P==0]: ClxFE0's rising edge is the active signal for capture or trigger operation in slave mode. And ClxFE0 will not be inverted.

[CH0NP==0, CH0P==1]: ClxFE0's falling edge is the active signal for capture or trigger operation in slave mode. And ClxFE0 will be inverted.

[CH0NP==1, CH0P==0]: Reserved.

[CH0NP==1, CH0P==1]: ClxFE0's falling and rising edge are both the active signal for capture or trigger operation in slave mode. And ClxFE0 will be not inverted.

This bit cannot be modified when PROT [1:0] bit-field in TIMERx\_CCHP register is 11 or 10.

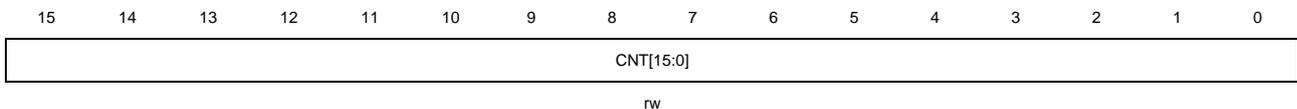
0	CH0EN	<p>Channel 0 capture/compare function enable</p> <p>When channel 0 is configured in output mode, setting this bit enables CH0_O signal in active state. When channel 0 is configured in input mode, setting this bit enables the capture event in channel0.</p> <p>0: Channel 0 disabled 1: Channel 0 enabled</p>
---	-------	---

### Counter register (TIMERx\_CNT)

Address offset: 0x24

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



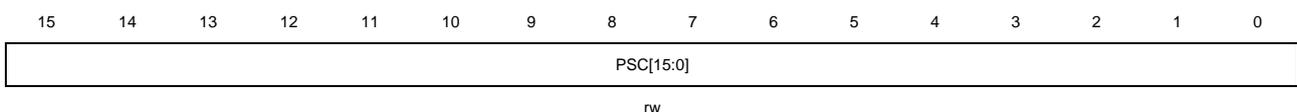
Bits	Fields	Descriptions
15:0	CNT[15:0]	This bit-field indicates the current counter value. Writing to this bit-field can change the value of the counter.

### Prescaler register (TIMERx\_PSC)

Address offset: 0x28

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



Bits	Fields	Descriptions
15:0	PSC[15:0]	<p>Prescaler value of the counter clock</p> <p>The PSC clock is divided by (PSC+1) to generate the counter clock. The value of this</p>

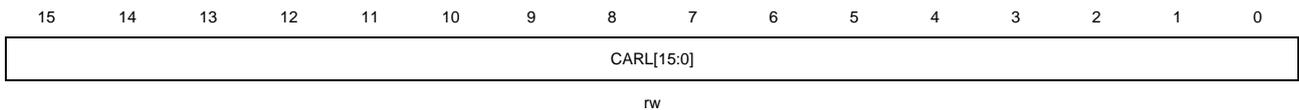
bit-filed will be loaded to the corresponding shadow register at every update event.

## Counter auto reload register (TIMERx\_CAR)

Address offset: 0x2C

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



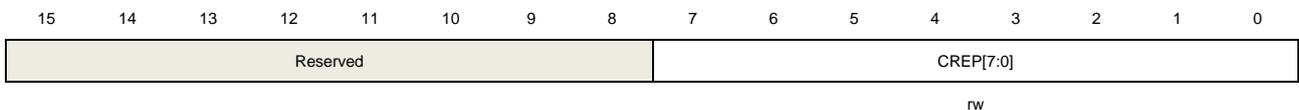
Bits	Fields	Descriptions
15:0	CARL[15:0]	Counter auto reload value This bit-filed specifies the auto reload value of the counter.

## Counter repetition register (TIMERx\_CREP)

Address offset: 0x30

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



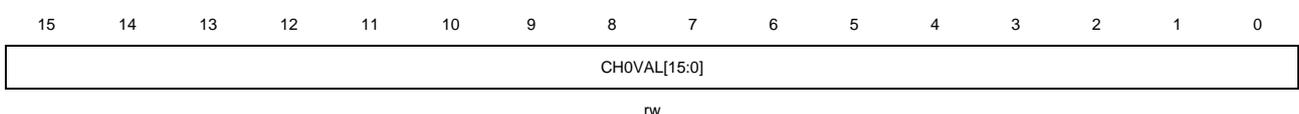
Bits	Fields	Descriptions
15:8	Reserved	Must be kept at reset value.
7:0	CREP[7:0]	Counter repetition value This bit-filed specifies the update event generation rate. Each time the repetition counter counting down to zero, an update event is generated. The update rate of the shadow registers is also affected by this bit-filed when these shadow registers are enabled.

## Channel 0 capture/compare value register (TIMERx\_CH0CV)

Address offset: 0x34

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



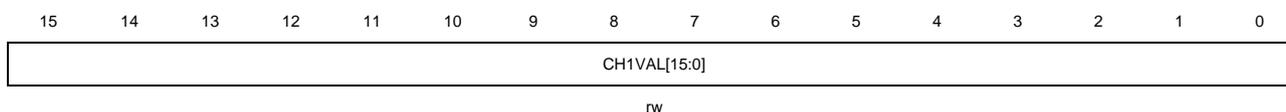
Bits	Fields	Descriptions
15:0	CH0VAL[15:0]	<p>Capture or compare value of channel0</p> <p>When channel 0 is configured in input mode, this bit-filed indicates the counter value corresponding to the last capture event. And this bit-filed is read-only.</p> <p>When channel 0 is configured in output mode, this bit-filed contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates every update event.</p>

### Channel 1 capture/compare value register (TIMERx\_CH1CV)

Address offset: 0x38

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



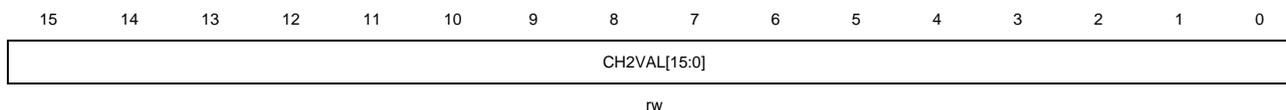
Bits	Fields	Descriptions
15:0	CH1VAL[15:0]	<p>Capture or compare value of channel1</p> <p>When channel 1 is configured in input mode, this bit-filed indicates the counter value corresponding to the last capture event. And this bit-filed is read-only.</p> <p>When channel 1 is configured in output mode, this bit-filed contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates every update event.</p>

### Channel 2 capture/compare value register (TIMERx\_CH2CV)

Address offset: 0x3C

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



Bits	Fields	Descriptions
15:0	CH2VAL[15:0]	<p>Capture or compare value of channel 2</p> <p>When channel 2 is configured in input mode, this bit-filed indicates the counter value corresponding to the last capture event. And this bit-filed is read-only.</p> <p>When channel 2 is configured in output mode, this bit-filed contains value to be compared to the counter. When the corresponding shadow register is enabled, the</p>

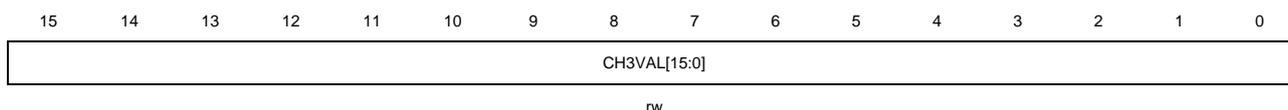
shadow register updates every update event.

## Channel 3 capture/compare value register (TIMERx\_CH3CV)

Address offset: 0x40

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



rw

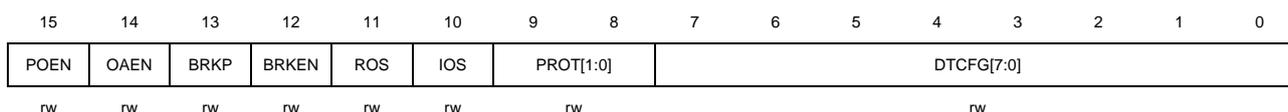
Bits	Fields	Descriptions
15:0	CH3VAL[15:0]	<p>Capture or compare value of channel 3</p> <p>When channel3 is configured in input mode, this bit-filed indicates the counter value corresponding to the last capture event. And this bit-filed is read-only.</p> <p>When channel 3 is configured in output mode, this bit-filed contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates every update event.</p>

## Channel complementary protection register (TIMERx\_CCHP)

Address offset: 0x44

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



rw

rw

rw

rw

rw

rw

rw

rw

Bits	Fields	Descriptions
15	POEN	<p>Primary output enable</p> <p>This bit s set by software or automatically by hardware depending on the OAEN bit. It is cleared asynchronously by hardware as soon as the break input is active. When one of channels is configured in output mode, setting this bit enables the channel outputs (CHx_O and CHx_ON) if the corresponding enable bits (CHxEN, CHxNEN in TIMERx_CHCTL2 register) have been set.</p> <p>0: Channel outputs are disabled or forced to idle state. 1: Channel outputs are enabled.</p>
14	OAEN	<p>Output automatic enable</p> <p>This bit specifies whether the POEN bit can be set automatically by hardware.</p> <p>0: POEN can be not set by hardware. 1: POEN can be set by hardware automatically at the next update event, if the break</p>

		input is not active. This bit can be modified only when PROT [1:0] bit-filed in TIMERx_CCHP register is 00.
13	BRKP	<p>Break polarity</p> <p>This bit specifies the polarity of the BRKIN input signal.</p> <p>0: BRKIN input active low 1: BRKIN input active high</p>
12	BRKEN	<p>Break enable</p> <p>This bit can be set to enable the BRKIN and CCS clock failure event inputs.</p> <p>0: Break inputs disabled 1: Break inputs enabled</p> <p>This bit can be modified only when PROT [1:0] bit-filed in TIMERx_CCHP register is 00.</p>
11	ROS	<p>Run mode off-state configure</p> <p>When POEN bit is set, this bit specifies the output state for the channels which has a complementary output and has been configured in output mode.</p> <p>0: When POEN bit is set, the channel output signals (CHx_O/CHx_ON) are disabled. 1: When POEN bit is set, the channel output signals (CHx_O/CHx_ON) are enabled, with relationship to CHxEN/CHxNEN bits in TIMERx_CHCTL2 register.</p> <p>This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 10 or 11.</p>
10	IOS	<p>Idle mode off-state configure</p> <p>When POEN bit is reset, this bit specifies the output state for the channels which has been configured in output mode.</p> <p>0: When POEN bit is reset, the channel output signals (CHx_O/CHx_ON) are disabled. 1: When POEN bit is reset, he channel output signals (CHx_O/CHx_ON) are enabled, with relationship to CHxEN/CHxNEN bits in TIMERx_CHCTL2 register.</p> <p>This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 10 or 11.</p>
9:8	PROT[1:0]	<p>Complementary register protect control</p> <p>This bit-filed specifies the write protection property of registers.</p> <p>00: protect disable. No write protection. 01: PROT mode 0. The ISOx/ISOxN bits in TIMERx_CTL1 register and the BRKEN/BRKP/OAEN/DTCFG bits in TIMERx_CCHP register are writing protected. 10: PROT mode 1. In addition of the registers in PROT mode 0, the CHxP/CHxNP bits in TIMERx_CHCTL2 register (if related channel is configured in output mode) and the ROS/IOS bits in TIMERx_CCHP register are writing protected. 11: PROT mode 2. In addition of the registers in PROT mode 1, the CHxCOMCTL/CHxCOMSEN bits in TIMERx_CHCTL0/1 registers (if the related channel is configured in output) are writing protected.</p>

This bit-field can be written only once after the reset. Once the TIMERx\_CCHP register has been written, this bit-field will be writing protected.

7:0      DTCFG[7:0]      Dead time configure

This bit-field controls the value of the dead-time, which is inserted before the output transitions. The relationship between DTCFG value and the duration of dead-time is as follow:

DTCFG [7:5] =3'b0xx: DTvalue =DTCFG [7:0]x t<sub>DT</sub>, t<sub>DT</sub>=t<sub>DTs</sub>.

DTCFG [7:5] =3'b 10x: DTvalue = (64+DTCFG [5:0])x t<sub>DT</sub>, t<sub>DT</sub> =t<sub>DTs</sub>\*2.

DTCFG [7:5] =3'b 110: DTvalue = (32+DTCFG [4:0])x t<sub>DT</sub>, t<sub>DT</sub>=t<sub>DTs</sub>\*8.

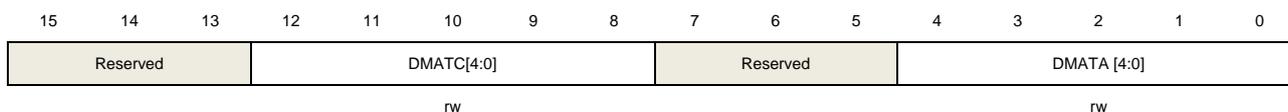
DTCFG [7:5] =3'b 111: DTvalue = (32+DTCFG [4:0])x t<sub>DT</sub>, t<sub>DT</sub> =t<sub>DTs</sub>\*16.

This bit can be modified only when PROT [1:0] bit-filed in TIMERx\_CCHP register is 00.

## DMA configuration register (TIMERx\_DMACFG)

Address offset: 0x48  
Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



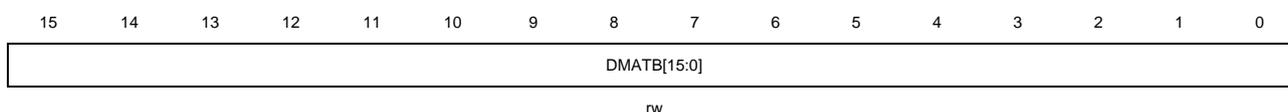
Bits	Fields	Descriptions
15:13	Reserved	Must be kept at reset value.
12:8	DMATC [4:0]	DMA transfer count This filed is defined the number of DMA will access(R/W) the register of TIMERx_DMATB
7:5	Reserved	Must be kept at reset value.
4:0	DMATA [4:0]	DMA transfer access start address This filed define the first address for the DMA access the TIMERx_DMATB. When access is done through the TIMERx_DMA address first time, this bit-field specifies the address you just access. And then the second access to the TIMERx_DMATB, you will access the address of start address + 0x4.  5'b0_0000: TIMERx_CTL0 5'b0_0001: TIMERx_CTL1 ... In a word: Start Address = TIMERx_CTL0 + DMATA*4

## DMA transfer buffer register (TIMERx\_DMATB)

Address offset: 0x4C

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



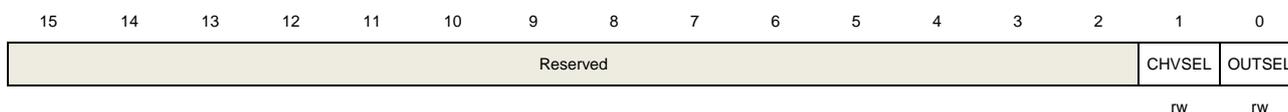
Bits	Fields	Descriptions
15:0	DMATB[15:0]	<p>DMA transfer buffer</p> <p>When a read or write operation is assigned to this register, the register located at the address range (Start Addr + Transfer Timer* 4) will be accessed.</p> <p>The transfer Timer is calculated by hardware, and ranges from 0 to DMATC.</p>

## Configuration register (TIMERx\_CFG) of GD32F170xx and GD32F190xx devices

Address offset: 0xFC

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



Bits	Fields	Descriptions
15:2	Reserved	Must be kept at reset value
1	CHVSEL	<p>Write CHxVAL register selection</p> <p>This bit-field set and reset by software.</p> <p>1: If write the CHxVAL register, the write value is same as the CHxVAL value, the write access ignored</p> <p>0: No effect</p>
0	OUTSEL	<p>The output value selection</p> <p>This bit-field set and reset by software</p> <p>1: If POEN and IOS is 0, the output disabled</p> <p>0: No effect</p>

## 15.2. General level0 timer (TIMERx, x=1, 2)

### 15.2.1. Overview

The general level0 timer module (TIMER1, 2) is a four-channel timer that supports input capture, output compare. They can generate PWM signals to control motor or be used for power management applications. The general level0 time reference is a 16-bit or 32-bit counter that can be used as an unsigned counter.

In addition, the general level0 timers can be programmed and be used to count or time external events that drive other timers.

Timers are completely independent, but there may be synchronized to provide a larger timer with their counters incrementing in unison.

### 15.2.2. Characteristics

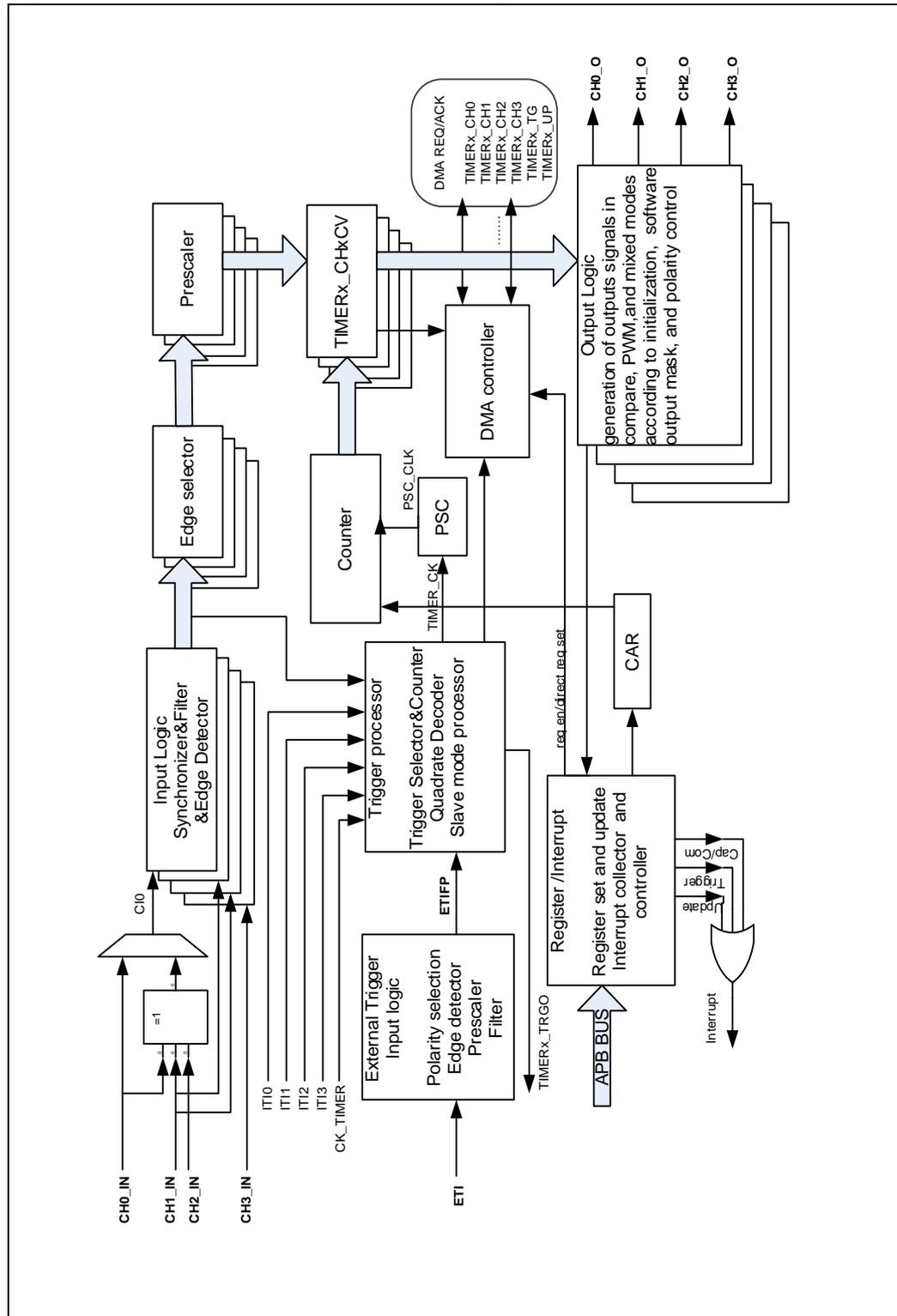
- Total channel num: 4.
- Counter width: 16bit (TIMER2), 32bit (TIMER1).
- Source of count clock is selectable:  
internal clock, internal trigger, external input, external trigger.
- Multiple counter modes: count up, count down, count up/down.
- Quadrature decoder: used to track motion and determine both rotation direction and position.
- Hall sensor: for 3-phase motor control.
- Programmable prescaler: 16 bit. Factor can be changed on the go.
- Each channel is user-configurable:  
Input capture mode, output compare mode, programmable PWM mode, single pulse mode
- Auto-reload function.
- Interrupt output or DMA request on: update, trigger event, and compare/capture event.
- Daisy chaining of timer modules to allow a single timer to initiate multiple timing events.
- Timer synchronization allows selected timers to start counting on the same clock cycle.
- Timer Master/Slave mode controller.

### 15.2.3. Block diagram

[Figure 15-32. General Level 0 timer block diagram](#) provides details on the internal

configuration of the general level0 timer.

Figure 15-32. General Level 0 timer block diagram



### 15.2.4. Function overview

#### Clock selection

The general level0 TIMER has the capability of being clocked by either the CK\_TIMER or an alternate clock source controlled by SMC (TIMERx\_SMCFG bit [2:0]).

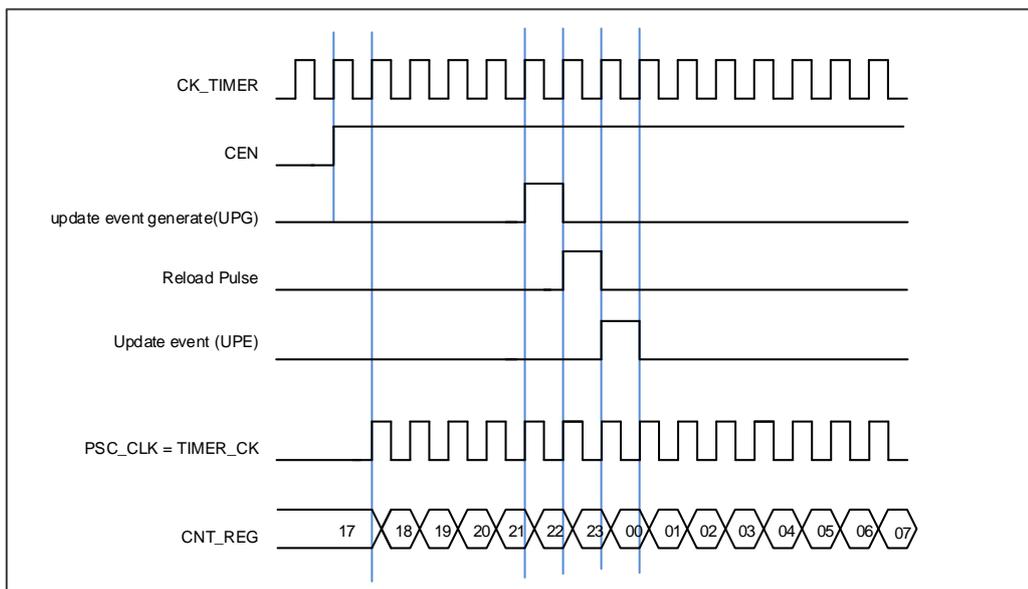
- SMC [2:0] == 3'b000. Internal timer clock CK\_TIMER which is from module RCU.

The default internal clock source is the CK\_TIMER used to drive the counter prescaler when the slave mode is disabled (SMC [2:0] == 3'b000). When the CEN is set, the CK\_TIMER will be divided by PSC value to generate PSC\_CLK.

In this mode, the TIMER\_CK, driven counter's prescaler to count, is equal to CK\_TIMER which is from RCU.

If the slave mode controller is enabled by setting SMC [2:0] in the TIMERx\_SMCFG register to an available value including 0x1, 0x2, 0x3 and 0x7, the prescaler is clocked by other clock sources selected by the TRGS [2:0] in the TIMERx\_SMCFG register and described as follows. When the slave mode selection bits SMC [2:0] are set to 0x4, 0x5 or 0x6, the internal clock TIMER\_CK is the counter prescaler driving clock source.

**Figure 15-33. Normal mode, internal clock divided by 1**



- SMC [2:0] == 3'b111(external clock mode 0). External input pin source

The TIMER\_CK, driven counter's prescaler to count, can be triggered by the event of rising or falling edge on the external pin TIMERx\_CI0/TIMERx\_CI1. This mode can be selected by setting SMC [2:0] to 0x7 and the TRGS [2:0] to 0x4, 0x5 or 0x6.

And, the counter prescaler can also be driven by rising edge on the internal trigger input pin ITI0/1/2/3. This mode can be selected by setting SMC [2:0] to 0x7 and the TRGS [2:0] to 0x0,

0x1, 0x2 or 0x3.

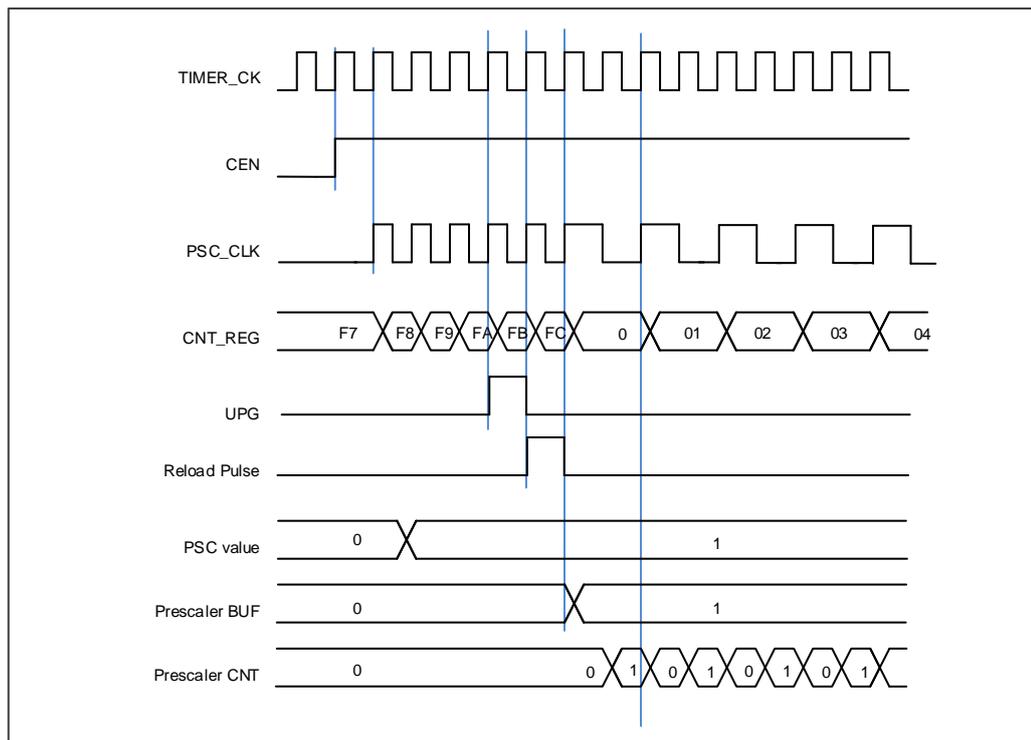
- SMC1== 1'b1(external clock mode 1). External input pin source (ETI)

The TIMER\_CK, driven counter's prescaler to count, can be triggered by the event of rising or falling edge on the external pin ETI. This mode can be selected by setting the SMC1 bit in the TIMEx\_SMCFG register to 1. The other way to select the ETI signal as the clock source is set the SMC [2:0] to 0x7 and the TRGS [2:0] to 0x7 respectively. Note that the ETI signal is derived from the ETI pin sampled by a digital filter. When the clock source is selected to come from the ETI signal, the trigger controller including the edge detection circuitry will generate a clock pulse during each ETI signal rising edge to clock the counter prescaler.

## Prescaler

The prescaler can divide the timer clock (TIMER\_CK) to the counter clock (PSC\_CLK) by any factor between 1 and 65536. It is controlled through prescaler register (TIMEx\_PSC) which can be changed on the go but be taken into account at the next update event.

**Figure 15-34. Counter timing diagram with prescaler division change from 1 to 2**



## Up counting mode

In this mode, the counter counts up continuously from 0 to the counter-reload value, which is defined in the TIMEx\_CAR register, in a count-up direction. Once the counter reaches the counter reload value, the counter restarts to count once again from 0. The update event is generated at each counter overflow. The counting direction bit DIR in the TIMEx\_CTL1 register should be set to 0 for the up counting mode.

When the update event is set by the UPG bit in the TIMEx\_SWEVG register, the counter

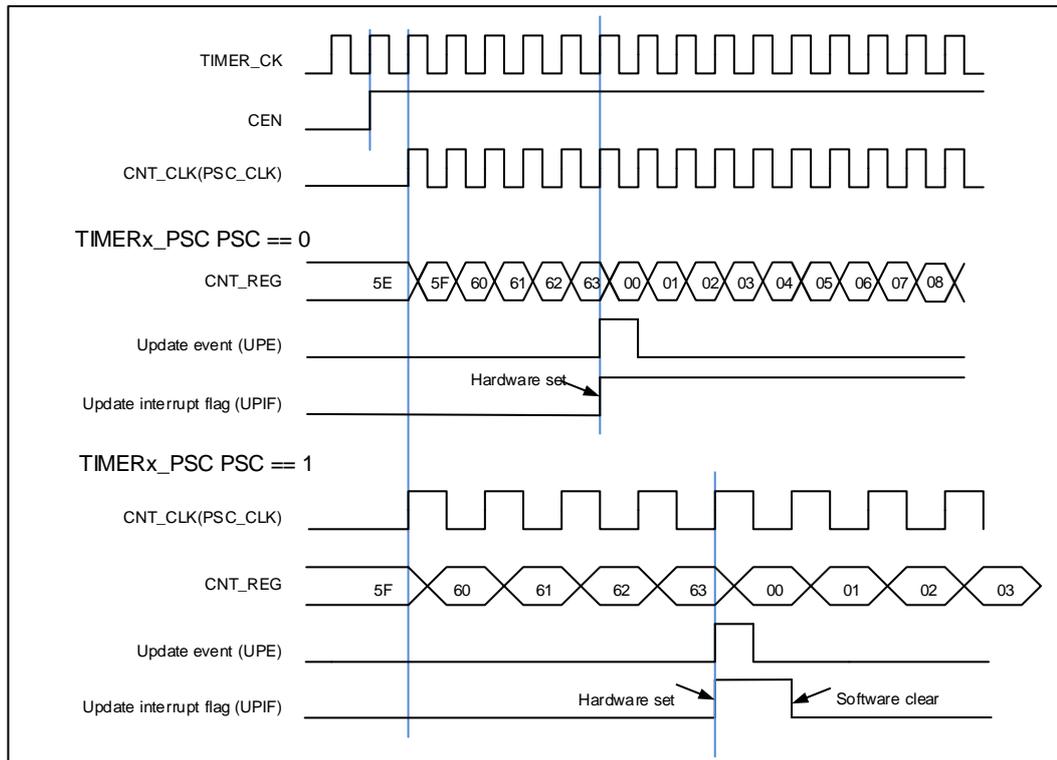
value will be initialized to 0 and generates an update event.

If the UPDIS bit in TIMEx\_CTL0 register is set, the update event is disabled.

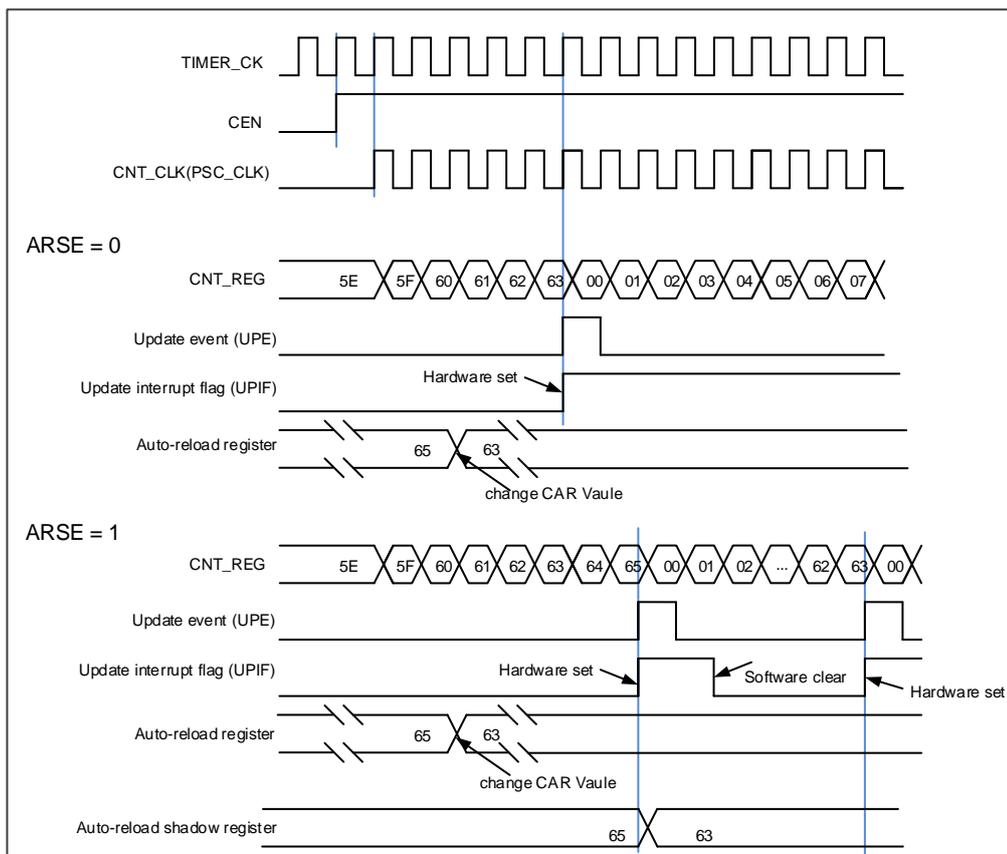
When an update event occurs, all the registers (repetition counter, auto reload register, prescaler register) are updated.

The following figures show some examples of the counter behavior for different clock prescaler factor when TIMEx\_CAR=0x63.

**Figure 15-35. Up-counter timechart, PSC=0/1**



**Figure 15-36. Up-counter timechart, change `TIMERx_CAR` on the go.**



### Down counting mode

In this mode, the counter counts down continuously from the counter-reload value, which is defined in the `TIMERx_CAR` register, to 0 in a count-down direction. Once the counter reaches to 0, the counter restarts to count again from the counter-reload value. If the repetition counter is set, the update event was generated after the number (`TIMERx_CREP+1`) of underflow. Else the update event is generated at each counter underflow. The counting direction bit `DIR` in the `TIMERx_CTL0` register should be set to 1 for the down-counting mode.

When the update event is set by the `UPG` bit in the `TIMERx_SWEVG` register, the counter value will be initialized to the counter-reload value and generates an update event.

If the `UPDIS` bit in `TIMERx_CTL0` register is set, the update event is disabled.

When an update event occurs, all the registers (repetition counter, auto reload register, prescaler register) are updated.

The following figures show some examples of the counter behavior for different clock frequencies when `TIMERx_CAR=0x63`.

Figure 15-37. Down-counter timechart, PSC=0/1

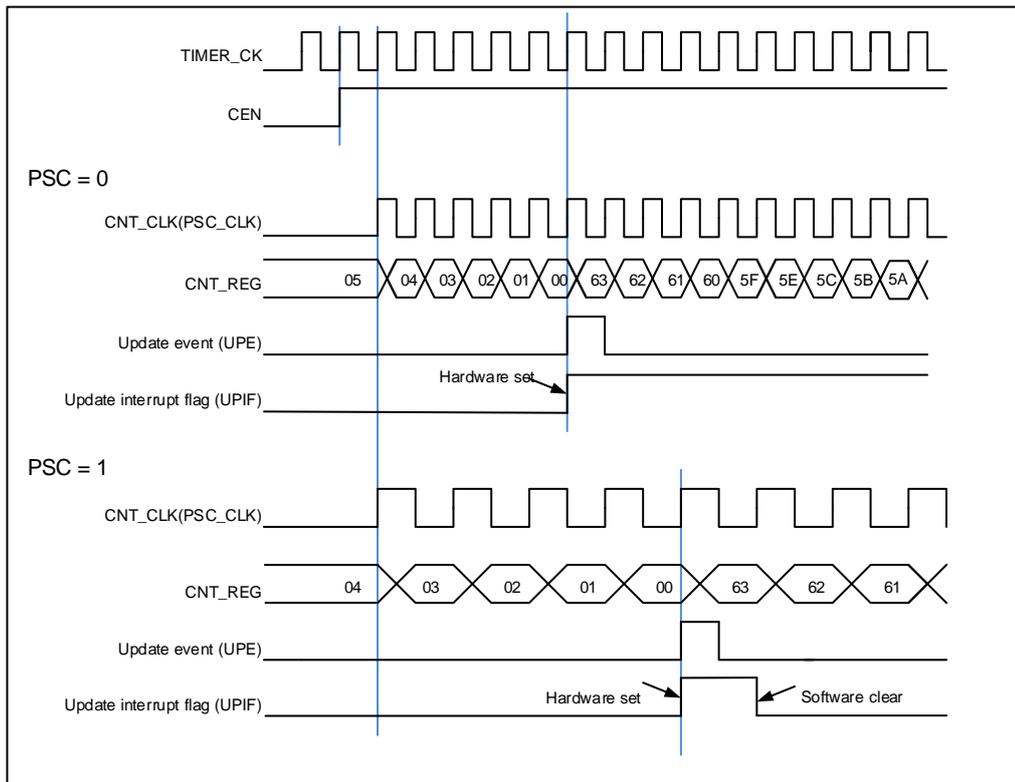
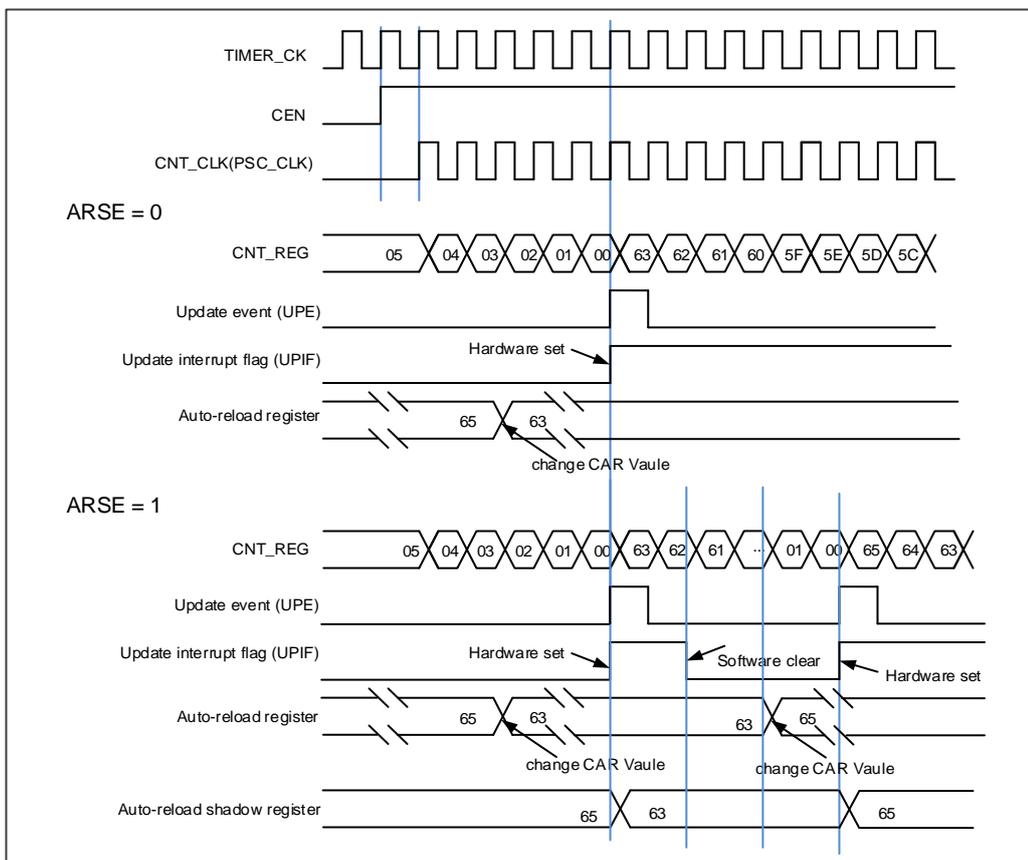


Figure 15-38. Down-counter timechart, change TIMERx\_CAR on the go.



### Center-aligned counting mode

In the center-aligned counting mode, the counter counts up from 0 to the counter-reload value and then counts down to 0 alternatively. The TIMER module generates an overflow event when the counter counts to the counter-reload value subtract 1 in the up-counting mode and generates an underflow event when the counter counts to 1 in the down-counting mode. The counting direction bit DIR in the TIMEx\_CTL0 register is read-only and indicates the counting direction when in the center-aligned mode. The counting direction is updated by hardware automatically.

Setting the UPG bit in the TIMEx\_SWEVG register will initialize the counter value to 0 irrespective of whether the counter is counting up or down in the center-align counting mode and generates an update event.

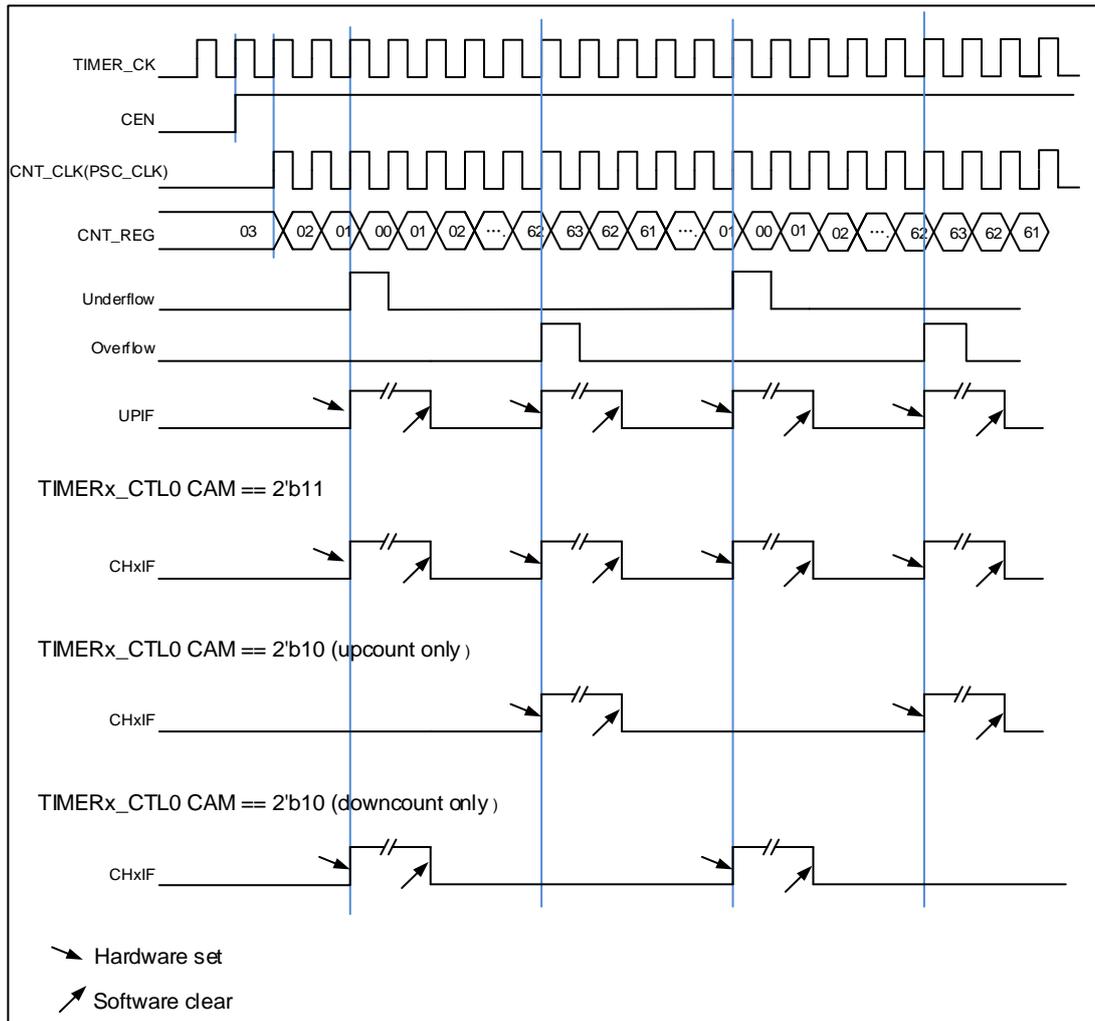
The UPIF bit in the TIMEx\_SWEVG register can be set to 1 when an underflow event at count-down (CAM in TIMEx\_CTL0 is “2'b01”), an overflow event at count-up (CAM in TIMEx\_CTL0 is “2'b10”) or both of them occur (CAM in TIMEx\_CTL0 is “2'b11”).

If the UPDIS bit in the TIMEx\_CTL0 register is set, the update event is disabled.

When an update event occurs, all the registers (repetition counter, autoreload register, prescaler register) are updated.

The following figures show some examples of the counter behavior for different clock frequencies when TIMEx\_CAR=0x63. TIMEx\_PSC=0x0

**Figure 15-39. Center-aligned counter timechart**



## Capture/compare channels

The general level0 TIMER has four independent channels which can be used as capture inputs or compare match outputs. Each channel is built around a channel capture compare register including an input stage, channel controller and an output stage.

### ■ Input capture mode

Capture mode allows the channel to perform measurements such as pulse timing, frequency, period, duty cycle and so on. The input stage consists of a digital filter, a channel polarity selection, edge detection and a channel prescaler. When a selected edge occurs on the channel input, the current value of the counter is captured into the `TIMERx_CHxCV` register, at the same time the `CHxIF` bit is set and the channel interrupt is generated if enabled by `CHxIE = 1`.



**Result:** When you wanted input signal is got, `TIMERx_CHxCV` will be set by counter's value. And `CHxIF` is asserted. If the `CHxIF` is high, the `CHxOF` will be asserted also. The interrupt and DMA request will be asserted based on the your configuration of `CHxIE` and `CHxDEN` in `TIMERx_DMAINTEN`

**Direct generation:** If you want to generate a DMA request or interrupt, you can set `CHxG` by software directly.

The input capture mode can be also used for pulse width measurement from signals on the `TIMERx_CHx` pins. For example, PWM signal connect to `CI0` input. Select channel 0 capture signals to `CI0` by setting `CH0MS` to `2'b01` in the channel control register (`TIMERx_CHCTL0`) and set capture on rising edge. Select channel 1 capture signal to `CI0` by setting `CH1MS` to `2'b10` in the channel control register (`TIMERx_CHCTL0`) and set capture on falling edge. The counter set to restart mode and restart on channel 0 rising edge. Then the `TIMERx_CH0CV` can measure the PWM period and the `TIMERx_CH1CV` can measure the PWM duty.

## ■ Output compare mode

In Output Compare mode, the `TIMERx` can generate timed pulses with programmable position, polarity, duration, and frequency. When the counter matches the value in the `CHxVAL` register of an output compare channel, the channel (n) output can be set, cleared, or toggled based on `CHxCOMCTL`. when the counter reaches the value in the `CHxVAL` register, the `CHxIF` bit is set and the channel (n) interrupt is generated if `CHxIE = 1`. And the DMA request will be assert, if `CHxDEN = 1`.

So the process can be divided to several steps as below:

**Step1:** Clock configuration. Such as clock source, clock prescaler and so on.

**Step2:** Compare mode configuration.

- \* Set the shadow enable mode by `CHxCOMSEN`
- \* Set the output mode (Set/Clear/Toggle) by `CHxCOMCTL`.
- \* Select the active high polarity by `CHxP/CHxNP`
- \* Enable the output by `CHxEN`

**Step3:** Interrupt/DMA-request enables configuration by `CHxIE/ CHxDEN`

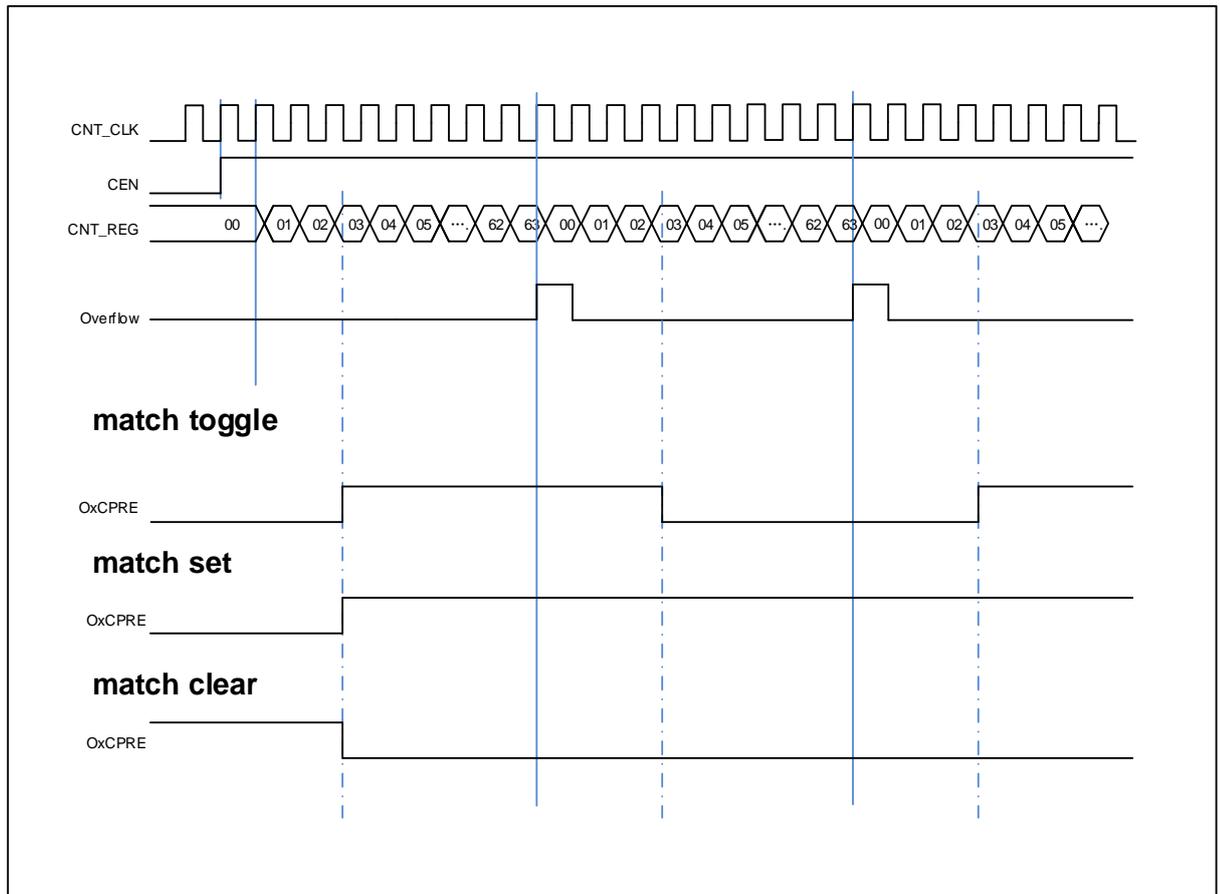
**Step4:** Compare output timing configuration by `TIMERx_CAR` and `TIMERx_CHxCV`.

About the `CHxVAL`, you can change it on the go to meet the waveform you expected.

**Step5:** Start the counter by `CEN`.

The timechart below show the three compare modes toggle/set/clear. `CAR=0x63`, `CHxVAL=0x3`

Figure 15-41. Output-compare under three modes



**PWM mode**

In the output PWM mode (by setting the CHxCOMCTL bits to 3'b110 (PWM mode0) or to 3'b111(PWM mode1), the channel can outputs PWM waveform according to the TIMERx\_CAR registers and TIMERx\_CHxCV registers.

Based on the counter mode, we have can also divide PWM into EAPWM (Edge aligned PWM) and CAPWM (Centre aligned PWM).

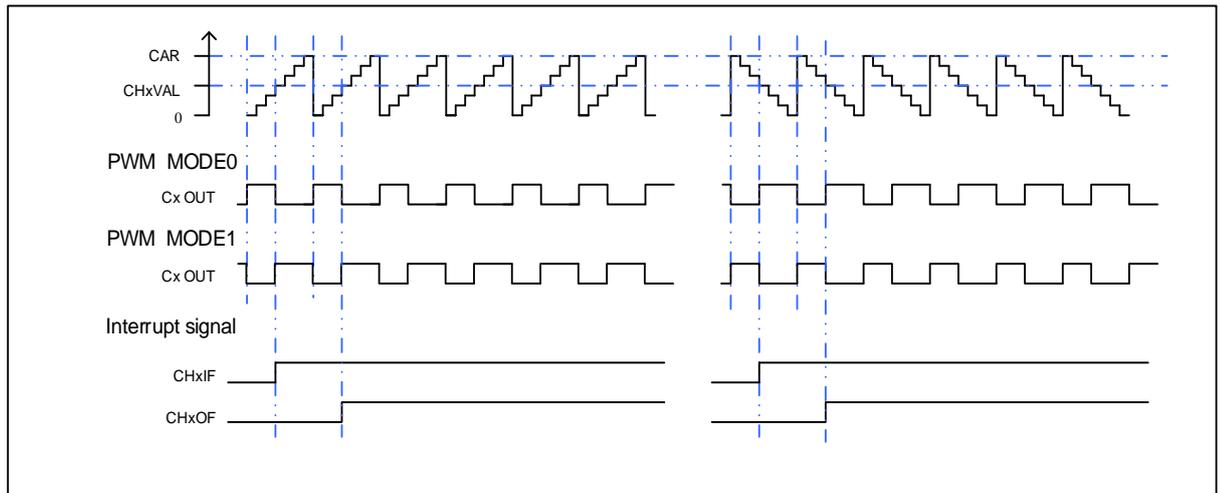
The EAPWM period is determined by TIMERx\_CAR and duty cycle is by TIMERx\_CHxCV. [Figure 15-42. EAPWM timechart](#) shows the EAPWM output and interrupts waveform.

The CAPWM period is determined by 2\*TIMERx\_CAR, and duty cycle is determined by 2\*TIMERx\_CHxCV. [Figure 15-43. CAPWM timechart](#) shows the CAPWM output and interrupts waveform.

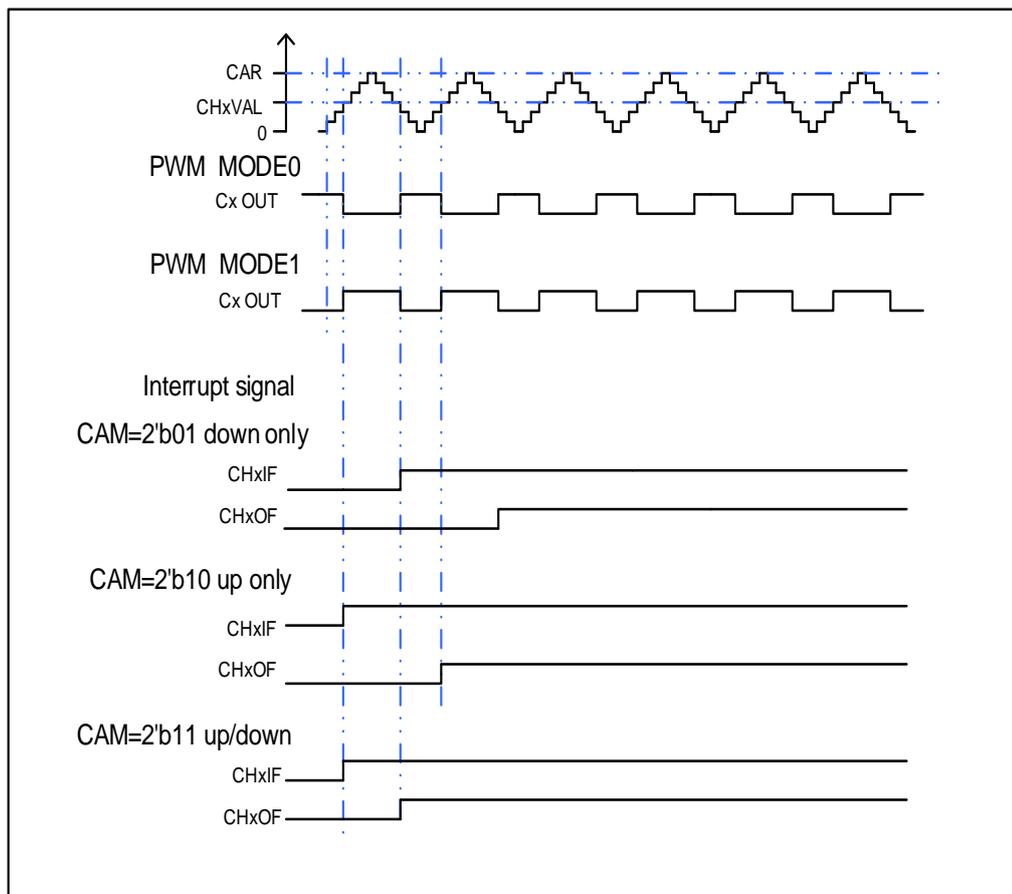
If TIMERx\_CHxCV is greater than TIMERx\_CAR, the output will be always active under PWM mode0 (CHxCOMCTL==3'b110).

And if TIMERx\_CHxCV is equal to zero, the output will be always inactive under PWM mode0 (CHxCOMCTL==3'b110).

**Figure 15-42. EAPWM timechart**



**Figure 15-43. CAPWM timechart**



**Channel output reference signal**

When the TIMERx is used in the compare match output mode, the OxCPRE signal (Channel

x Output prepare signal) is defined by setting the CHxCOMCTL field. The OxCPRE signal has several types of output function. These include, keeping the original level by setting the CHxCOMCTL field to 0x00, set to 1 by setting the CHxCOMCTL field to 0x01, set to 0 by setting the CHxCOMCTL field to 0x02 or signal toggle by setting the CHxCOMCTL field to 0x03 when the counter value matches the content of the TIMERx\_CHxCV register.

The PWM mode 0 and PWM mode 1 outputs are also another kind of OxCPRE output which is setup by setting the CHxCOMCTL field to 0x06/0x07. In these modes, the OxCPRE signal level is changed according to the counting direction and the relationship between the counter value and the TIMERx\_CHxCV content. With regard to a more detail description refer to the relative bit definition.

Another special function of the OxCPRE signal is a forced output which can be achieved by setting the CHxCOMCTL field to 0x04/0x05. Here the output can be forced to an inactive/active level irrespective of the comparison condition between the counter and the TIMERx\_CHxCV values.

The OxCPRE signal can be forced to 0 when the ETIFE signal is derived from the external ETI pin and when it is set to a high level by setting the CHxCOMCEN bit to 1 in the TIMERx\_CHCTL0 register. The OxCPRE signal will not return to its active level until the next update event occurs.

## Quadrature decoder

The quadrature decoder function uses two quadrature inputs CI0 and CI1 derived from the TIMERx\_CH0 and TIMERx\_CH1 pins respectively to interact to generate the counter value. The DIR bit is modified by hardware automatically during each input source transition. The input source can be either CI0 only, CI1 only or both CI0 and CI1, the selection made by setting the SMC [2:0] to 0x01, 0x02 or 0x03. The mechanism for changing the counter direction is shown in the following table. The quadrature decoder can be regarded as an external clock with a directional selection. This means that the counter counts continuously in the interval between 0 and the counter-reload value. Therefore, users must configure the TIMERx\_CAR register before the counter starts to count.

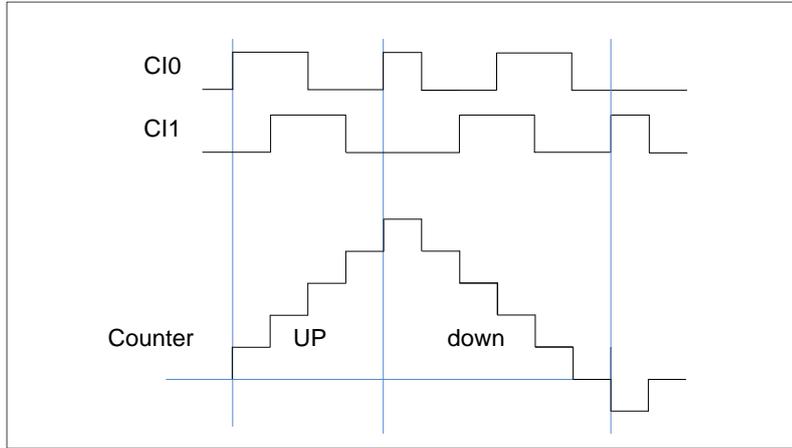
**Table 15-5. Counting direction versus encoder signals**

Counting mode	Level	CI0FE0		CI1FE1	
		Rising	Falling	Rising	Falling
CI0 only counting	CI1FE1=High	Down	Up	-	-
	CI1FE1=Low	Up	Down	-	-
CI1 only counting	CI0FE0=High	-	-	Up	Down
	CI0FE0=Low	-	-	Down	Up
CI0 and CI1 counting	CI1FE1=High	Down	Up	X	X
	CI1FE1=Low	Up	Down	X	X
	CI0FE0=High	X	X	Up	Down

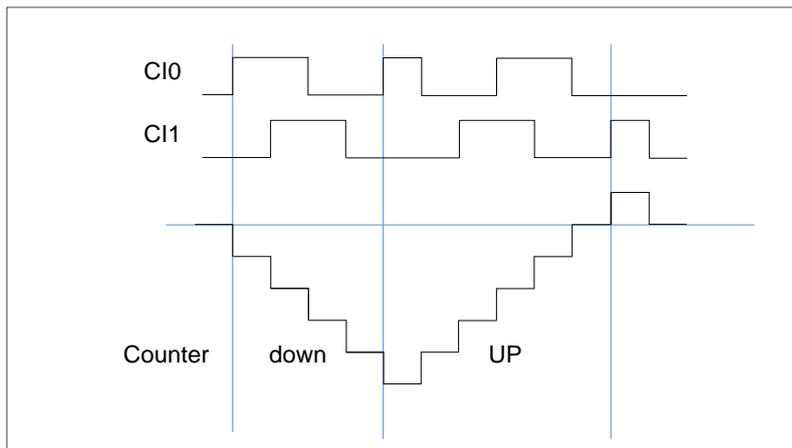
	CI0FE0=Low	X	X	Down	Up
--	------------	---	---	------	----

**Note:** "-" means "no counting"; "X" means impossible.

**Figure 15-44. Example of counter operation in encoder interface mode**



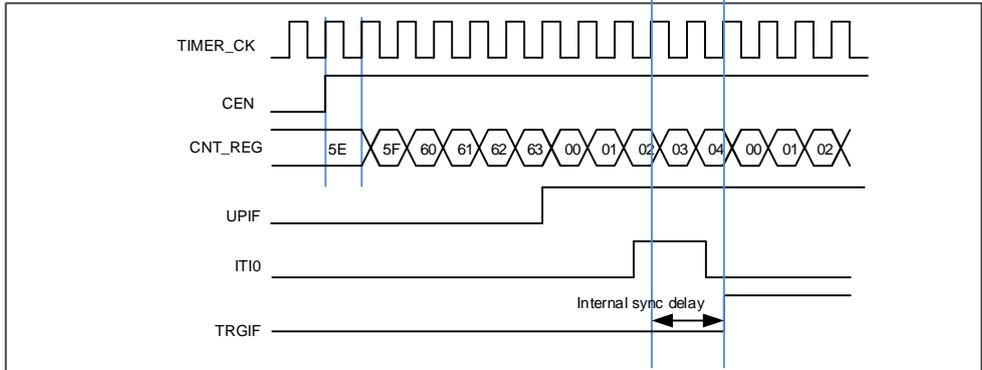
**Figure 15-45. Example of encoder interface mode with CI0FE0 polarity inverted**

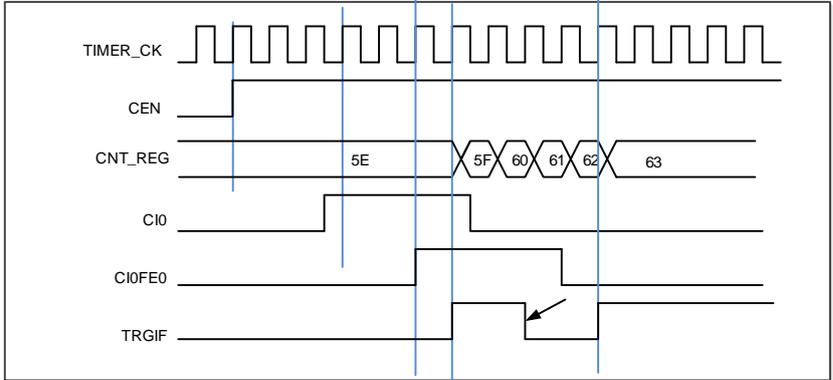
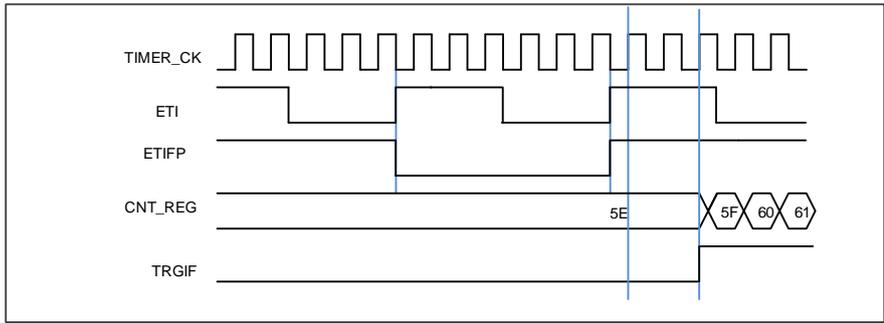


**Slave controller**

The TIMERx can be synchronized with a trigger in several modes including the restart mode, the pause mode and the event mode which is selected by the SMC [2:0] in the TIMERx\_SMCFG register. The trigger input of these modes can be selected by the TRGS [2:0] in the TIMERx\_SMCFG register.

**Table 15-6. Counting direction versus encoder signals**

	Mode Selection	Source Selection	Polarity Selection	Filter and Prescaler
LIST	SMC[2:0] 3'b100 (restart mode) 3'b101 (pause mode) 3'b110 (event mode)	TRGS[2:0] 000: ITI0 001: ITI1 010: ITI2 011: ITI3 100: CI0F_ED 101: CI0FE0 110: CI1FE1 111: ETIFP	If you choose the CI0FE0 or CI1FE1, configure the CHxP and CHxNP for the polarity selection and inversion.  If you choose the ETIF, configure the ETP for polarity selection and inversion.	For the ITIx no filter and prescaler can be used.  For the CIx, configure Filter by CHxCAPFLT, no prescaler can be used.  For the ETIF, configure Filter by ETFC and Prescaler by ETPSC.
Exam1	Restart mode  The counter can be clear and restart when a rising trigger input.	TRGS[2:0]=3'b000  ITI0 is the selection.	For ITI0, no polarity selector can be used.	For the ITI0, no filter and prescaler can be used.
<p><b>Figure 15-46. Restart mode</b></p> 				
Exam2	Pause mode  The counter can be paused when the trigger input is low.	TRGS[2:0]=3'b101  CI0FE0 is the selection.	TI0S=0.(Non-xor) [CH0NP==0, CH0P==0] no inverted. Capture will be sensitive to the rising edge only.	Filter is bypass in this example.

	Mode Selection	Source Selection	Polarity Selection	Filter and Prescaler
	<p><b>Figure 15-47. Pause mode</b></p> 			
Exam3	Event mode The counter will start to count when a rising trigger input.	$TRGS[2:0]=3'b111$ ETIF is the selection.	$ETP = 0$ no polarity change.	$ETPSC = 1$ , divided by 2. $ETFC = 0$ , no filter
	<p><b>Figure 15-48. Event mode</b></p> 			

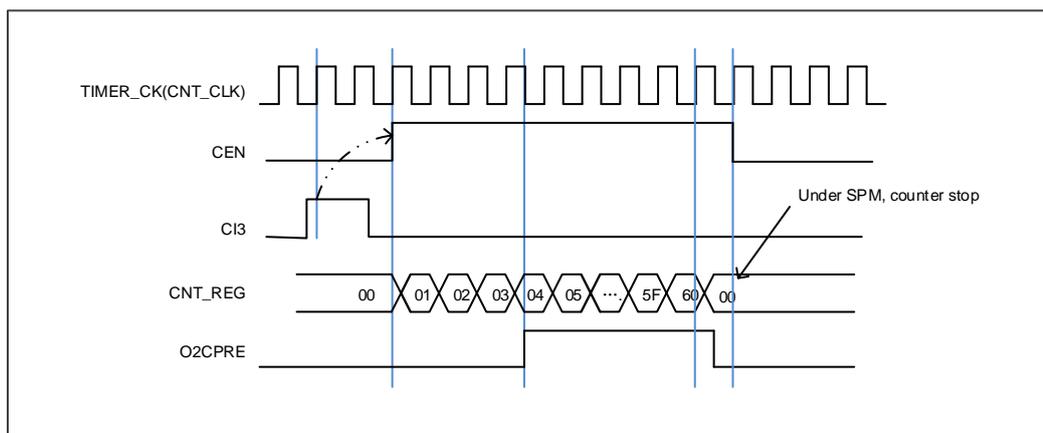
## Single pulse mode

Single pulse mode is opposite to the repetitive mode, which can be enabled by setting SPM in `TIMERx_CTL0`. When you set SPM, the counter will be clear and stop when the next update event automatically. In order to get pulse waveform, you can set the `TIMERx` to PWM mode or compare by `CHxCOMCTL`.

Once the timer is set to operate in the single pulse mode, it is not necessary to set the timer enable bit `CEN` in the `TIMERx_CTL0` register to 1 to enable the counter. The trigger to generate a pulse can be sourced from the trigger signals edge or by setting the `CEN` bit to 1 using software. Setting the `CEN` bit to 1 or a trigger from the trigger signals edge can generate a pulse and then keep the `CEN` bit at a high state until the update event occurs or the `CEN` bit is written to 0 by software. If the `CEN` bit is cleared to 0 using software, the counter will be stopped and its value held. If the `CEN` bit is automatically cleared to 0 by a hardware update event, the counter will be reinitialized.

In the single pulse mode, the trigger active edge which sets the CEN bit to 1 will enable the counter. However, there exist several clock delays to perform the comparison result between the counter value and the `TIMERx_CHxCV` value. In order to reduce the delay to a minimum value, the user can set the `CHxCOMFEN` bit in each `TIMERx_CHCTL0/1` register. After a trigger rising occurs in the single pulse mode, the `OxCPRE` signal will immediately be forced to the state which the `OxCPRE` signal will change to, as the compare match event occurs without taking the comparison result into account. The `CHxCOMFEN` bit is available only when the output channel is configured to operate in the `PWM0` or `PWM1` output mode and the trigger source is derived from the trigger signal.

**Figure 15-49. Single pulse mode `TIMERx_CHxCV = 0x04` `TIMERx_CAR=0x60`**



## Timers interconnection

Refer to [Advanced timer \(TIMERx, x=0\)](#).

**Table 15-7. `TIMERx(x=1,2)` interconnection**

Slave TIMER	IT10(TRGS = 000)	IT11(TRGS = 001)	IT12(TRGS = 010)	IT13(TRGS = 011)
<b>TIMER1</b>	TIMER0	TIMER14	TIMER2	Reserved
<b>TIMER2</b>	TIMER0	TIMER1	TIMER14	Reserved

## Timer DMA mode

Timer's DMA mode is the function that configures timer's register by DMA module. The relative registers are `TIMERx_DMACFG` and `TIMERx_DMATB`; Of course, you have to enable a DMA request which will be asserted by some internal interrupt event. When the interrupt event was asserted, `TIMERx` will send a request to DMA, which is configured to `M2P` mode and `PADDR` is `TIMERx_DMATB`, then DMA will access the `TIMERx_DMATB`. In fact, register `TIMERx_DMATB` is only a buffer; timer will map the `TIMERx_DMATB` to an internal register, appointed by the field of `DMATA` in `TIMERx_DMACFG`. If the field of `DMATC` in `TIMERx_DMACFG` is 0(1 transfer), then the timer's DMA request is finished. While if

TIMERx\_DMATC is not 0, such as 3( 4 transfers), then timer will send 3 more requests to DMA, and DMA will access timer's registers DMASAR+0x4, DMASAR+0x8, DMASAR+0xc at the next 3 accesses to TIMERx\_DMATB. In one word, one time DMA internal interrupt event assert, DMATC+1 times request will be send by TIMERx.

If one more time DMA request event coming, TIMERx will repeat the process as above.

### **Timer debug mode**

When the Cortex™-M3 halted, and the TIMERx\_HOLD configuration bit in DBG\_CTL0 register set to 1, the TIMERx counter stops.

## 15.2.5. TIMERx registers(x=1, 2)

### Control register 0 (TIMERx\_CTL0)

Address offset: 0x00

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CKDIV[1:0]	ARSE	CAM[1:0]	DIR	SPM	UPS	UPDIS	CEN		
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
15:10	Reserved	Must be kept at reset value
9:8	CKDIV[1:0]	<p>Clock division</p> <p>The CKDIV bits can be configured by software to specify division ratio between the timer clock (TIMER_CK) and the dead-time and sampling clock (DTS), which is used by the dead-time generators and the digital filters.</p> <p>00: <math>f_{DTS}=f_{TIMER\_CK}</math>            01: <math>f_{DTS}=f_{TIMER\_CK}/2</math>            10: <math>f_{DTS}=f_{TIMER\_CK}/4</math>            11: Reserved</p>
7	ARSE	<p>Auto-reload shadow enable</p> <p>0: The shadow register for TIMERx_CAR register is disabled            1: The shadow register for TIMERx_CAR register is enabled</p>
6:5	CAM[1:0]	<p>Counter aligns mode selection</p> <p>00: No center-aligned mode (edge-aligned mode). The direction of the counter is specified by the DIR bit.</p> <p>01: Center-aligned and counting down assert mode. The counter counts under center-aligned and channel is configured in output mode (CHxMS=00 in TIMERx_CHCTL0 register). Only when the counter is counting down, compare interrupt flag of channels can be set.</p> <p>10: Center-aligned and counting up assert mode. The counter counts under center-aligned and channel is configured in output mode (CHxMS=00 in TIMERx_CHCTL0 register). Only when the counter is counting up, compare interrupt flag of channels can be set.</p> <p>11: Center-aligned and counting up/down assert mode. The counter counts under center-aligned and channel is configured in output mode (CHxMS=00 in TIMERx_CHCTL0 register). Both when the counter is counting up and counting down, compare interrupt flag of channels can be set.</p> <p>After the counter is enabled, cannot be switched from 0x00 to non 0x00.</p>

4	DIR	<p>Direction</p> <p>0: Count up</p> <p>1: Count down</p> <p>This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.</p>
3	SPM	<p>Single pulse mode.</p> <p>0: Counter continues after update event.</p> <p>1: The CEN is cleared by hardware and the counter stops at next update event.</p>
2	UPS	<p>Update source</p> <p>This bit is used to select the update event sources by software.</p> <p>0: When enabled, any of the following events generate an update interrupt or DMA request:</p> <ul style="list-style-type: none"> <li>- The UPG bit is set</li> <li>- The counter generates an overflow or underflow event</li> <li>- The slave mode controller generates an update event.</li> </ul> <p>1: When enabled, only counter overflow/underflow generates an update interrupt or DMA request.</p>
1	UPDIS	<p>Update disable.</p> <p>This bit is used to enable or disable the update event generation.</p> <p>0: update event enable. The update event is generate and the buffered registers are loaded with their preloaded values when one of the following events occurs:</p> <ul style="list-style-type: none"> <li>- The UPG bit is set</li> <li>- The counter generates an overflow or underflow event</li> <li>- The slave mode controller generates an update event.</li> </ul> <p>1: update event disable. The buffered registers keep their value, while the counter and the prescaler are reinitialized if the UG bit is set or if the slave mode controller generates a hardware reset event.</p>
0	CEN	<p>Counter enable</p> <p>0: Counter disable</p> <p>1: Counter enable</p> <p>The CEN bit must be set by software when timer works in external clock, pause mode and encoder mode. While in event mode, the hardware can set the CEN bit automatically.</p>

## Control register 1 (TIMERx\_CTL1)

Address offset: 0x04

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)

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

Reserved	TI0S	MMC[2:0]	DMAS	Reserved
	rw	rw	rw	

Bits	Fields	Descriptions
15:8	Reserved	Must be kept at reset value
7	TI0S	Channel 0 trigger input selection 0: The TIMERx_CH0 pin input is selected as channel 0 trigger input. 1: The result of combinational XOR of TIMERx_CH0, CH1 and CH2 pins is selected as channel 0 trigger input.
6:4	MMC[2:0]	Master mode control These bits control the selection of TRGO signal, which is sent in master mode to slave timers for synchronization function. 000: Reset. When the UPG bit in the TIMERx_SWEVG register is set or a reset is generated by the slave mode controller, a TRGO pulse occurs. And in the latter case, the signal on TRGO is delayed compared to the actual reset. 001: Enable. This mode is useful to start several timers at the same time or to control a window in which a slave timer is enabled. In this mode the master mode controller selects the counter enable signal TIMERx_EN as TRGO. The counter enable signal is set when CEN control bit is set or the trigger input in pause mode is high. There is a delay between the trigger input in pause mode and the TRGO output, except if the master-slave mode is selected. 010: Update. In this mode the master mode controller selects the update event as TRGO. 011: Capture/compare pulse. In this mode the master mode controller generates a TRGO pulse when a capture or a compare match occurred. 100: Compare. In this mode the master mode controller selects the O0CPRE signal is used as TRGO 101: Compare. In this mode the master mode controller selects the O1CPRE signal is used as TRGO 110: Compare. In this mode the master mode controller selects the O2CPRE signal is used as TRGO 111: Compare. In this mode the master mode controller selects the O3CPRE signal is used as TRGO
3	DMAS	DMA request source selection 0: DMA request of channel x is sent when channel x event occurs. 1: DMA request of channel x is sent when update event occurs.
2:0	Reserved	Must be kept at reset value.

### Slave mode configuration register (TIMERx\_SMCFG)

Address offset: 0x08

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	SMC1	ETPSC[1:0]		ETFC[3:0]			MSM	TRGS[2:0]		OCRC	SMC[2:0]				
rw	rw	rw		rw			rw	rw			rw				

Bits	Fields	Descriptions
15	ETP	External trigger polarity This bit specifies the polarity of ETI signal 0: ETI is active at high level or rising edge. 1: ETI is active at low level or falling edge.
14	SMC1	Part of SMC for enable External clock mode1. In external clock mode 1, the counter is clocked by any active edge on the ETIF signal. 0: External clock mode 1 disabled 1: External clock mode 1 enabled. It is possible to simultaneously use external clock mode 1 with the restart mode, pause mode or event mode. But the TRGS bits must not be 3'b111 in this case. The external clock input will be ETIF if external clock mode 0 and external clock mode 1 are enabled at the same time. <b>Note:</b> External clock mode 0 enable is in this register's SMC bit-filed.
13:12	ETPSC[1:0]	External trigger prescaler The frequency of external trigger signal ETI must not be at higher than 1/4 of TIMER_CK frequency. When the external trigger signal is a fast clock, the prescaler can be enabled to reduce ETI frequency. 00: Prescaler disable 01: ETI frequency will be divided by 2 10: ETI frequency will be divided by 4 11: ETI frequency will be divided by 8
11:8	ETFC[3:0]	External trigger filter control An event counter is used in the digital filter, in which a transition on the output occurs after N input events. This bit-field specifies the frequency used to sample ETI signal and the length of the digital filter applied to ETI. 0000: Filter disabled. $f_{SAMP} = f_{DTS}$ , $N=1$ . 0001: $f_{SAMP} = f_{TIMER\_CK}$ , $N=2$ . 0010: $f_{SAMP} = f_{TIMER\_CK}$ , $N=4$ . 0011: $f_{SAMP} = f_{TIMER\_CK}$ , $N=8$ . 0100: $f_{SAMP} = f_{DTS}/2$ , $N=6$ . 0101: $f_{SAMP} = f_{DTS}/2$ , $N=8$ . 0110: $f_{SAMP} = f_{DTS}/4$ , $N=6$ . 0111: $f_{SAMP} = f_{DTS}/4$ , $N=8$ . 1000: $f_{SAMP} = f_{DTS}/8$ , $N=6$ . 1001: $f_{SAMP} = f_{DTS}/8$ , $N=8$ .

		1010: $f_{SAMP}=f_{DTS}/16$ , N=5.
		1011: $f_{SAMP}=f_{DTS}/16$ , N=6.
		1100: $f_{SAMP}=f_{DTS}/16$ , N=8.
		1101: $f_{SAMP}=f_{DTS}/32$ , N=5.
		1110: $f_{SAMP}=f_{DTS}/32$ , N=6.
		1111: $f_{SAMP}=f_{DTS}/32$ , N=8.
7	MSM	<p>Master-slave mode</p> <p>This bit can be used to synchronize selected timers to begin counting at the same time. The TRGI is used as the start event, and through TRGO, timers are connected together.</p> <p>0: Master-slave mode disable 1: Master-slave mode enable</p>
6:4	TRGS[2:0]	<p>Trigger selection</p> <p>This bit-field specifies which signal is selected as the trigger input, which is used to synchronize the counter.</p> <p>000: Internal trigger input 0 (IT10) 001: Internal trigger input 1 (IT11) 010: Internal trigger input 2 (IT12) 011: Reserved 100: CI0 edge flag (CI0F_ED) 101: channel 0 input Filtered output (CI0FE0) 110: channel 1 input Filtered output (CI1FE1) 111: External trigger input filter output(ETIFP)</p> <p>These bits must not be changed when slave mode is enabled.</p>
3	OCRC	<p>OCPRE clear source selection</p> <p>0: OCPRE_CLR_INT is connected to the OCPRE_CLR input 1: OCPRE_CLR_INT is connected to ETIF</p>
2:0	SMC[2:0]	<p>Slave mode control</p> <p>000: Disable mode. The slave mode is disabled; The prescaler is clocked directly by the internal clock (TIMER_CK) when CEN bit is set high. 001: Quadrature decoder mode 0.The counter counts on CI1FE1 edge, while the direction depends on CI0FE0 level. 010: Quadrature decoder mode 1.The counter counts on CI0FE0 edge, while the direction depends on CI1FE1 level. 011: Quadrature decoder mode 2.The counter counts on both CI0FE0 and CI1FE1 edge, while the direction depends on each other. 100: Restart mode. The counter is reinitialized and the shadow registers are updated on the rising edge of the selected trigger input. 101: Pause mode. The trigger input enables the counter clock when it is high and disables the counter when it is low. 110: Event mode. A rising edge of the trigger input enables the counter. The counter cannot be disabled by the slave mode controller.</p>

111: External clock mode0. The counter counts on the rising edges of the selected trigger.

## DMA and interrupt enable register (TIMERx\_DMAINTEN)

Address offset: 0x0C

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	TRGDEN	Reserved	CH3DEN	CH2DEN	CH1DEN	CH0DEN	UPDEN	Reserved	TRGIE	Reserved	CH3IE	CH2IE	CH1IE	CH0IE	UPIE
	rw		rw	rw	rw	rw	rw		rw		rw	rw	rw	rw	rw

Bits	Fields	Descriptions
15	Reserved	Must be kept at reset value.
14	TRGDEN	Trigger DMA request enable 0: disabled 1: enabled
13	Reserved	Must be kept at reset value.
12	CH3DEN	Channel 3 capture/compare DMA request enable 0: disabled 1: enabled
11	CH2DEN	Channel 2 capture/compare DMA request enable 0: disabled 1: enabled
10	CH1DEN	Channel 1 capture/compare DMA request enable 0: disabled 1: enabled
9	CH0DEN	Channel 0 capture/compare DMA request enable 0: disabled 1: enabled
8	UPDEN	Update DMA request enable 0: disabled 1: enabled
7	Reserved	Must be kept at reset value.
6	TRGIE	Trigger interrupt enable 0: disabled 1: enabled

5	Reserved	Must be kept at reset value.
4	CH3IE	Channel 3 capture/compare interrupt enable 0: disabled 1: enabled
3	CH2IE	Channel 2 capture/compare interrupt enable 0: disabled 1: enabled
2	CH1IE	Channel 1 capture/compare interrupt enable 0: disabled 1: enabled
1	CH0IE	Channel 0 capture/compare interrupt enable 0: disabled 1: enabled
0	UPIE	Update interrupt enable 0: disabled 1: enabled

### Interrupt flag register (TIMERx\_INTF)

Address offset: 0x10

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved				CH3OF	CH2OF	CH1OF	CH0OF	Reserved		TRGIF	Reserved	CH3IF	CH3IF	CH1IF	CH0IF	UPIF
				rc_w0	rc_w0	rc_w0	rc_w0			rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

Bits	Fields	Descriptions
15:13	Reserved	Must be kept at reset value.
12	CH3OF	Channel 3 over capture flag Refer to CH0OF description
11	CH2OF	Channel 2 over capture flag Refer to CH0OF description
10	CH1OF	Channel 1 over capture flag Refer to CH0OF description
9	CH0OF	Channel 0 over capture flag When channel 0 is configured in input mode, this flag is set by hardware when a capture event occurs while CH0IF flag has already been set. This flag is cleared by software.

		0: No over capture interrupt occurred 1: Over capture interrupt occurred
8:7	Reserved	Must be kept at reset value.
6	TRGIF	Trigger interrupt flag This flag is set by hardware on trigger event and cleared by software. When the slave mode controller is enabled in all modes but pause mode, an active edge on trigger input generates a trigger event. When the slave mode controller is enabled in pause mode both edges on trigger input generates a trigger event. 0: No trigger event occurred. 1: Trigger interrupt occurred.
5	Reserved	Must be kept at reset value.
4	CH3IF	Channel 3 's capture/compare interrupt enable Refer to CH0IF description
3	CH2IF	Channel 2 's capture/compare interrupt enable Refer to CH0IF description
2	CH1IF	Channel 1 's capture/compare interrupt flag Refer to CH0IF description
1	CH0IF	Channel 0 's capture/compare interrupt flag This flag is set by hardware and cleared by software. When channel 0 is in input mode, this flag is set when a capture event occurs. When channel 0 is in output mode, this flag is set when a compare event occurs. 0: No Channel 1 interrupt occurred 1: Channel 1 interrupt occurred
0	UPIF	Update interrupt flag This bit is set by hardware on an update event and cleared by software. 0: No update interrupt occurred 1: Update interrupt occurred

### Software event generation register (TIMERx\_SWEVG)

Address offset: 0x14

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved										TRGG	Reserved	CH3G	CH2G	CH1G	CH0G	UPG

Bits	Fields	Descriptions
15:7	Reserved	Must be kept at reset value.
6	TRGG	<p>Trigger event generation</p> <p>This bit is set by software and cleared by hardware automatically. When this bit is set, the TRGIF flag in TIMERx_STAT register is set, related interrupt or DMA transfer can occur if enabled.</p> <p>0: No generate a trigger event 1: Generate a trigger event</p>
5	Reserved	Must be kept at reset value.
4	CH3G	<p>Channel 3's capture or compare event generation</p> <p>Refer to CH0G description</p>
3	CH2G	<p>Channel 2's capture or compare event generation</p> <p>Refer to CH0G description</p>
2	CH1G	<p>Channel 1's capture or compare event generation</p> <p>Refer to CH0G description</p>
1	CH0G	<p>Channel 0's capture or compare event generation</p> <p>This bit is set by software in order to generate a capture or compare event in channel 0, it is automatically cleared by hardware. When this bit is set, the CH1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. In addition, if channel 1 is configured in input mode, the current value of the counter is captured in TIMERx_CH0CV register, and the CH0OF flag is set if the CH0IF flag was already high.</p> <p>0: No generate a channel 1 capture or compare event 1: Generate a channel 1 capture or compare event</p>
0	UPG	<p>This bit can be set by software, and cleared by hardware automatically. When this bit is set, the counter is cleared if the center-aligned or up counting mode is selected, else (down counting) it takes the auto-reload value. The prescaler counter is cleared at the same time.</p> <p>0: No generate an update event 1: Generate an update event</p>

### Channel control register 0 (TIMERx\_CHCTL0)

Address offset: 0x18

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH1COMCEN	CH1COMCTL[2:0]			CH1COMSEN	CH1COMFEN	CH1MS[1:0]		CH0COMCEN	CH0COMCTL[2:0]			CH0COMSEN	CH0COMFEN	CH0MS[1:0]	
CH1CAPFLT[3:0]				CH1CAPPSC[1:0]					CH0CAPFLT[3:0]			CH0CAPPSC[1:0]			
Rw				rw		rw			rw			rw		rw	

**Output compare mode:**

Bits	Fields	Descriptions
15	CH1COMCEN	Channel 1 output compare clear enable Refer to CH0COMCEN description
14:12	CH1COMCTL[2:0]	Channel 1 compare output control Refer to CH0COMCTL description
11	CH1COMSEN	Channel 1 output compare shadow enable Refer to CH0COMSEN description
10	CH1COMFEN	Channel 1 output compare fast enable Refer to CH0COMSEN description
9:8	CH1MS[1:0]	Channel 1 mode selection This bit-field specifies the direction of the channel and the input signal selection. This bit-field is writable only when the channel is not active. (CH1EN bit in TIMERx_CHCTL2 register is reset). 00: Channel 1 is configured as output 01: Channel 1 is configured as input, IS1 is connected to CI0FE1 10: Channel 1 is configured as input, IS1 is connected to CI1FE1 11: Channel 1 is configured as input, IS1 is connected to ITS. This mode is working only if an internal trigger input is selected through TRGS bits in TIMERx_SMCFG register.
7	CH0COMCEN	Channel 0 output compare clear enable. When this bit is set, the O0CPRE signal is cleared when High level is detected on ETIF input. 0: Channel 0 output compare clear disable 1: Channel 0 output compare clear enable
6:4	CH0COMCTL[2:0]	Channel 0 compare output control This bit-field controls the behavior of the output reference signal O0CPRE which drives CH0_O and CH0_ON. O0CPRE is active high, while CH0_O and CH0_ON active level depends on CH0P and CH0NP bits. 000: Frozen. The O0CPRE signal keeps stable, independent of the comparison between the register TIMERx_CH0CV and the counter TIMERx_CNT. 001: Set the channel output. O0CPRE signal is forced high when the counter matches the output compare register TIMERx_CH0CV. 010: Clear the channel output. O0CPRE signal is forced low when the counter matches the output compare register TIMERx_CH0CV.

		<p>011: Toggle on match. O0CPRE toggles when the counter matches the output compare register <code>TIMERx_CH0CV</code>.</p> <p>100: Force low. O0CPRE is forced low level.</p> <p>101: Force high. O0CPRE is forced high level.</p> <p>110: PWM mode0. When counting up, O0CPRE is active as long as the counter is smaller than <code>TIMERx_CH0CV</code> else inactive. When counting down, O0CPRE is inactive as long as the counter is larger than <code>TIMERx_CH0CV</code> else active.</p> <p>111: PWM mode1. When counting up, O0CPRE is inactive as long as the counter is smaller than <code>TIMERx_CH0CV</code> else active. When counting down, O0CPRE is active as long as the counter is larger than <code>TIMERx_CH0CV</code> else inactive.</p> <p>When configured in PWM mode, the O0CPRE level changes only when the output compare mode switches from “frozen” mode to “PWM” mode or when the result of the comparison changes.</p> <p>This bit cannot be modified when <code>PROT [1:0]</code> bit-filed in <code>TIMERx_CCHP</code> register is 11 and <code>CH0MS</code> bit-filed is 00(COMPARE MODE).</p>
3	CH0COMSEN	<p>Channel 0 compare output shadow enable</p> <p>When this bit is set, the shadow register of <code>TIMERx_CH0CV</code> register, which updates at each update event, will be enabled.</p> <p>0: Channel 0 output compare shadow disable</p> <p>1: Channel 0 output compare shadow enable</p> <p>The PWM mode can be used without validating the shadow register only in single pulse mode (<code>SPM</code> bit in <code>TIMERx_CTL0</code> register is set).</p> <p>This bit cannot be modified when <code>PROT [1:0]</code> bit-filed in <code>TIMERx_CCHP</code> register is 11 and <code>CH0MS</code> bit-filed is 00.</p>
2	CH0COMFEN	<p>Channel 0 output compare fast enable</p> <p>When this bit is set, the effect of an event on the trigger in input on the capture/compare output will be accelerated if the channel is configured in PWM0 or PWM1 mode. The output channel will treat an active edge on the trigger input as a compare match, and <code>CH0_O</code> is set to the compare level independently from the result of the comparison.</p> <p>0: Channel 0 output quickly compare disable. The minimum delay from an edge on the trigger input to activate <code>CH0_O</code> output is 5 clock cycles.</p> <p>1: Channel 0 output quickly compare enable. The minimum delay from an edge on the trigger input to activate <code>CH0_O</code> output is 3 clock cycles.</p>
1:0	CH0MS[1:0]	<p>Channel 0 I/O mode selection</p> <p>This bit-field specifies the work mode of the channel and the input signal selection. This bit-field is writable only when the channel is not active. (<code>CH0EN</code> bit in <code>TIMERx_CHCTL2</code> register is reset.).</p> <p>00: Channel 0 is configured as output</p> <p>01: Channel 0 is configured as input, <code>IS0</code> is connected to <code>CI0FE0</code></p> <p>10: Channel 0 is configured as input, <code>IS0</code> is connected to <code>CI1FE0</code></p> <p>11: Channel 0 is configured as input, <code>IS0</code> is connected to <code>ITS</code>. This mode is</p>

working only if an internal trigger input is selected through TRGS bits in  
 TIMERx\_SMCFG register.

**Input capture mode:**

Bits	Fields	Descriptions
15:12	CH1CAPFLT[3:0]	Channel 1 input capture filter control Refer to CH0CAPFLT description
11:10	CH1CAPPSC[1:0]	Channel 1 input capture prescaler Refer to CH0CAPPSC description
9:8	CH1MS[1:0]	Channel 1 mode selection Same as Output compare mode
7:4	CH0CAPFLT[3:0]	Channel 0 input capture filter control An event counter is used in the digital filter, in which a transition on the output occurs after N input events. This bit-field specifies the frequency used to sample CIO input signal and the length of the digital filter applied to CIO. 0000: Filter disabled, $f_{SAMP}=f_{DTS}$ , $N=1$ 0001: $f_{SAMP}=f_{TIMER\_CK}$ , $N=2$ 0010: $f_{SAMP}=f_{TIMER\_CK}$ , $N=4$ 0011: $f_{SAMP}=f_{TIMER\_CK}$ , $N=8$ 0100: $f_{SAMP}=f_{DTS}/2$ , $N=6$ 0101: $f_{SAMP}=f_{DTS}/2$ , $N=8$ 0110: $f_{SAMP}=f_{DTS}/4$ , $N=6$ 0111: $f_{SAMP}=f_{DTS}/4$ , $N=8$ 1000: $f_{SAMP}=f_{DTS}/8$ , $N=6$ 1001: $f_{SAMP}=f_{DTS}/8$ , $N=8$ 1010: $f_{SAMP}=f_{DTS}/16$ , $N=5$ 1011: $f_{SAMP}=f_{DTS}/16$ , $N=6$ 1100: $f_{SAMP}=f_{DTS}/16$ , $N=8$ 1101: $f_{SAMP}=f_{DTS}/32$ , $N=5$ 1110: $f_{SAMP}=f_{DTS}/32$ , $N=6$ 1111: $f_{SAMP}=f_{DTS}/32$ , $N=8$
3:2	CH0CAPPSC[1:0]	Channel 0 input capture prescaler This bit-field specifies the factor of the prescaler on channel 0 input. The prescaler is reset when CH0EN bit in TIMERx_CHCTL2 register is clear. 00: Prescaler disable, capture is done on each channel input edge 01: Capture is done every 2 channel input edges 10: Capture is done every 4 channel input edges 11: Capture is done every 8 channel input edges
1:0	CH0MS[1:0]	Channel 0 mode selection Same as Output compare mode

**Channel control register 1 (TIMERx\_CHCTL1)**

Address offset: 0x1C

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH3COM CEN	CH3COMCTL[2:0]			CH3COM SEN	CH3COM FEN	CH3MS[1:0]		CH2COM CEN	CH2COMCTL[2:0]			CH2COM SEN	CH2COM FEN	CH2MS[1:0]	
CH3CAPFLT[3:0]				CH3CAPPSC[1:0]				CH2CAPFLT[3:0]			CH2CAPPSC[1:0]				
Rw				rw		rw		rw			rw		rw		

**Output compare mode:**

Bits	Fields	Descriptions
15	CH3COMCEN	Channel 3 output compare clear enable Refer to CH0COMCEN description
14:12	CH3COMCTL[2:0]	Channel 3 compare output control Refer to CH0COMCTL description
11	CH3COMSEN	Channel 3 output compare shadow enable Refer to CH0COMSEN description
10	CH3COMFEN	Channel 3 output compare fast enable Refer to CH0COMSEN description
9:8	CH3MS[1:0]	Channel 3 mode selection This bit-field specifies the direction of the channel and the input signal selection. This bit-field is writable only when the channel is not active. (CH3EN bit in TIMERx_CHCTL2 register is reset). 00: Channel 3 is configured as output 01: Channel 3 is configured as input, IS3 is connected to CI2FE3 10: Channel 3 is configured as input, IS3 is connected to CI3FE3 11: Channel 3 is configured as input, IS3 is connected to ITS. This mode is working only if an internal trigger input is selected through TRGS bits in TIMERx_SMCFG register.
7	CH2COMCEN	Channel 2 output compare clear enable. When this bit is set, the O2CPRE signal is cleared when High level is detected on ETIF input. 0: Channel 2 output compare clear disable 1: Channel 2 output compare clear enable
6:4	CH2COMCTL[2:0]	Channel 2 compare output control This bit-field controls the behavior of the output reference signal O2CPRE which drives CH2_O and CH2_ON. O2CPRE is active high, while CH2_O and CH2_ON active level depends on CH2P and CH2NP bits.

		<p>000: Frozen. The O2CPRE signal keeps stable, independent of the comparison between the output compare register <code>TIMERx_CH2CV</code> and the counter <code>TIMERx_CNT</code>.</p> <p>001: Set high on match. O2CPRE signal is forced high when the counter matches the output compare register <code>TIMERx_CH2CV</code>.</p> <p>010: Set low on match. O2CPRE signal is forced low when the counter matches the output compare register <code>TIMERx_CH2CV</code>.</p> <p>011: Toggle on match. O2CPRE toggles when the counter matches the output compare register <code>TIMERx_CH2CV</code>.</p> <p>100: Force low. O2CPRE is forced low level.</p> <p>101: Force high. O2CPRE is forced high level.</p> <p>110: PWM mode0. When counting up, O0CPRE is active as long as the counter is smaller than <code>TIMERx_CH0CV</code> else inactive. When counting down, O0CPRE is inactive as long as the counter is larger than <code>TIMERx_CH0CV</code> else active.</p> <p>111: PWM mode1. When counting up, O0CPRE is inactive as long as the counter is smaller than <code>TIMERx_CH0CV</code> else active. When counting down, O0CPRE is active as long as the counter is larger than <code>TIMERx_CH0CV</code> else inactive.</p> <p>When configured in PWM mode, the O2CPRE level changes only when the output compare mode switches from “frozen” mode to “PWM” mode or when the result of the comparison changes.</p> <p>This bit cannot be modified when <code>PROT [1:0]</code> bit-filed in <code>TIMERx_CCHP</code> register is 11 and <code>CH2MS</code> bit-filed is 00(COMPARE MODE).</p>
3	CH2COMSEN	<p>Channel 2 compare output shadow enable</p> <p>When this bit is set, the shadow register of <code>TIMERx_CH2CV</code> register, which updates at each update event will be enabled.</p> <p>0: Channel 2 output compare shadow disable</p> <p>1: Channel 2 output compare shadow enable</p> <p>The PWM mode can be used without validating the shadow register only in single pulse mode (<code>SPM</code> bit in <code>TIMERx_CTL0</code> register is set).</p> <p>This bit cannot be modified when <code>PROT [1:0]</code> bit-filed in <code>TIMERx_CCHP</code> register is 11 and <code>CH0MS</code> bit-filed is 00.</p>
2	CH2COMFEN	<p>Channel 2 output compare fast enable</p> <p>When this bit is set, the effect of an event on the trigger in input on the capture/compare output will be accelerated if the channel is configured in PWM1 or PWM2 mode. The output channel will treat an active edge on the trigger input as a compare match, and <code>CH2_O</code> is set to the compare level independently from the result of the comparison.</p> <p>0: Channel 2 output quickly compare disable. The minimum delay from an edge on the trigger input to activate <code>CH2_O</code> output is 5 clock cycles.</p> <p>1: Channel 2 output quickly compare enable. The minimum delay from an edge on the trigger input to activate <code>CH2_O</code> output is 3 clock cycles.</p>
1:0	CH2MS[1:0]	Channel 2 I/O mode selection

This bit-field specifies the work mode of the channel and the input signal selection.

This bit-field is writable only when the channel is not active. (CH2EN bit in TIMERx\_CHCTL2 register is reset).

00: Channel 2 is configured as output

01: Channel 2 is configured as input, IS2 is connected to CI2FE2

10: Channel 2 is configured as input, IS2 is connected to CI3FE2

11: Channel 2 is configured as input, IS2 is connected to ITS. This mode is working only if an internal trigger input is selected through TRGS bits in TIMERx\_SMCFG register.

#### Input capture mode:

Bits	Fields	Descriptions
15:12	CH3CAPFLT[3:0]	Channel 3 input capture filter control Refer to CH0CAPFLT description
11:10	CH3CAPPSC[1:0]	Channel 3 input capture prescaler Refer to CH0CAPPSC description
9:8	CH3MS[1:0]	Channel 3 mode selection Same as Output compare mode
7:4	CH2CAPFLT[3:0]	Channel 2 input capture filter control An event counter is used in the digital filter, in which a transition on the output occurs after N input events. This bit-field specifies the frequency used to sample CI2 input signal and the length of the digital filter applied to CI2. 0000: Filter disable, $f_{SAMP}=f_{DTS}$ , N=1 0001: $f_{SAMP}=f_{TIMER\_CK}$ , N=2 0010: $f_{SAMP}=f_{TIMER\_CK}$ , N=4 0011: $f_{SAMP}=f_{TIMER\_CK}$ , N=8 0100: $f_{SAMP}=f_{DTS}/2$ , N=6 0101: $f_{SAMP}=f_{DTS}/2$ , N=8 0110: $f_{SAMP}=f_{DTS}/4$ , N=6 0111: $f_{SAMP}=f_{DTS}/4$ , N=8 1000: $f_{SAMP}=f_{DTS}/8$ , N=6 1001: $f_{SAMP}=f_{DTS}/8$ , N=8 1010: $f_{SAMP}=f_{DTS}/16$ , N=5 1011: $f_{SAMP}=f_{DTS}/16$ , N=6 1100: $f_{SAMP}=f_{DTS}/16$ , N=8 1101: $f_{SAMP}=f_{DTS}/32$ , N=5 1110: $f_{SAMP}=f_{DTS}/32$ , N=6 1111: $f_{SAMP}=f_{DTS}/32$ , N=8
3:2	CH2CAPPSC[1:0]	Channel 2 input capture prescaler This bit-field specifies the factor of the prescaler on channel 2 input. The prescaler is reset when CH2EN bit in TIMERx_CHCTL2 register is clear.

00: Prescaler disable, capture is done on each channel input edge

01: Capture is done every 2 channel input edges

10: Capture is done every 4 channel input edges

11: Capture is done every 8 channel input edges

1:0 CH2MS[1:0] Channel 2 mode selection  
Same as output compare mode

### Channel control register 2 (TIMERx\_CHCTL2)

Address offset: 0x20

Reset value: 0x0000

This register can be accessed by half-word(16-bit) or word(32-bit)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH3NP	Reserved	CH3P	CH3EN	CH2NP	Reserved	CH2P	CH2EN	CH1NP	Reserved	CH1P	CH1EN	CH0NP	Reserved	CH0P	CH0EN
rw		rw	rw												

Bits	Fields	Descriptions
15	CH3NP	Channel 3 complementary output polarity Refer to CH0NP description
14	Reserved	Must be kept at reset value
13	CH3P	Channel 3 capture/compare function polarity Refer to CH0P description
12	CH3EN	Channel 3 capture/compare function enable Refer to CH0EN description
11	CH2NP	Channel 2 complementary output polarity Refer to CH0NP description
10	Reserved	Must be kept at reset value
9	CH2P	Channel 2 capture/compare function polarity Refer to CH0P description
8	CH2EN	Channel 2 capture/compare function enable Refer to CH0EN description
7	CH1NP	Channel 1 complementary output polarity Refer to CH0NP description
6	Reserved	Must be kept at reset value
5	CH1P	Channel 1 capture/compare function polarity Refer to CH0P description

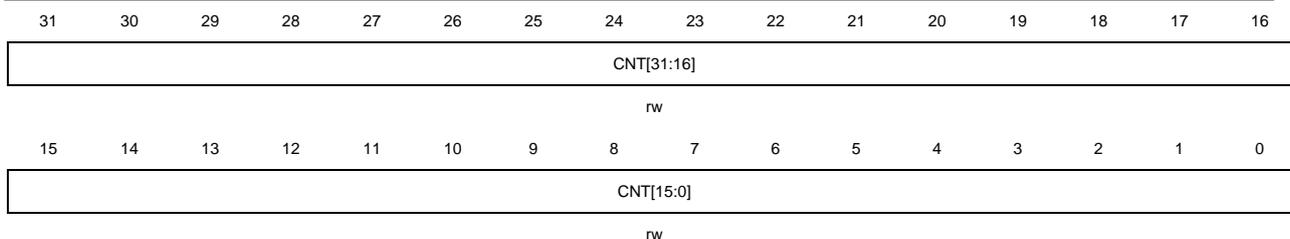
4	CH1EN	Channel 1 capture/compare function enable Refer to CH0EN description
3	CH0NP	Channel 0 complementary output polarity When channel 0 is configured in output mode, this bit should be keep reset value. When channel 0 is configured in input mode, In conjunction with CH0P, this bit is used to define the polarity of CI0. This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 or 10.
2	Reserved	Must be kept at reset value
1	CH0P	Channel 0 capture/compare function polarity When channel 0 is configured in output mode, this bit specifies the output signal polarity. 0: Channel 0 active high 1: Channel 0 active low When channel 0 is configured in input mode, this bit specifies the CI0 signal polarity. [CH0NP, CH0P] will select the active trigger or capture polarity for CI0FE0 or CI1FE0. [CH0NP==0, CH0P==0]: CIxFE0's rising edge is the active signal for capture or trigger operation in slave mode. And CIxFE0 will not be inverted. [CH0NP==0, CH0P==1]: CIxFE0's falling edge is the active signal for capture or trigger operation in slave mode. And CIxFE0 will be inverted. [CH0NP==1, CH0P==0]: Reserved. [CH0NP==1, CH0P==1]: CIxFE0's falling and rising edge are both the active signal for capture or trigger operation in slave mode. And CIxFE0 will be not inverted. This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 or 10.
0	CH0EN	Channel 0 capture/compare function enable When channel 0 is configured in output mode, setting this bit enables CH0_O signal in active state. When channel 0 is configured in input mode, setting this bit enables the capture event in channel0. 0: Channel 0 disabled 1: Channel 0 enabled

### Counter register (TIMERx\_CNT) (x=1)

Address offset: 0x24

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



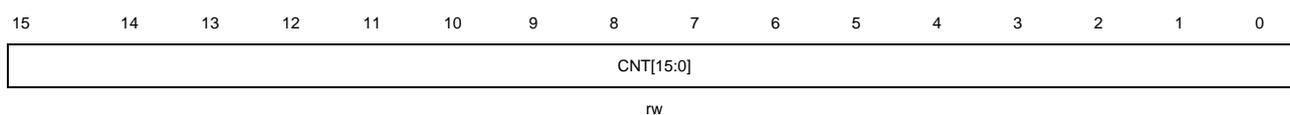
Bits	Fields	Descriptions
15:0	CNT[31:0]	This bit-field indicates the current counter value. Writing to this bit-field can change the value of the counter.

### Counter register (TIMERx\_CNT) (x=2)

Address offset: 0x24

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



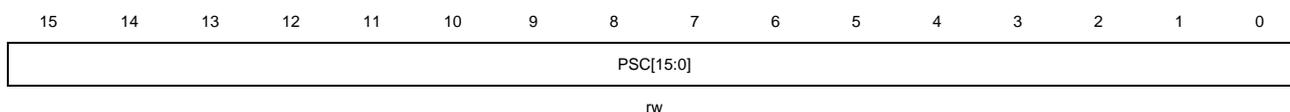
Bits	Fields	Descriptions
15:0	CNT[15:0]	This bit-field indicates the current counter value. Writing to this bit-field can change the value of the counter.

### Prescaler register (TIMERx\_PSC)

Address offset: 0x28

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



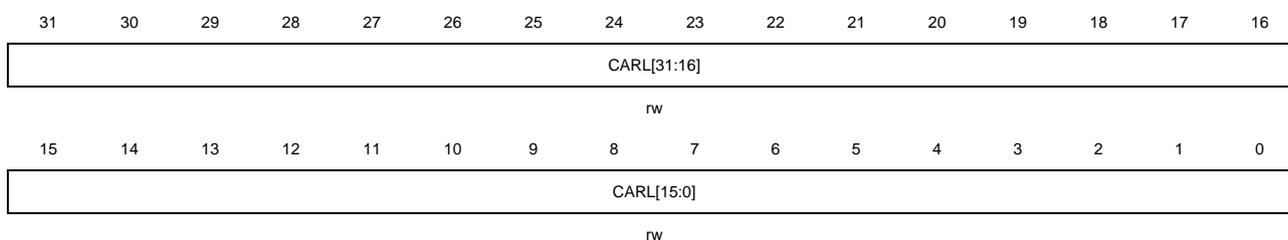
Bits	Fields	Descriptions
15:0	PSC[15:0]	Prescaler value of the counter clock The PSC clock is divided by (PSC+1) to generate the counter clock. The value of this bit-field will be loaded to the corresponding shadow register at every update event.

### Counter auto reload register (TIMERx\_CAR) (x=1)

Address offset: 0x2C

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



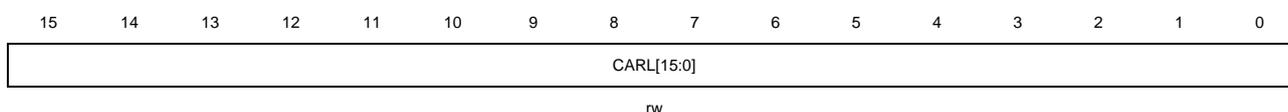
Bits	Fields	Descriptions
31:0	CARL[31:0]	Counter auto reload value This bit-field specifies the auto reload value of the counter.

### Counter auto reload register (TIMERx\_CAR) (x=2)

Address offset: 0x2C

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



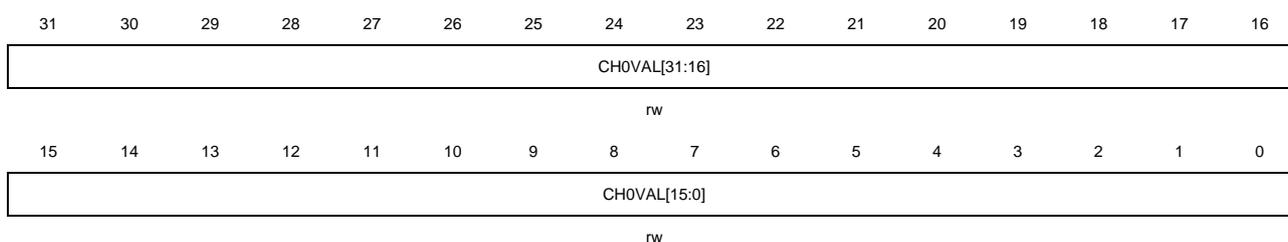
Bits	Fields	Descriptions
15:0	CARL[15:0]	Counter auto reload value This bit-field specifies the auto reload value of the counter.

### Channel 0 capture/compare value register (TIMERx\_CH0CV) (x=1)

Address offset: 0x34

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



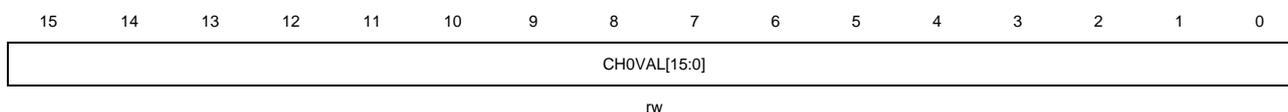
Bits	Fields	Descriptions
31:0	CH0VAL[31:0]	Capture or compare value of channel0 When channel 0 is configured in input mode, this bit-filed indicates the counter value corresponding to the last capture event. And this bit-filed is read-only. When channel 0 is configured in output mode, this bit-filed contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates every update event.

### Channel 0 capture/compare value register (TIMERx\_CH0CV) (x=2)

Address offset: 0x34

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



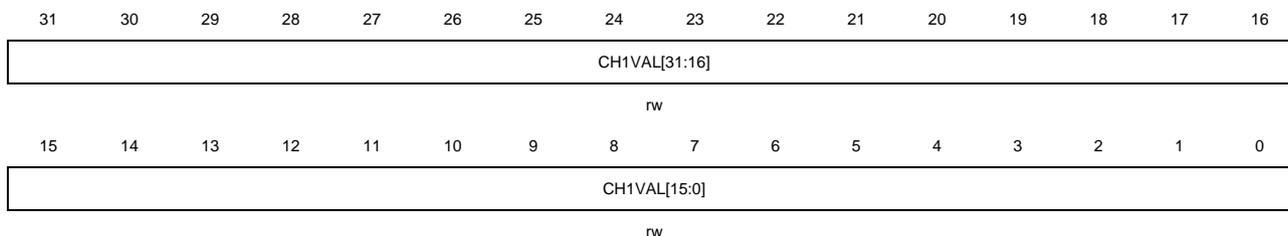
Bits	Fields	Descriptions
15:0	CH0VAL[15:0]	Capture or compare value of channel0 When channel 0 is configured in input mode, this bit-filed indicates the counter value corresponding to the last capture event. And this bit-filed is read-only. When channel 0 is configured in output mode, this bit-filed contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates every update event.

### Channel 1 capture/compare value register (TIMERx\_CH1CV) (x=1)

Address offset: 0x38

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



Bits	Fields	Descriptions
31:0	CH1VAL[31:0]	Capture or compare value of channel1 When channel 1 is configured in input mode, this bit-filed indicates the counter

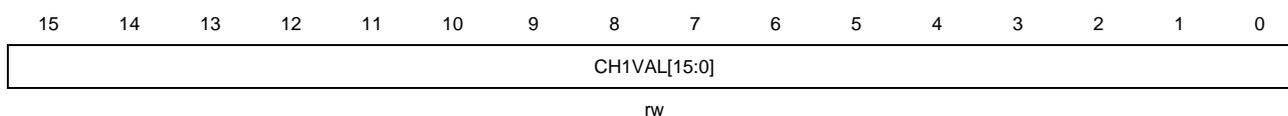
value corresponding to the last capture event. And this bit-field is read-only.  
 When channel 1 is configured in output mode, this bit-field contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates every update event.

### Channel 1 capture/compare value register (TIMERx\_CH1CV) (x=2)

Address offset: 0x38

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



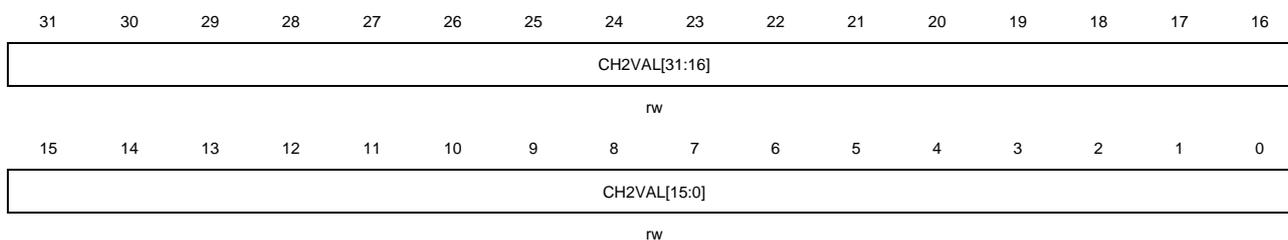
Bits	Fields	Descriptions
15:0	CH1VAL[15:0]	Capture or compare value of channel1  When channel 1 is configured in input mode, this bit-field indicates the counter value corresponding to the last capture event. And this bit-field is read-only.  When channel 1 is configured in output mode, this bit-field contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates every update event.

### Channel 2 capture/compare value register (TIMERx\_CH2CV) (x=1)

Address offset: 0x3C

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



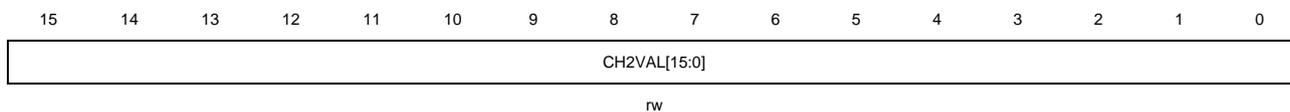
Bits	Fields	Descriptions
31:0	CH2VAL[31:0]	Capture or compare value of channel 2  When channel 2 is configured in input mode, this bit-field indicates the counter value corresponding to the last capture event. And this bit-field is read-only.  When channel 2 is configured in output mode, this bit-field contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates every update event.

## Channel 2 capture/compare value register (TIMERx\_CH2CV) (x=2)

Address offset: 0x3C

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



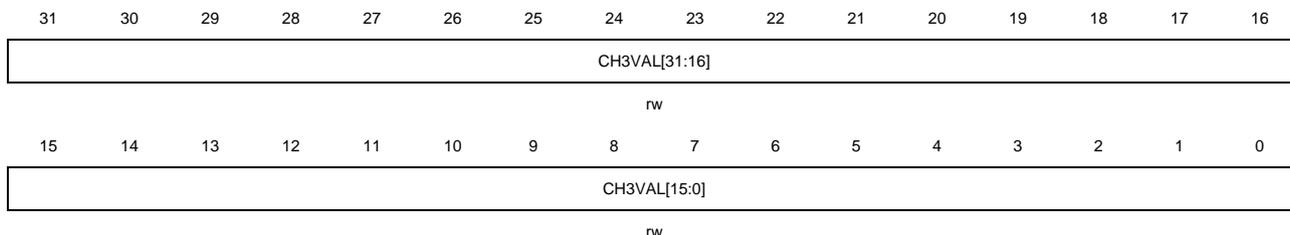
Bits	Fields	Descriptions
15:0	CH2VAL[15:0]	<p>Capture or compare value of channel 2</p> <p>When channel 2 is configured in input mode, this bit-filed indicates the counter value corresponding to the last capture event. And this bit-filed is read-only.</p> <p>When channel 2 is configured in output mode, this bit-filed contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates every update event.</p>

## Channel 3 capture/compare value register (TIMERx\_CH3CV) (x=1)

Address offset: 0x40

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



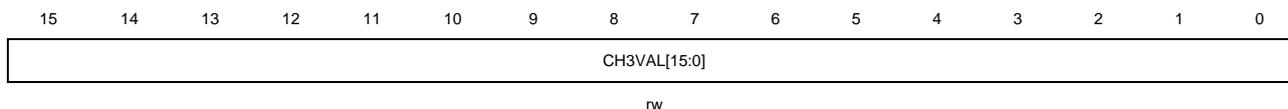
Bits	Fields	Descriptions
31:0	CH3VAL[31:0]	<p>Capture or compare value of channel 3</p> <p>When channel3 is configured in input mode, this bit-filed indicates the counter value corresponding to the last capture event. And this bit-filed is read-only.</p> <p>When channel 3 is configured in output mode, this bit-filed contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates every update event.</p>

## Channel 3 capture/compare value register (TIMERx\_CH3CV) (x=2)

Address offset: 0x40

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



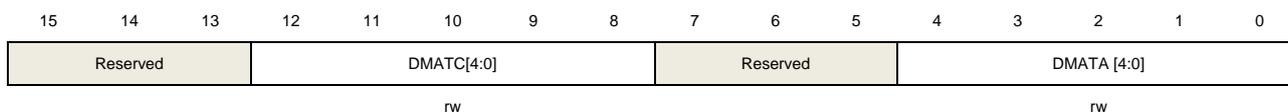
Bits	Fields	Descriptions
15:0	CH3VAL[15:0]	Capture or compare value of channel 3 When channel3 is configured in input mode, this bit-field indicates the counter value corresponding to the last capture event. And this bit-field is read-only. When channel 3 is configured in output mode, this bit-field contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates every update event.

### DMA configuration register (TIMERx\_DMACFG)

Address offset: 0x48

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



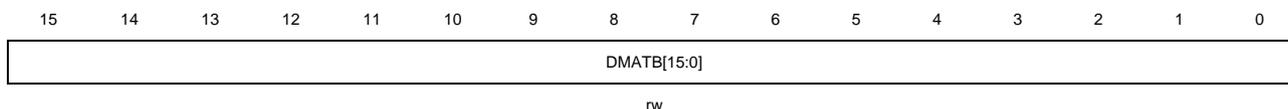
Bits	Fields	Descriptions
15:13	Reserved	Must be kept at reset value.
12:8	DMATC [4:0]	DMA transfer count This field is defined the number of DMA will access(R/W) the register of TIMERx_DMATB
7:5	Reserved	Must be kept at reset value.
4:0	DMATA [4:0]	DMA transfer access start address This field define the first address for the DMA access the TIMERx_DMATB. When access is done through the TIMERx_DMA address first time, this bit-field specifies the address you just access. And then the second access to the TIMERx_DMATB, you will access the address of start address + 0x4.  5'b0_0000: TIMERx_CTL0 5'b0_0001: TIMERx_CTL1 ... In a word: Start Address = TIMERx_CTL0 + DMASAR*4

## DMA transfer buffer register (TIMERx\_DMATB)

Address offset: 0x4C

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



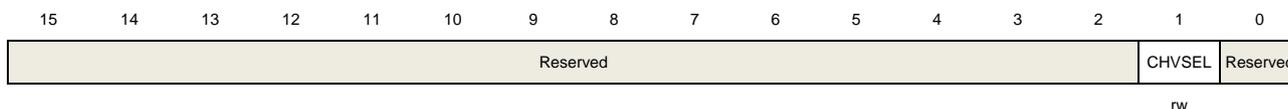
Bits	Fields	Descriptions
15:0	DMATB[15:0]	<p>DMA transfer buffer</p> <p>When a read or write operation is assigned to this register, the register located at the address range (Start Addr + Transfer Timer* 4) will be accessed.</p> <p>The transfer Timer is calculated by hardware, and ranges from 0 to DMATC.</p>

## Configuration register (TIMERx\_CFG ) of GD32F170xx and GD32F190xx devices

Address offset: 0xFC

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



Bits	Fields	Descriptions
15:2	Reserved	Must be kept at reset value
1	CHVSEL	<p>Write CHxVAL register selection</p> <p>This bit-field set and reset by software.</p> <p>1: If write the CHxVAL register, the write value is same as the CHxVAL value, the write access ignored</p> <p>0: No effect</p>
0	Reserved	Must be kept at reset value

## 15.3. General level2 timer (TIMERx, x=13)

### 15.3.1. Overview

The general level2 timer module (TIMER 13) is a one-channel timer that supports input capture, output compare. They can generate PWM signals to control motor or be used for power management applications. The general level2 time reference is a 16-bit counter that can be used as an unsigned counter.

In addition, the general level2 timers can be programmed and be used to count or time external events that drive other timers.

### 15.3.2. Characteristics

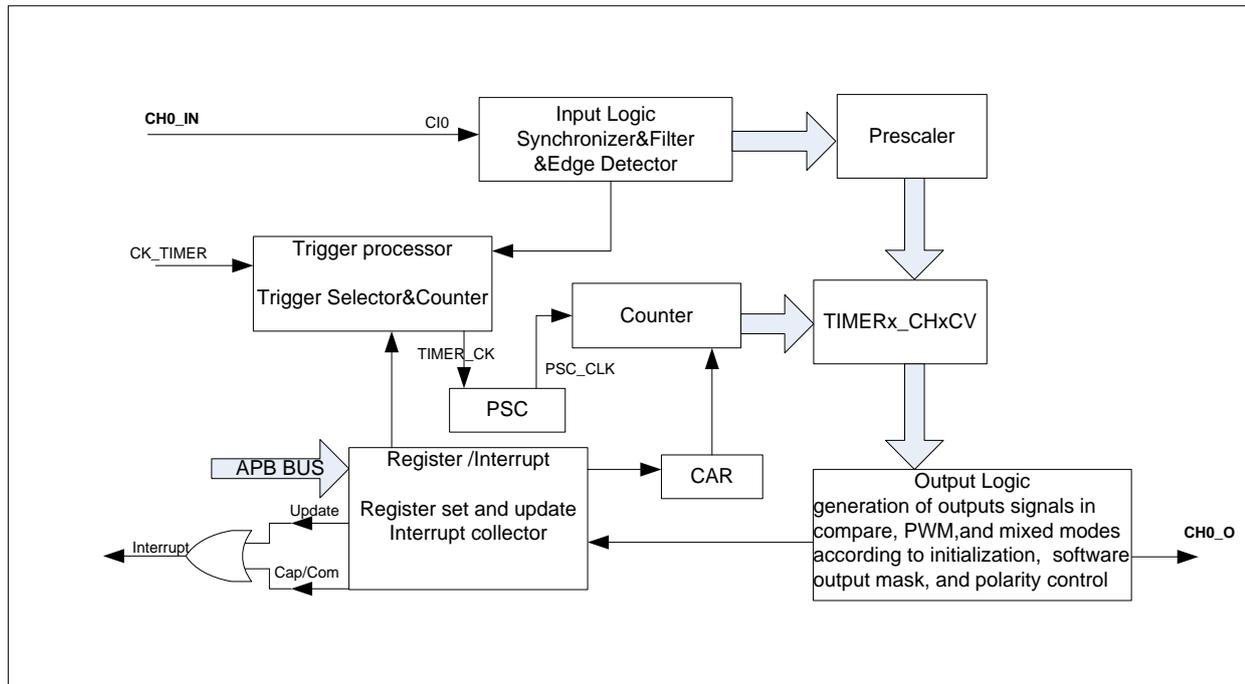
- Total channel num: 1.
- Counter width: 16bit.
- Source of count clock: internal clock.
- Counter mode: count up only.
- Programmable prescaler: 16 bit. Factor can be changed on the go.
- Each channel is user-configurable:  
Input capture mode, output compare mode, programmable and PWM mode.
- Auto-reload function.
- Interrupt output on: update, compare/capture event.

### 15.3.3. Block diagram

[Figure 15-50. General level2 timer block diagram](#) provides details on the internal

configuration of the general level2 timer.

**Figure 15-50. General level2 timer block diagram**



### 15.3.4. Function overview

#### Clock selection

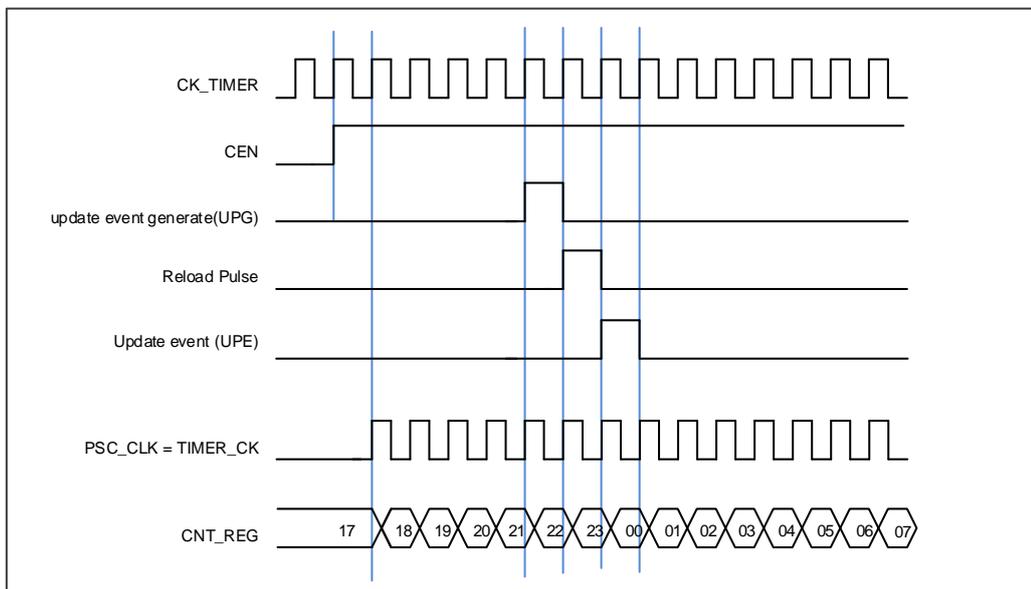
The general level2 TIMER can only being clocked by the CK\_TIMER.

- Internal timer clock CK\_TIMER which is from module RCU

The general level2 TIMER has only one clock source which is the internal CK\_TIMER, used to drive the counter prescaler. When the CEN is set, the CK\_TIMER will be divided by PSC value to generate PSC\_CLK.

The TIMER\_CK, driven counter's prescaler to count, is equal to CK\_TIMER which is from RCU

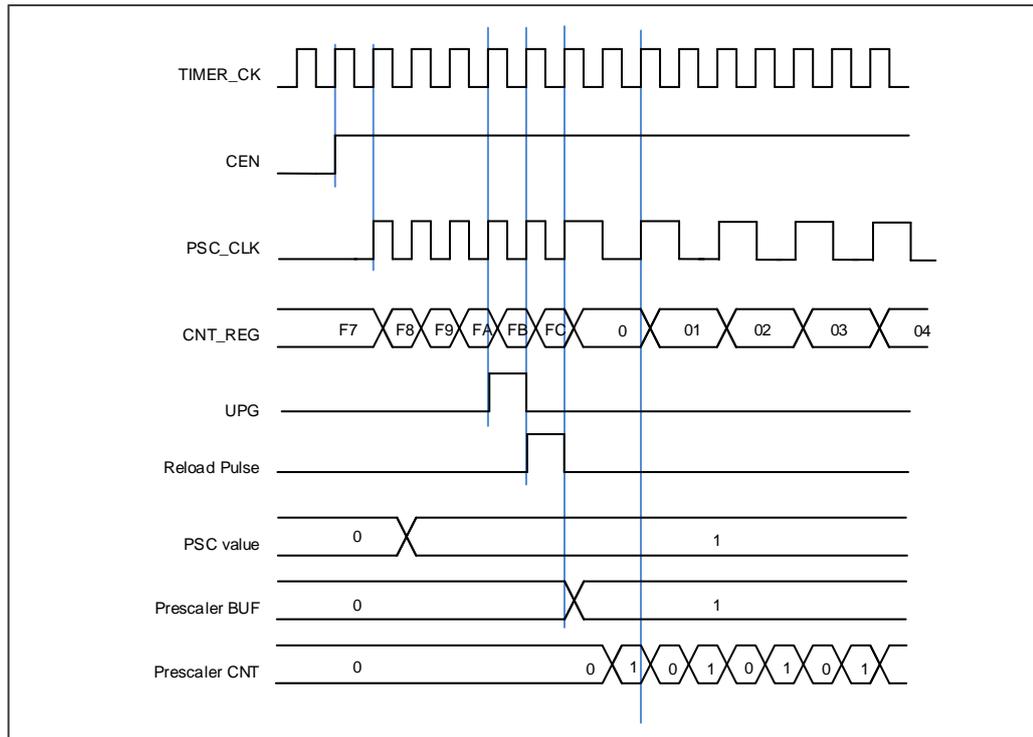
**Figure 15-51. Normal mode, internal clock divided by 1**



#### Prescaler

The prescaler can divide the timer clock (TIMER\_CK) to the counter clock (PSC\_CLK) by any factor between 1 and 65536. It is controlled through prescaler register (TIMERx\_PSC) which can be changed on the go but be taken into account at the next update event.

**Figure 15-52. Counter timing diagram with prescaler division change from 1 to 2**



### Up counting mode

In this mode, the counter counts up continuously from 0 to the counter-reload value, which is defined in the `TIMERx_CAR` register, in a count-up direction. Once the counter reaches the counter reload value, the counter restarts to count once again from 0. The update event is generated at each counter overflow. The counting direction bit `DIR` in the `TIMERx_CTL1` register should be set to 0 for the up counting mode.

When the update event is set by the `UPG` bit in the `TIMERx_SWEVG` register, the counter value will be initialized to 0 and generates an update event.

If the `UPDIS` bit in `TIMERx_CTL0` register is set, the update event is disabled.

When an update event occurs, all the registers (repetition counter, auto reload register, prescaler register) are updated.

The following figures show some examples of the counter behavior for different clock prescaler factor when `TIMERx_CAR=0x63`.

Figure 15-53. Up-counter timechart, PSC=0/1

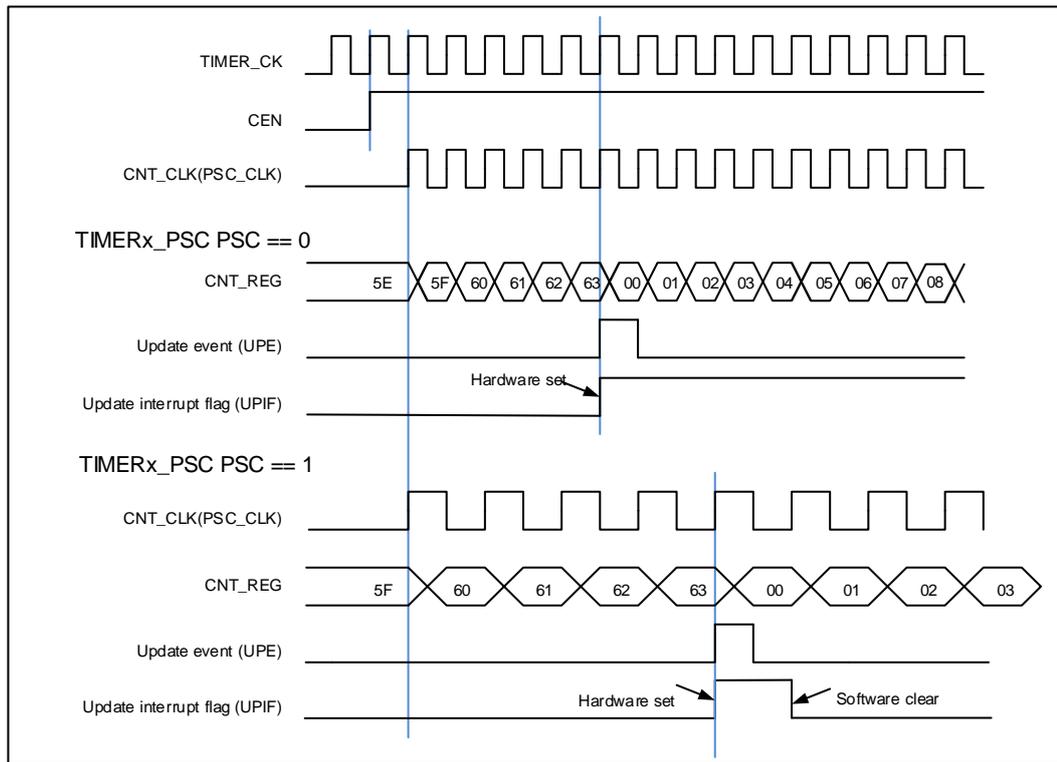
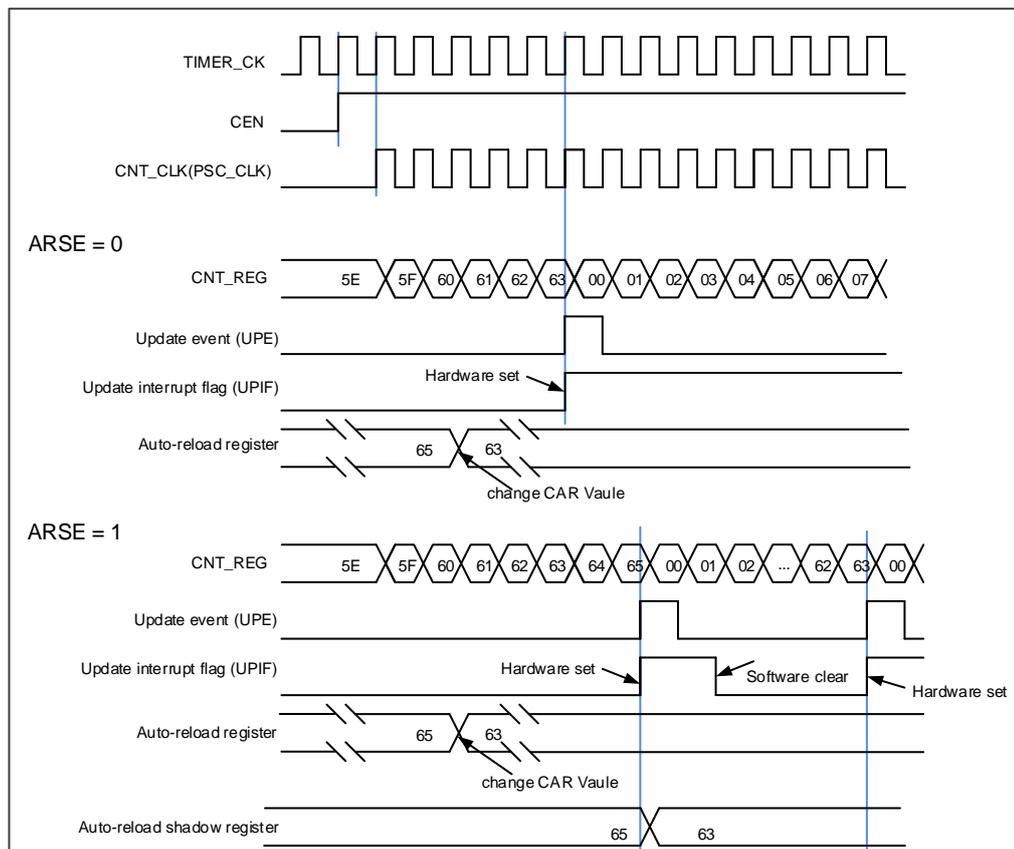


Figure 15-54. Up-counter timechart, change TIMERx\_CAR on the go



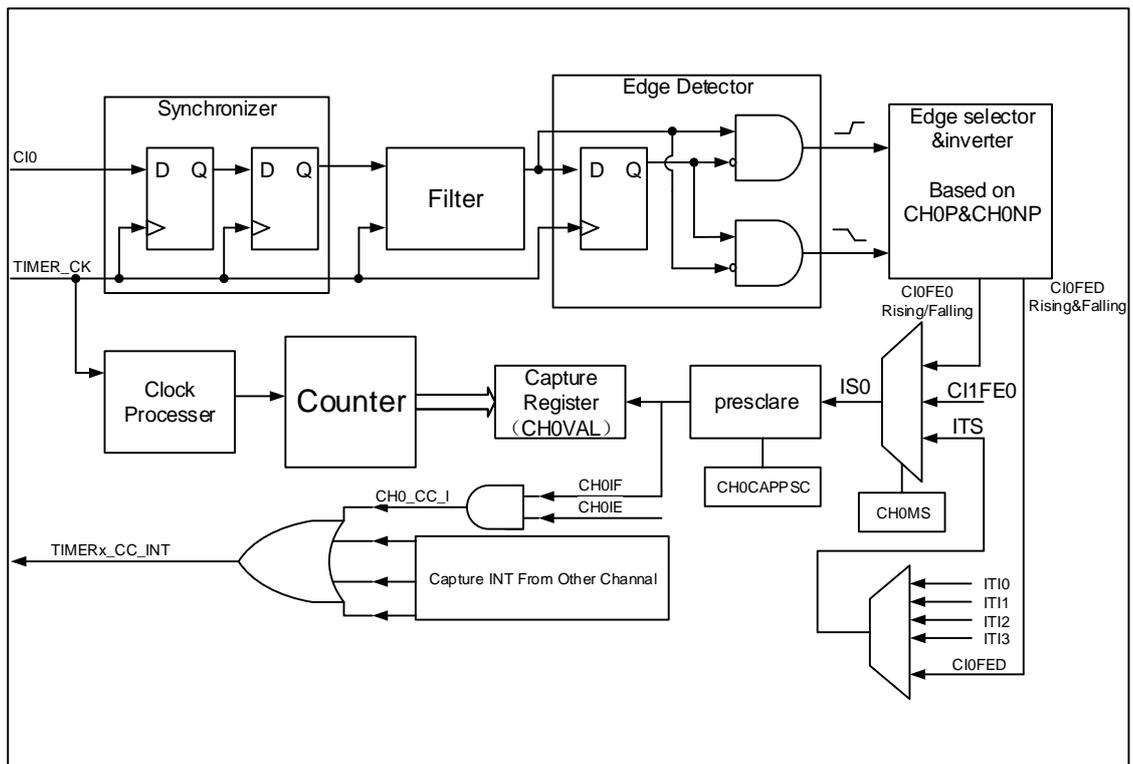
### Capture/compare channels

The general level2 timer has one independent channel which can be used as capture inputs or compare match outputs. Each channel is built around a channel capture compare register including an input stage, channel controller and an output stage.

■ **Input capture mode**

Capture mode allows the channel to perform measurements such as pulse timing, frequency, period, duty cycle and so on. The input stage consists of a digital filter, a channel polarity selection, edge detection and a channel prescaler. When a selected edge occurs on the channel input, the current value of the counter is captured into the `TIMERx_CHxCV` register, at the same time the `CHxIF` bit is set and the channel interrupt is generated if enabled by `CHxIE = 1`.

**Figure 15-55. Input capture logic**



First, the channel input signal (`Cix`) is synchronized to `TIMER_CK` domain, and then sampled by a digital filter to generate a filtered input signal. Then through the edge detector, the rising and fall edge are detected. You can select one of them by `CHxP`. One more selector is for the other channel and trig, controlled by `CHxMS`. The `IC_presclare` make several the input event generate one effective capture event. On the capture event, `CHxVAL` will restore the value of Counter.

So the process can be divided to several steps as below:

**Step1:** Filter configuration. (`CHxCAPFLT` in `TIMERx_CHCTL0`)

Based on the input signal and requested signal quality, configure compatible CHxCAPFLT.

**Step2:** Edge selection. (CHxP/CHxNP in TIMERx\_CHCTL2)

Rising or falling edge, choose one by CHxP/CHxNP.

**Step3:** Capture source selection. (CHxMS in TIMERx\_CHCTL0)

As soon as you select one input capture source by CHxMS, you have set the channel to input mode (CHxMS!=0x0) and TIMERx\_CHxCV cannot be written any more.

**Step4:** Interrupt enable. (CHxIE in TIMERx\_DMAINTEN)

Enable the related interrupt enable; you can get the interrupt.

**Step5:** Capture enables. (CHxEN in TIMERx\_CHCTL2)

**Result:** When you wanted input signal is got, TIMERx\_CHxCV will be set by Counter's value. And CHxIF is asserted. If the CHxIF is high, the CHxOF will be asserted also. The interrupt will be asserted based on the your configuration of CHxIE in TIMERx\_DMAINTEN

**Direct generation:** If you want to generate a DMA request or Interrupt, you can set CHxG by software directly.

## ■ Output compare mode

In Output Compare mode, the TIMERx can generate timed pulses with programmable position, polarity, duration, and frequency. When the counter matches the value in the CHxVAL register of an output compare channel, the channel (n) output can be set, cleared, or toggled based on CHxCOMCTL. when the counter reaches the value in the CHxVAL register, the CHxIF bit is set and the channel (n) interrupt is generated if CHxIE = 1.

So the process can be divided to several steps as below:

**Step1:** Clock configuration. Such as clock source, clock prescaler and so on.

**Step2:** Compare mode configuration.

- \* Set the shadow enable mode by CHxCOMSEN
- \* Set the output mode (Set/Clear/Toggle) by CHxCOMCTL.
- \* Select the active high polarity by CHxP/CHxNP
- \* Enable the output by CHxEN

**Step3:** Interrupt/DMA-request enables configuration by CHxIE

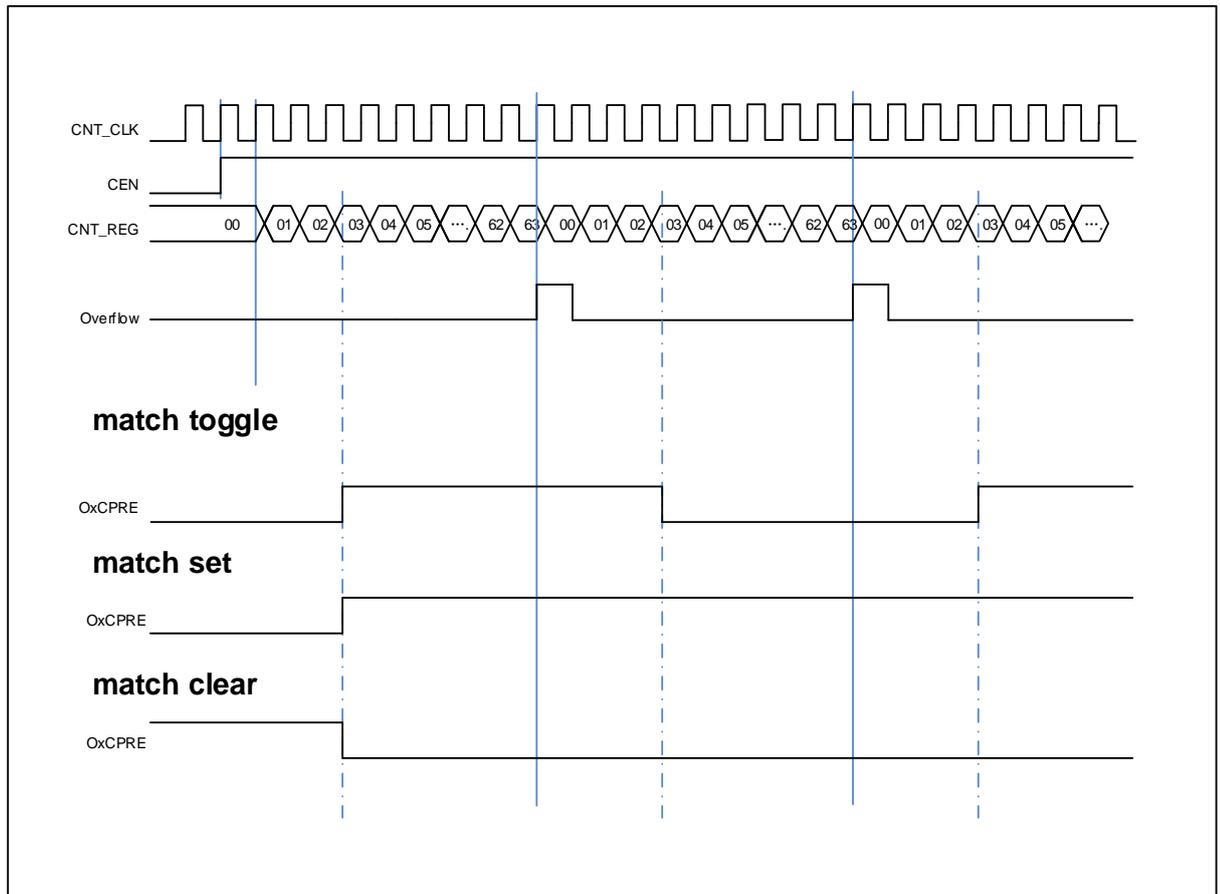
**Step4:** Compare output timing configuration by TIMERx\_CAR and TIMERx\_CHxCV.

About the CHxVAL, you can change it on the go to meet the waveform you expected.

**Step5:** Start the counter by CEN.

The timechart [Figure 15-56. Output-compare under three modes](#) show the three compare modes toggle/set/clear. CAR=0x63, CHxVAL=0x3

Figure 15-56. Output-compare under three modes



### PWM mode

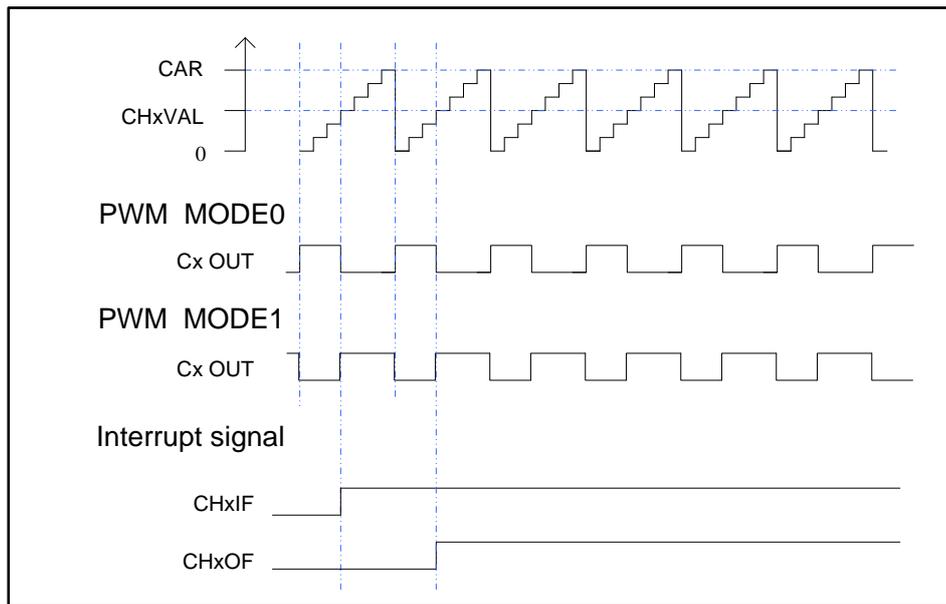
In the output PWM mode (by setting the CHxCOMCTL bits to 3'b110 (PWM mode0) or to 3'b111(PWM mode1), the channel can generate PWM waveform according to the TIMERx\_CAR registers and TIMERx\_CHxCV registers.

The period is determined by TIMERx\_CAR and duty cycle is determined by TIMERx\_CHxCV. [Figure 15-57. PWM mode timechart](#) shows the PWM output mode and interrupts waveform.

If TIMERx\_CHxCV is greater than TIMERx\_CAR, the output will be always active under PWM mode0 (CHxCOMCTL==3'b110).

And if TIMERx\_CHxCV is equal to zero, the output will be always inactive under PWM mode0 (CHxCOMCTL==3'b110).

**Figure 15-57. PWM mode timechart**



**Channel output reference signal**

When the TIMERx is used in the compare match output mode, the OxCPRE signal (Channel x Output prepare signal) is defined by setting the CHxCOMCTL filed. The OxCPRE signal has several types of output function. These include, keeping the original level by setting the CHxCOMCTL field to 0x00, set to 1 by setting the CHxCOMCTL field to 0x01, set to 0 by setting the CHxCOMCTL field to 0x02 or signal toggle by setting the CHxCOMCTL field to 0x03 when the counter value matches the content of the TIMERx\_CHxCV register.

The PWM mode 0 and PWM mode 1 outputs are also another kind of OxCPRE output which is setup by setting the CHxCOMCTL field to 0x06/0x07. In these modes, the OxCPRE signal level is changed according to the counting direction and the relationship between the counter value and the TIMERx\_CHxCV content. With regard to a more detail description refer to the relative bit definition.

Another special function of the OxCPRE signal is a forced output which can be achieved by setting the CHxCOMCTL field to 0x04/0x05. Here the output can be forced to an inactive/active level irrespective of the comparison condition between the counter and the TIMERx\_CHxCV values.

**Timer debug mode**

When the Cortex™-M3 halted, and the TIMERx\_HOLD configuration bit in DBG\_CTL0 register set to 1, the TIMERx counter stops.

### 15.3.5. TIMERx registers(x=13)

#### Control register 0 (TIMERx\_CTL0)

Address offset: 0x00

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CKDIV[1:0]	ARSE	Reserved					UPS	UPDIS	CEN
						rw	rw						rw	rw	rw

Bits	Fields	Descriptions
15:10	Reserved	Must be kept at reset value
9:8	CKDIV[1:0]	Clock division The CKDIV bits can be configured by software to specify division ratio between the timer clock (TIMER_CK) and the dead-time and sampling clock (DTS), which is used by the dead-time generators and the digital filters. 00: $f_{DTS}=f_{TIMER\_CK}$ 01: $f_{DTS}= f_{TIMER\_CK} /2$ 10: $f_{DTS}= f_{TIMER\_CK} /4$ 11: Reserved
7	ARSE	Auto-reload shadow enable 0: The shadow register for TIMERx_CAR register is disabled 1: The shadow register for TIMERx_CAR register is enabled
6:3	Reserved	Must be kept at reset value
2	UPS	Update source This bit is used to select the update event sources by software. 0: When enabled, any of the following events generate an update interrupt or DMA request: <ul style="list-style-type: none"> <li>- The UPG bit is set</li> <li>- The counter generates an overflow or underflow event</li> <li>- The slave mode controller generates an update event.</li> </ul> 1: When enabled, only counter overflow/underflow generates an update interrupt or DMA request.
1	UPDIS	Update disable. This bit is used to enable or disable the update event generation. 0: update event enable. The update event is generate and the buffered registers are loaded with their preloaded values when one of the following events occurs: <ul style="list-style-type: none"> <li>- The UPG bit is set</li> </ul>

- The counter generates an overflow or underflow event
- The slave mode controller generates an update event.

1: update event disable. The buffered registers keep their value, while the counter and the prescaler are reinitialized if the UG bit is set or if the slave mode controller generates a hardware reset event.

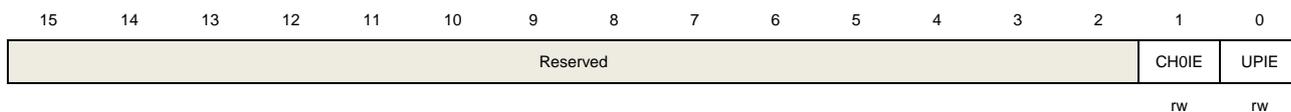
0            CEN            Counter enable  
 0: Counter disable  
 1: Counter enable  
 The CEN bit must be set by software when timer works in external clock, pause mode and encoder mode. While in event mode, the hardware can set the CEN bit automatically.

### Interrupt enable register (TIMERx\_DMAINTEN)

Address offset: 0x0C

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



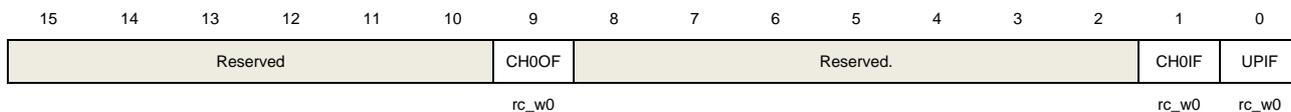
Bits	Fields	Descriptions
15:2	Reserved	Must be kept at reset value.
1	CH0IE	Channel 0 capture/compare interrupt enable 0: disabled 1: enabled
0	UPIE	Update interrupt enable 0: disabled 1: enabled

### Interrupt flag register (TIMERx\_INTF)

Address offset: 0x10

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



Bits	Fields	Descriptions
------	--------	--------------

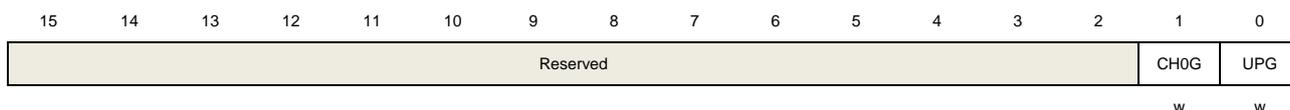
15:10	Reserved	Must be kept at reset value.
9	CH0OF	<p>Channel 0 over capture flag</p> <p>When channel 0 is configured in input mode, this flag is set by hardware when a capture event occurs while CH0IF flag has already been set. This flag is cleared by software.</p> <p>0: No over capture interrupt occurred</p> <p>1: Over capture interrupt occurred</p>
8:2	Reserved	Must be kept at reset value.
1	CH0IF	<p>Channel 0 's capture/compare interrupt flag</p> <p>This flag is set by hardware and cleared by software. When channel 0 is in input mode, this flag is set when a capture event occurs. When channel 0 is in output mode, this flag is set when a compare event occurs.</p> <p>0: No Channel 1 interrupt occurred</p> <p>1: Channel 1 interrupt occurred</p>
0	UPIF	<p>Update interrupt flag</p> <p>This bit is set by hardware on an update event and cleared by software.</p> <p>0: No update interrupt occurred</p> <p>1: Update interrupt occurred</p>

### Software event generation register (TIMERx\_SWEVG)

Address offset: 0x14

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



Bits	Fields	Descriptions
15:2	Reserved	Must be kept at reset value.
1	CH0G	<p>Channel 0's capture or compare event generation</p> <p>This bit is set by software in order to generate a capture or compare event in channel 0, it is automatically cleared by hardware. When this bit is set, the CH1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. In addition, if channel 1 is configured in input mode, the current value of the counter is captured in TIMERx_CH0CV register, and the CH0OF flag is set if the CH0IF flag was already high.</p> <p>0: No generate a channel 1 capture or compare event</p> <p>1: Generate a channel 1 capture or compare event</p>

0	UPG	<p>This bit can be set by software, and cleared by hardware automatically. When this bit is set, the counter is cleared. The prescaler counter is cleared at the same time.</p> <p>0: No generate an update event 1: Generate an update event</p>
---	-----	---

### Channel control register 0 (TIMERx\_CHCTL0)

Address offset: 0x18

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved.								Reserved	CH0COMCTL[2:0]		CH0COM SEN	CH0COM FEN	CH0MS[1:0]		
								CH0CAPFLT[3:0]			CH0CAPPSC[1:0]				
								rw		rw		rw			

#### Output compare mode:

Bits	Fields	Descriptions
15:7	Reserved	Must be kept at reset value.
6:4	CH0COMCTL[2:0]	<p>Channel 0 compare output control</p> <p>This bit-field controls the behavior of the output reference signal O0CPRE which drives CH0_O and CH0_ON. O0CPRE is active high, while CH0_O and CH0_ON active level depends on CH0P and CH0NP bits.</p> <p>000: Frozen. The O0CPRE signal keeps stable, independent of the comparison between the register TIMERx_CH0CV and the counter TIMERx_CNT.</p> <p>001: Set the channel output. O0CPRE signal is forced high when the counter matches the output compare register TIMERx_CH0CV.</p> <p>010: Clear the channel output. O0CPRE signal is forced low when the counter matches the output compare register TIMERx_CH0CV.</p> <p>011: Toggle on match. O0CPRE toggles when the counter matches the output compare register TIMERx_CH0CV.</p> <p>100: Force low. O0CPRE is forced low level.</p> <p>101: Force high. O0CPRE is forced high level.</p> <p>110: PWM mode0. When counting up, O0CPRE is active as long as the counter is smaller than TIMERx_CH0CV else inactive. When counting down, O0CPRE is inactive as long as the counter is larger than TIMERx_CH0CV else active.</p> <p>111: PWM mode1. When counting up, O0CPRE is inactive as long as the counter is smaller than TIMERx_CH0CV else active. When counting down, O0CPRE is active as long as the counter is larger than TIMERx_CH0CV else inactive.</p> <p>When configured in PWM mode, the O0CPRE level changes only when the output compare mode switches from “frozen” mode to “PWM” mode or when the result of the comparison changes.</p> <p>This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is</p>

11 and CH0MS bit-filed is 00(COMPARE MODE).

- |     |            |   |
|-----|------------|---|
| 3   | CH0COMSEN  | <p>Channel 0 compare output shadow enable</p> <p>When this bit is set, the shadow register of TIMERx_CH0CV register, which updates at each update event, will be enabled.</p> <p>0: Channel 0 output compare shadow disable<br/>1: Channel 0 output compare shadow enable</p> <p>The PWM mode can be used without validating the shadow register only in single pulse mode (SPM bit in TIMERx_CTL0 register is set).</p> <p>This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 and CH0MS bit-filed is 00.</p>  |
| 2   | CH0COMFEN  | <p>Channel 0 output compare fast enable</p> <p>When this bit is set, the effect of an event on the trigger in input on the capture/compare output will be accelerated if the channel is configured in PWM0 or PWM1 mode. The output channel will treat an active edge on the trigger input as a compare match, and CH0_O is set to the compare level independently from the result of the comparison.</p> <p>0: Channel 0 output quickly compare disable. The minimum delay from an edge on the trigger input to activate CH0_O output is 5 clock cycles.<br/>1: Channel 0 output quickly compare enable. The minimum delay from an edge on the trigger input to activate CH0_O output is 3 clock cycles.</p> |
| 1:0 | CH0MS[1:0] | <p>Channel 0 I/O mode selection</p> <p>This bit-field specifies the work mode of the channel and the input signal selection. This bit-field is writable only when the channel is not active. (CH0EN bit in TIMERx_CHCTL2 register is reset).).</p> <p>00: Channel 0 is configured as output<br/>01: Channel 0 is configured as input, IS0 is connected to CI0FE0<br/>10: Channel 0 is configured as input, IS0 is connected to CI1FE0<br/>11: Channel 0 is configured as input, IS0 is connected to ITS. This mode is working only if an internal trigger input is selected through TRGS bits in TIMERx_SMCFG register.</p>   |

### Input capture mode:

Bits	Fields	Descriptions
15:8	Reserved	Must be kept at reset value.
7:4	CH0CAPFLT[3:0]	<p>Channel 0 input capture filter control</p> <p>An event counter is used in the digital filter, in which a transition on the output occurs after N input events. This bit-field specifies the frequency used to sample CI0 input signal and the length of the digital filter applied to CI0.</p> <p>0000: Filter disabled, <math>f_{SAMP}=f_{DTS}</math>, <math>N=1</math> 0001: <math>f_{SAMP}=f_{TIMER\_CK}</math>, <math>N=2</math> 0010: <math>f_{SAMP}=f_{TIMER\_CK}</math>, <math>N=4</math></p>

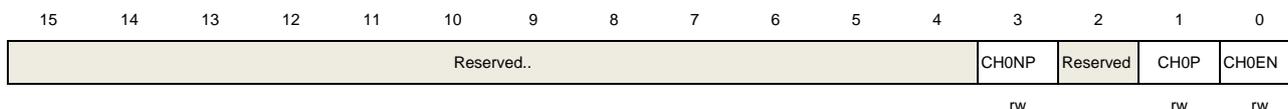
- 0011:  $f_{SAMP} = f_{TIMER\_CK}, N=8$
  - 0100:  $f_{SAMP} = f_{DTS}/2, N=6$
  - 0101:  $f_{SAMP} = f_{DTS}/2, N=8$
  - 0110:  $f_{SAMP} = f_{DTS}/4, N=6$
  - 0111:  $f_{SAMP} = f_{DTS}/4, N=8$
  - 1000:  $f_{SAMP} = f_{DTS}/8, N=6$
  - 1001:  $f_{SAMP} = f_{DTS}/8, N=8$
  - 1010:  $f_{SAMP} = f_{DTS}/16, N=5$
  - 1011:  $f_{SAMP} = f_{DTS}/16, N=6$
  - 1100:  $f_{SAMP} = f_{DTS}/16, N=8$
  - 1101:  $f_{SAMP} = f_{DTS}/32, N=5$
  - 1110:  $f_{SAMP} = f_{DTS}/32, N=6$
  - 1111:  $f_{SAMP} = f_{DTS}/32, N=8$
- 3:2      CH0CAPPSC[1:0]      Channel 0 input capture prescaler  
 This bit-field specifies the factor of the prescaler on channel 0 input. The prescaler is reset when CH0EN bit in TIMEx\_CHCTL2 register is clear.  
 00: Prescaler disable, capture is done on each channel input edge  
 01: Capture is done every 2 channel input edges  
 10: Capture is done every 4 channel input edges  
 11: Capture is done every 8 channel input edges
- 1:0      CH0MS[1:0]      Channel 0 mode selection  
 Same as output compare mode

## Channel control register 2 (TIMEx\_CHCTL2)

Address offset: 0x20

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word(32-bit)



Bits	Fields	Descriptions
15:4	Reserved	Must be kept at reset value
3	CH0NP	Channel 0 complementary output polarity When channel 0 is configured in output mode, this bit specifies the complementary output signal polarity. 0: Channel 0 active high 1: Channel 0 active low When channel 0 is configured in input mode, In conjunction with CH0P, this bit is used to define the polarity of CI0.

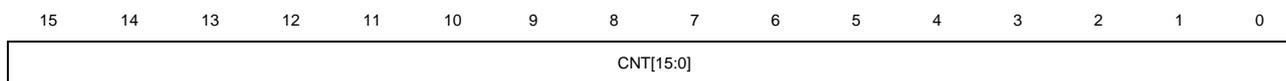
		This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 or 10.
2	Reserved	Must be kept at reset value
1	CH0P	<p>Channel 0 capture/compare polarity</p> <p>When channel 0 is configured in output mode, this bit specifies the output signal polarity.</p> <p>0: Channel 0 active high</p> <p>1: Channel 0 active low</p> <p>When channel 0 is configured in input mode, this bit specifies the CIO signal polarity. [CH0NP, CH0P] will select the active trigger or capture polarity for CIOFE0 or C11FE0. [CH0NP==0, CH0P==0]: C1xFE0's rising edge is the active signal for capture or trigger operation in slave mode. And C1xFE0 will not be inverted.</p> <p>[CH0NP==0, CH0P==1]: C1xFE0's falling edge is the active signal for capture or trigger operation in slave mode. And C1xFE0 will be inverted.</p> <p>[CH0NP==1, CH0P==0]: Reserved.</p> <p>[CH0NP==1, CH0P==1]: C1xFE0's falling and rising edge are both the active signal for capture or trigger operation in slave mode. And C1xFE0 will be not inverted.</p> <p>This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 or 10.</p>
0	CH0EN	<p>Channel 0 capture/compare function enable</p> <p>When channel 0 is configured in input mode, setting this bit enables CH0_O signal in active state. When channel 0 is configured in output mode, setting this bit enables the capture event in channel0.</p> <p>0: Channel 0 disabled</p> <p>1: Channel 0 enabled</p>

### Counter register (TIMERx\_CNT)

Address offset: 0x24

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



rw

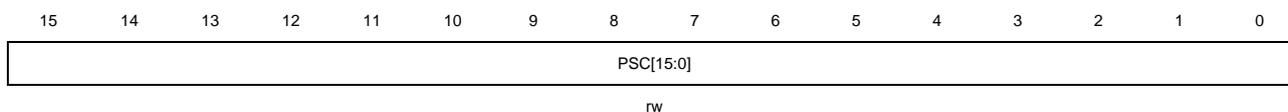
Bits	Fields	Descriptions
15:0	CNT[15:0]	This bit-filed indicates the current counter value. Writing to this bit-filed can change the value of the counter.

### Prescaler register (TIMERx\_PSC)

Address offset: 0x28

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



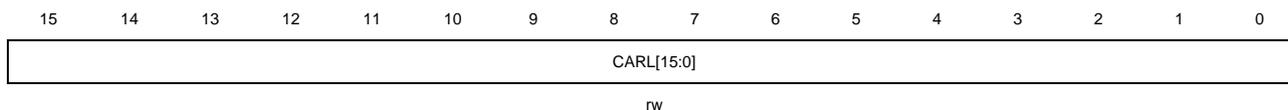
Bits	Fields	Descriptions
15:0	PSC[15:0]	Prescaler value of the counter clock The PSC clock is divided by (PSC+1) to generate the counter clock. The value of this bit-filed will be loaded to the corresponding shadow register at every update event.

### Counter auto reload register (TIMERx\_CAR)

Address offset: 0x2C

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



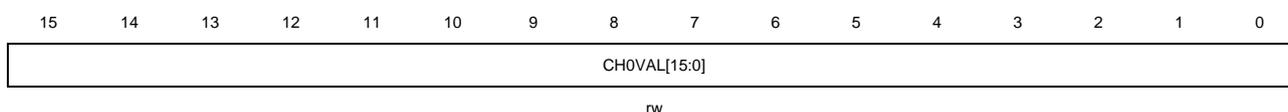
Bits	Fields	Descriptions
15:0	CARL[15:0]	Counter auto reload value This bit-filed specifies the auto reload value of the counter.

### Channel 0 capture/compare value register (TIMERx\_CH0CV)

Address offset: 0x34

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



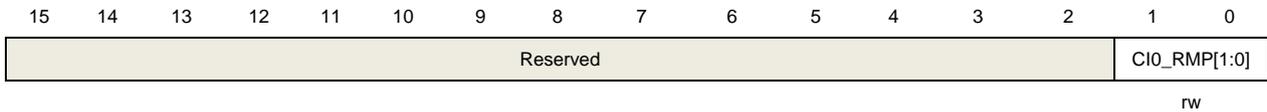
Bits	Fields	Descriptions
15:0	CH0VAL[15:0]	Capture or compare value of channel0 When channel 0 is configured in input mode, this bit-filed indicates the counter value corresponding to the last capture event. And this bit-filed is read-only. When channel 0 is configured in output mode, this bit-filed contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates every update event.

## Channel input remap register(TIMERx\_IRMP)

Address offset: 0x50

Reset value: 0x0000

This register can be accessed by half-word(16-bit) or word(32-bit)



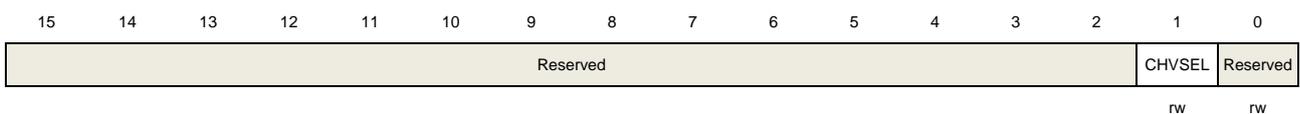
Bits	Fields	Descriptions
15:2	Reserved	Must be kept at reset value
1:0	CI0_RMP[1:0]	Channel 0 input remap 00: Channel 0 input is connected to GPIO(TIMER13_CH0) 01: Channel 0 input is connected to the RTCCLK 10: Channel 0 input is connected to HXTAL/32 clock 11: Channel 0 input is connected to CKOUTSEL(in GD32F130xx and GD32F150xx devices )/CKOUT0SEL(in GD32F170xx and GD32F190xx devices), which is controlled by RCU_CFG0.

## Configuration register (TIMERx\_CFG ) of GD32F170xx and GD32F190xx devices

Address offset: 0xFC

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



Bits	Fields	Descriptions
15:2	Reserved	Must be kept at reset value
1	CHVSEL	Write CHxVAL register selection This bit-field set and reset by software. 1: If write the CHxVAL register, the write value is same as the CHxVAL value, the write access ignored 0: No effect
0	Reserved	Must be kept at reset value

## 15.4. General level3 timer (TIMERx, x=14)

### 15.4.1. Overview

The general level3 timer module (TIMER14) is a two-channel timer that supports both input capture and output compare. They can generate PWM signals to control motor or be used for power management applications. The general level3 timer has a 16-bit counter that can be used as an unsigned counter.

In addition, the general level3timers can be programmed and be used for counting, their external events can be used to drive other timers.

Timer also includes a dead-time Insertion module which issuitable for motor control applications.

Timers are completely independent with each other, but they may be synchronized to provide a larger timer with their counters incrementing in unison.

### 15.4.2. Characteristics

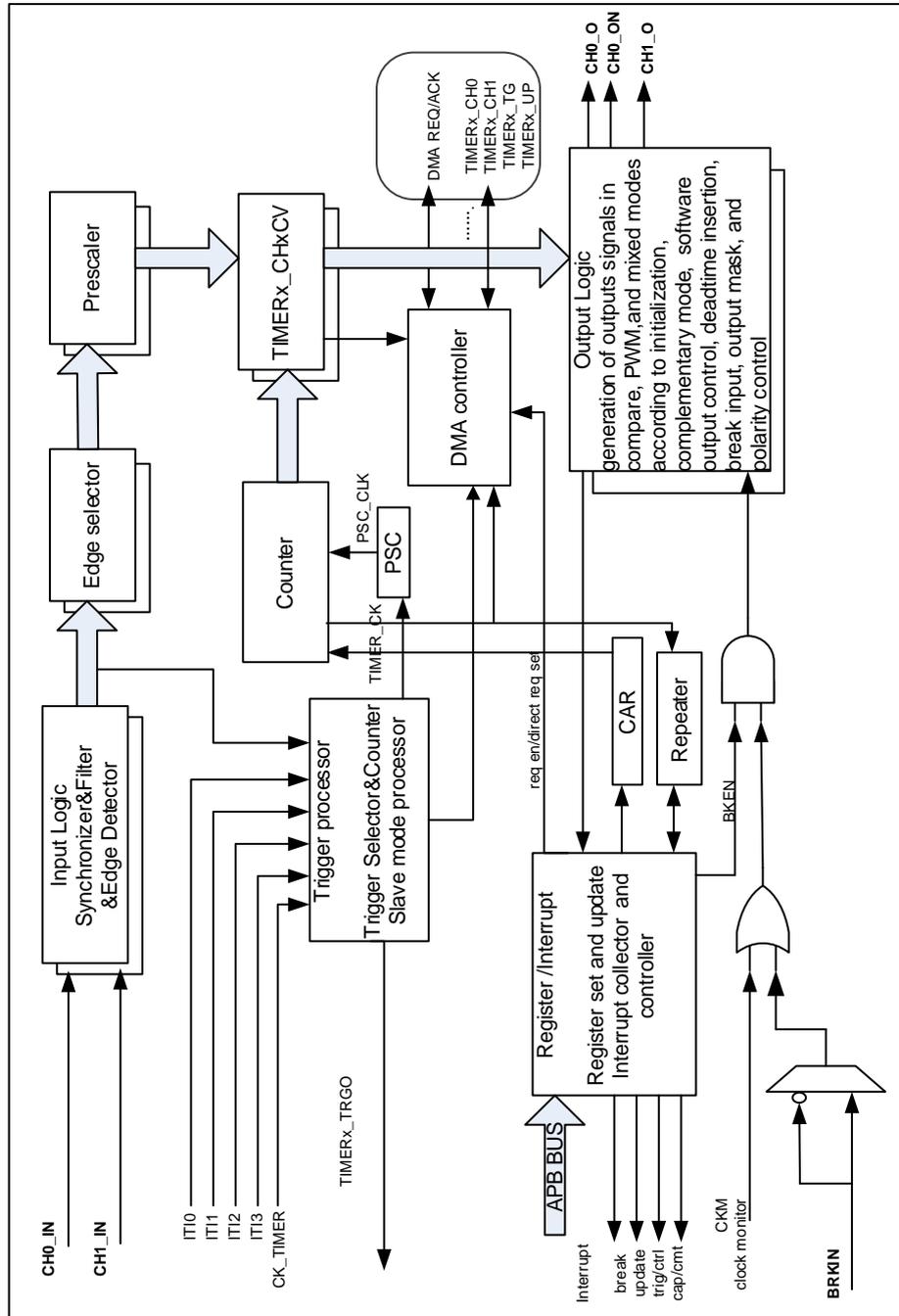
- Total channel num: 2.
- Counter width: 16 bit.
- Source of counter clock is selectable:  
internal clock, internal trigger, external input.
- Counter modes: count up only.
- Programmable prescaler: 16 bit.The factor can be changed on the go.
- Each channel is user-configurable:  
input capture mode, output compare mode, programmable PWM mode, single pulse mode
- Programmable dead time insertion.
- Auto reload function.
- Programmable counter repetition function.
- Break input.
- Interrupt output or DMA request on: update, trigger event, compare/capture event, and break input.
- Daisy chaining of timer modules allows a single timer to initiate multiple timers.
- Timer synchronization allows selected timers to start counting on the same clock cycle.
- Timer Master/Slave mode controller.

### 15.4.3. Block diagram

[Figure 15-58. General level3 timer block diagram](#) provides details of the internal

configuration of the general level3 timer.

Figure 15-58. General level3 timer block diagram



### 15.4.4. Function overview

#### Clock selection

The general level3 timer has the capability of being clocked by either the TIMER\_CK or an alternate clock source controlled by SMC (TIMERx\_SMCFG bit [2:0]).

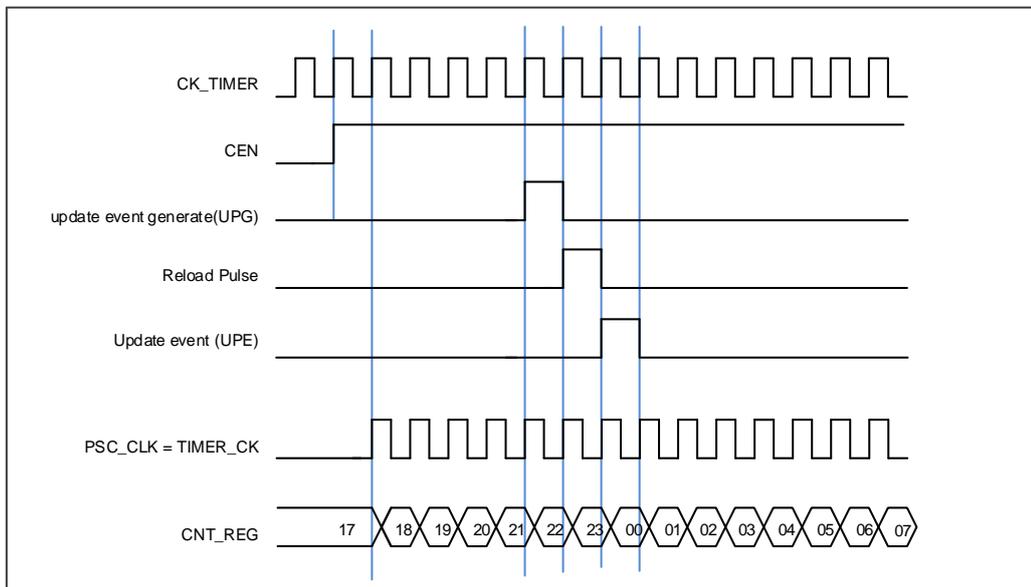
- SMC [2:0] == 3'b000. Internal clock CK\_TIMER is selected as timer clock source which is from module RCU.

The default clock source is the CK\_TIMER for driving the counter prescaler when the slave mode is disabled (SMC [2:0] == 3'b000). When the CEN is set, the CK\_TIMER will be divided by PSC value to generate PSC\_CLK.

In this mode, the TIMER\_CK, which drives counter's prescaler to count, is equal to CK\_TIMER which is from RCU.

If the slave mode controller is enabled by setting SMC [2:0] in the TIMERx\_SMCFG register to an available value 0x7, the prescaler is clocked by other clock sources selected by the TRGS [2:0] in the TIMERx\_SMCFG register, details as follows. When the slave mode selection bits SMC [2:0] are set to 0x4, 0x5 or 0x6, the internal clock TIMER\_CK is the counter prescaler driving clock source.

**Figure 15-59. Normal mode, internal clock divided by 1**



- SMC [2:0] == 3'b111 (external clock mode 0). External input pin is selected as timer clock source

The TIMER\_CK, which drives counter's prescaler to count, can be triggered by the event of rising or falling edge on the external pin TIMERx\_CH0/TIMERx\_CH1. This mode can be selected by setting SMC [2:0] to 0x7 and the TRGS [2:0] to 0x4, 0x5 or 0x6.

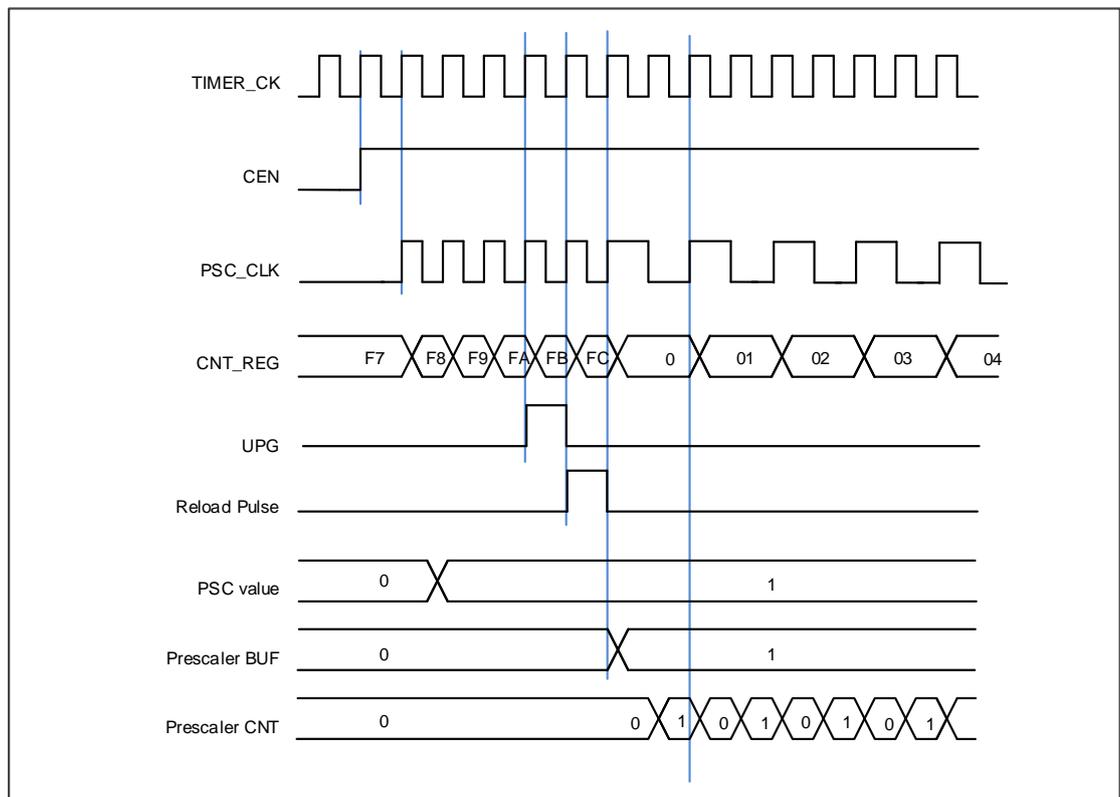
And, the counter prescaler can also be driven by rising edge on the internal trigger input pin ITI0/1/2/3. This mode can be selected by setting SMC [2:0] to 0x7 and the TRGS [2:0] to 0x0, 0x1, 0x2 or 0x3.

## Prescaler

The prescaler can divide the timer clock (TIMER\_CK) to a counter clock (PSC\_CLK) by any factor between 1 and 65536. It is controlled by prescaler register (TIMERx\_PSC) which can

be changed on the go but is taken into account at the next update event.

**Figure 15-60. Counter timing diagram with prescaler division change from 1 to 2**



### Up counting mode

In this mode, the counter counts up continuously from 0 to the counter-reload value, which is defined in the `TIMERx_CAR` register, in a count-up direction. Once the counter reaches the counter reload value, the counter restarts from 0. If the repetition counter is set, the update events will be generated after  $(\text{TIMERx\_CREP}+1)$  times of overflow. Otherwise the update event is generated each time when overflows. The counting direction bit `DIR` in the `TIMERx_CTL0` register should be set to 0 for the up counting mode.

Whenever, if the update event software trigger is enabled by setting the `UPG` bit in the `TIMERx_SWEVG` register, the counter value will be initialized to 0 and generates an update event.

If set the `UPDIS` bit in `TIMERx_CTL0` register, the update event is disabled.

When an update event occurs, all the registers (repetition counter, auto reload register, prescaler register) are updated.

[Figure 15-61. Up-counter timechart, PSC=0/1](#) show some examples of the counter

behavior for different clock prescaler factor when `TIMERx_CAR=0x63`.

**Figure 15-61. Up-counter timechart, PSC=0/1**

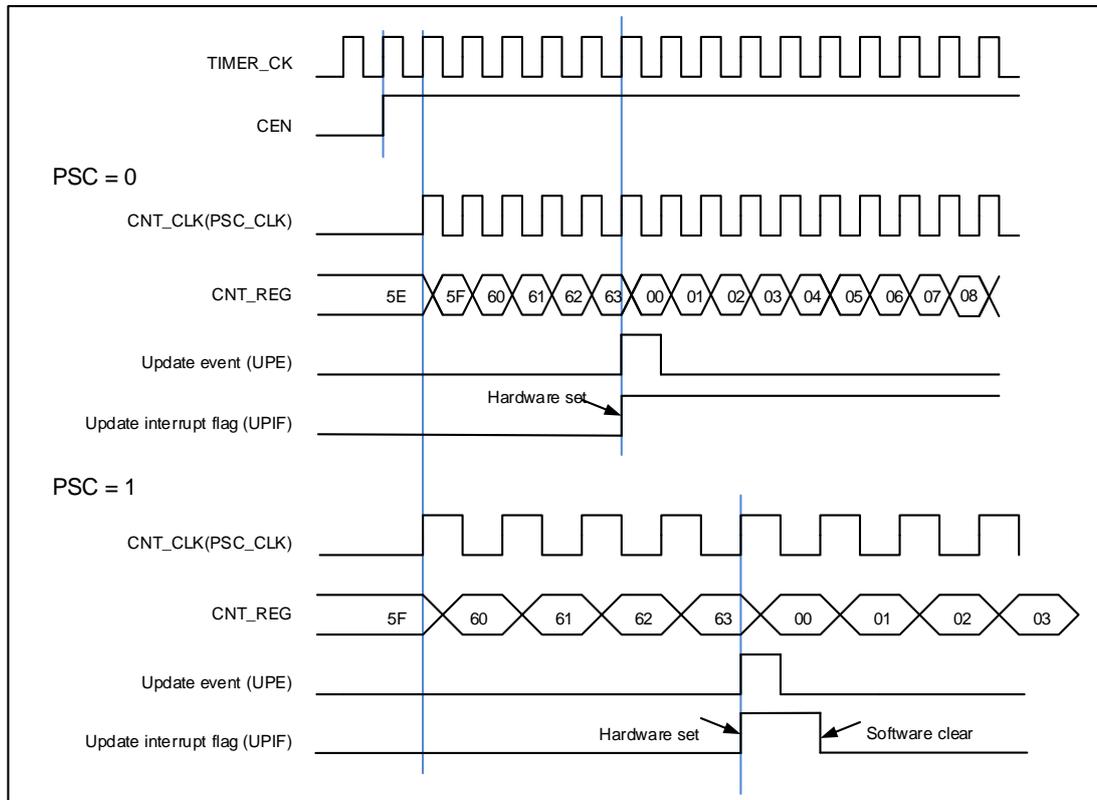
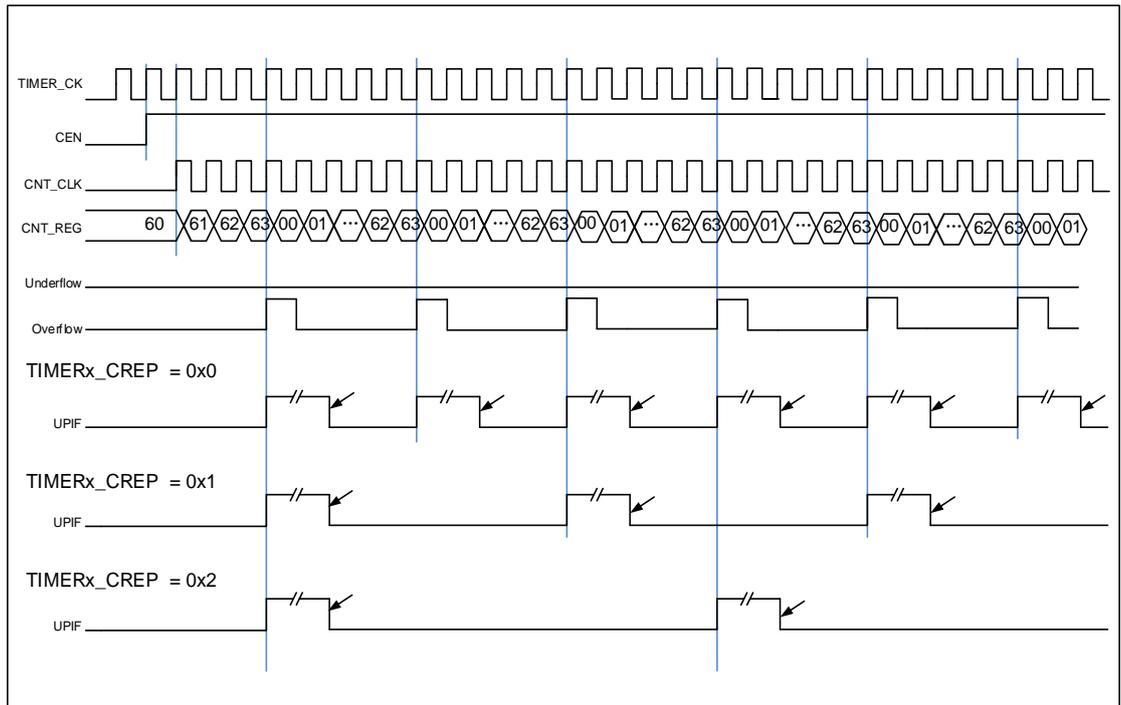




Figure 15-63. Repetition timechart for up-counter



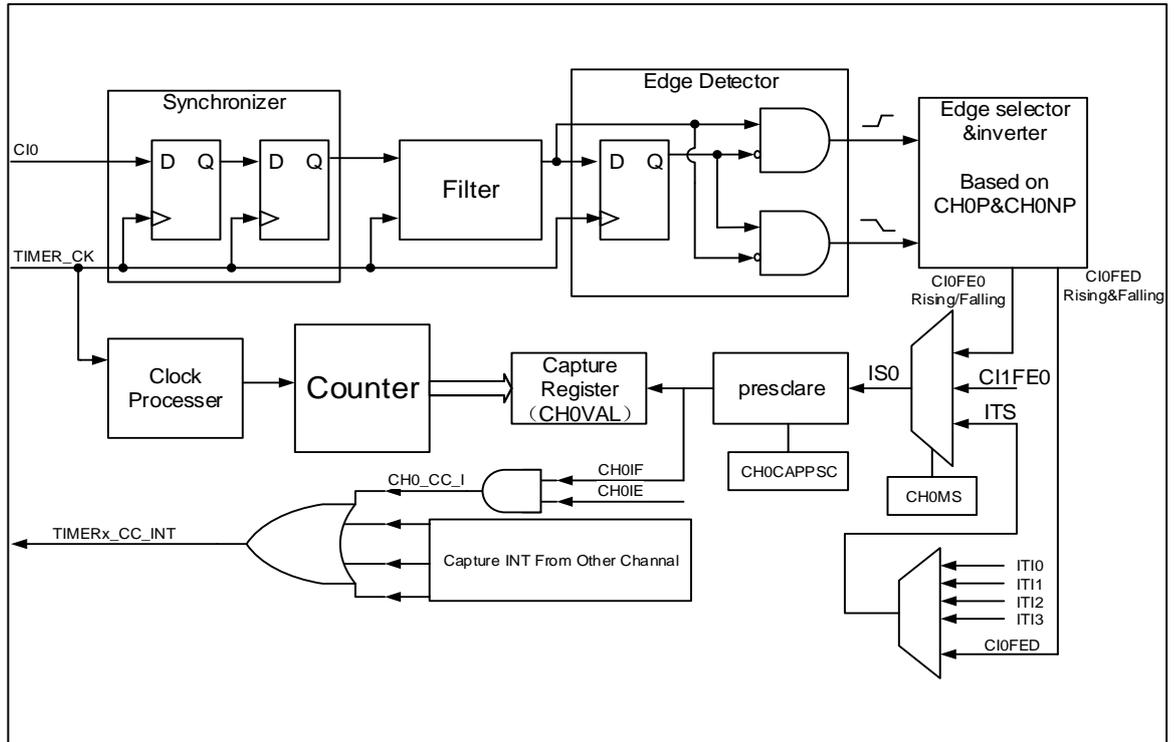
### Capture/compare channels

The general level3 timer has two independent channels which can be used as capture inputs or compare match outputs. Each channel is built around a channel capture compare register including an input stage, channel controller and an output stage.

#### ■ Input capture mode

Capture mode allows the channel to perform measurements such as pulse timing, frequency, period, duty cycle and so on. The input stage consists of a digital filter, a channel polarity selection, edge detection and a channel prescaler. When a selected edge occurs on the channel input, the current value of the counter is captured into the TIMEx\_CHxCV register, at the same time the CHxIF bit is set and the channel interrupt is generated if enabled by CHxIE = 1.

Figure 15-64. Input capture logic



Channels' input signals ( $Cix$ ) is the  $TIMERx\_CHx$  signal. First, the channel input signal ( $Cix$ ) is synchronized to  $TIMER\_CK$  domain, and then sampled by a digital filter to generate a filtered input signal. Then through the edge detector, the rising and falling edge are detected. You can select one of them by  $CHxP$ . One more selector is for the other channel and trig, controlled by  $CHxMS$ . The  $IC\_prescaler$  make several the input event generate one effective capture event. On the capture event,  $CHxVAL$  will restore the value of Counter.

So the process can be divided to several steps as below:

**Step1:** Filter configuration. ( $CHxCAPFLT$  in  $TIMERx\_CHCTL0$ )

Based on the input signal and requested signal quality, configure compatible  $CHxCAPFLT$ .

**Step2:** Edge selection. ( $CHxP/CHxNP$  in  $TIMERx\_CHCTL2$ )

Rising or falling edge, choose one by  $CHxP/CHxNP$ .

**Step3:** Capture source selection. ( $CHxMS$  in  $TIMERx\_CHCTL0$ )

As soon as you select one input capture source by  $CHxMS$ , you have set the channel to input mode ( $CHxMS \neq 0x0$ ) and  $TIMERx\_CHxCV$  cannot be written any more.

**Step4:** Interrupt enable. ( $CHxIE$  and  $CHxDEN$  in  $TIMERx\_DMAINTEN$ )

Enable the related interrupt enable; you can got the interrupt and DMA request.

**Step5:** Capture enables. ( $CHxEN$  in  $TIMERx\_CHCTL2$ )

**Result:** when you wanted input signal is got,  $TIMERx\_CHxCV$  will be set by counter's value.

And CHxIF is asserted. If the CHxIF is high, the CHxOF will be asserted also. The interrupt and DMA request will be asserted based on the configuration of CHxIE and CHxDEN in TIMERx\_DMAINTEN

**Direct generation:** if you want to generate a DMA request or Interrupt, you can set CHxG by software directly.

The input capture mode can be also used for pulse width measurement from signals on the TIMERx\_CHx pins. For example, PWM signal connect to CI0 input. Select channel 0 capture signals to CI0 by setting CH0MS to 2'b01 in the channel control register (TIMERx\_CHCTL0) and set capture on rising edge. Select channel 1 capture signal to CI0 by setting CH1MS to 2'b10 in the channel control register (TIMERx\_CHCTL0) and set capture on falling edge. The counter set to restart mode and restart on channel 0 rising edge. Then the TIMERx\_CH0CV can measure the PWM period and the TIMERx\_CH1CV can measure the PWM duty.

## ■ Output compare mode

In output compare mode, the TIMERx can generate timed pulses with programmable position, polarity, duration and frequency. When the counter matches the value in the CHxVAL register of an output compare channel, the channel (n) output can be set, cleared, or toggled based on CHxCOMCTL. When the counter reaches the value in the CHxVAL register, the CHxIF bit is set and the channel (n) interrupt is generated if CHxIE = 1. And the DMA request will be assert, if CHxDEN =1.

So the process can be divided to several steps as below:

**Step1:** Clock Configuration. Such as clock source, clock prescaler and so on.

**Step2:** Compare mode configuration.

- \* Set the shadow enable mode by CHxCOMSEN
- \* Set the output mode (Set/Clear/Toggle) by CHxCOMCTL.
- \* Select the active high polarity by CHxP/CHxNP
- \* Enable the output by CHxEN

**Step3:** Interrupt/DMA-request enables configuration by CHxIE/ CHxDEN

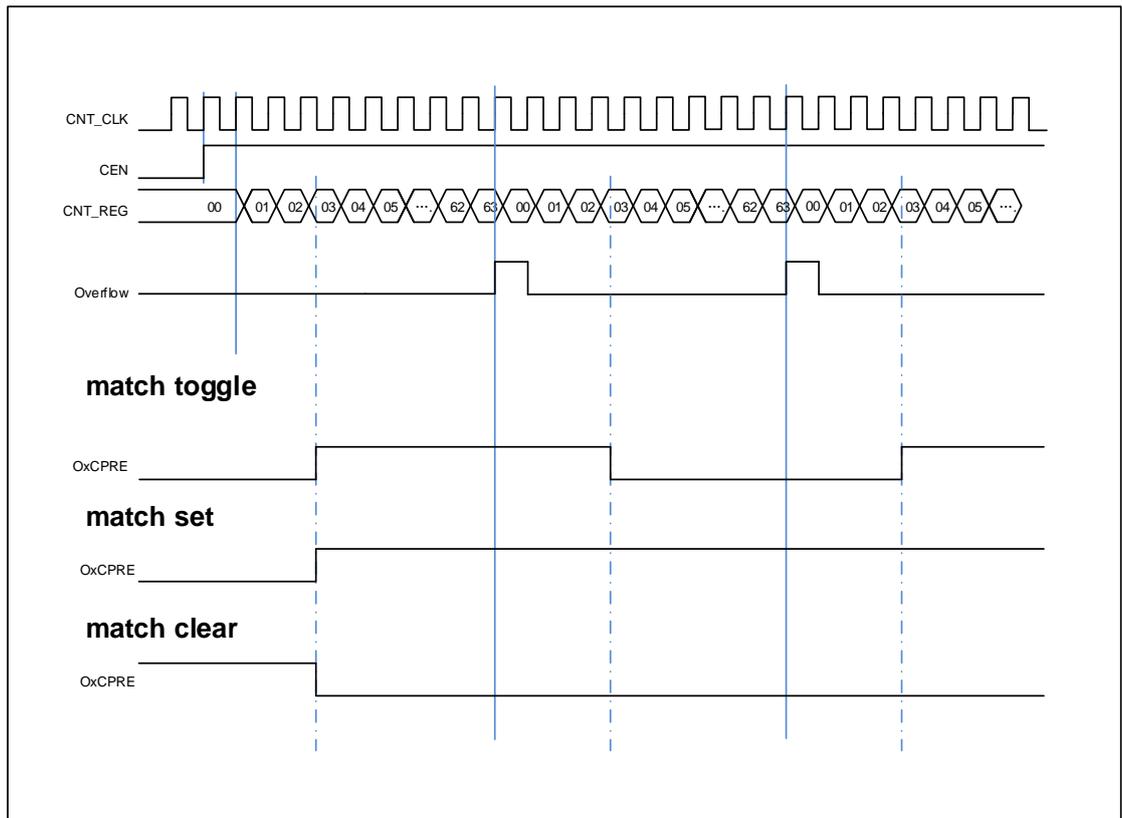
**Step4:** Compare output timing configuration by TIMERx\_CAR and TIMERx\_CHxCV

About the CHxVAL; you can change it on the go to meet the waveform you expected.

**Step5:** Start the counter by CEN.

The timechart below show the three compare modes toggle/set/clear. CAR=0x63, CHxVAL=0x3

Figure 15-65. Output-compare under three modes



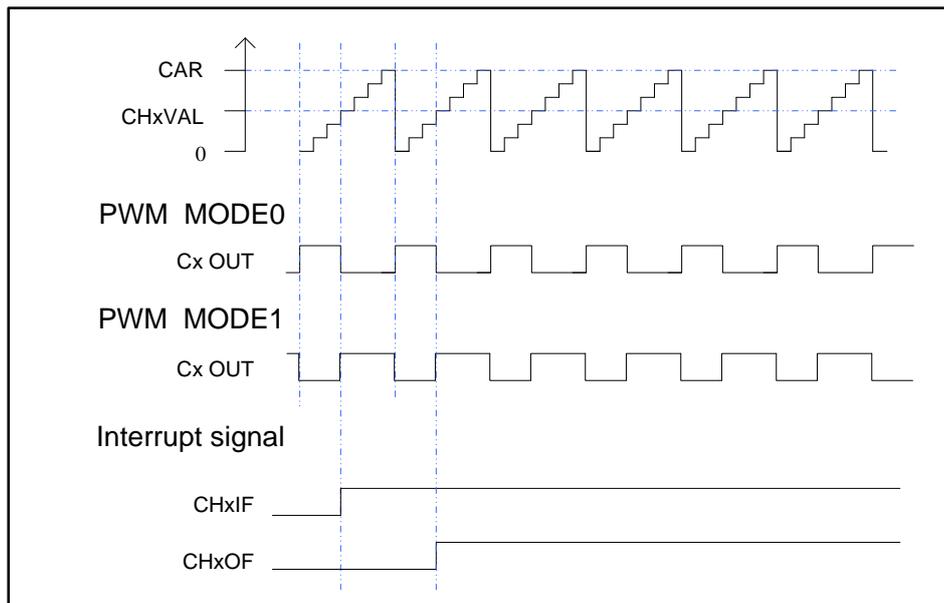
### PWM mode

In the output PWM mode (by setting the CHxCOMCTL bits to 3'b110 (PWM mode0) or to 3'b111(PWM mode1), the channel can generate PWM waveform according to the TIMEx\_CAR registers and TIMEx\_CHxCV registers.

The period is determined by TIMEx\_CAR and duty cycle is determined by TIMEx\_CHxCV. [Figure 15-57. PWM mode timechart](#) shows the PWM output mode and interrupts waveform.

If TIMEx\_CHxCV is greater than TIMEx\_CAR, the output will be always active under PWM mode0 (CHxCOMCTL==3'b110).

And if TIMEx\_CHxCV is equal to zero, the output will be always inactive under PWM mode0 (CHxCOMCTL==3'b110).

**Figure 15-66. PWM mode timechart**


### Channel output reference signal

When the TIMERx is used in the compare match output mode, the OxCPRE signal (Channel x Output prepare signal) is defined by setting the CHxCOMCTL field. The OxCPRE signal has several types of output function. These include, keeping the original level by setting the CHxCOMCTL field to 0x00, set to 1 by setting the CHxCOMCTL field to 0x01, set to 0 by setting the CHxCOMCTL field to 0x02 or signal toggle by setting the CHxCOMCTL field to 0x03 when the counter value matches the content of the TIMERx\_CHxCV register.

The PWM mode 0 and PWM mode 1 outputs are also another kind of OxCPRE output which is setup by setting the CHxCOMCTL field to 0x06/0x07. In these modes, the OxCPRE signal level is changed according to the counting direction and the relationship between the counter value and the TIMERx\_CHxCV content. With regard to a more detail description refer to the relative bit definition.

Another special function of the OxCPRE signal is a forced output which can be achieved by setting the CHxCOMCTL field to 0x04/0x05. Here the output can be forced to an inactive/active level irrespective of the comparison condition between the counter and the TIMERx\_CHxCV values.

### Outputs complementary

Function of complementary is for a pair of CHx\_O and CHx\_ON. Those two output signals cannot be active at the same time. The TIMERx has 2 channels, but only the first channel have this function. The complementary signals CHx\_O and CHx\_ON are controlled by a group of parameters: the CHxEN and CHxNEN bits in the TIMERx\_CHCTL2 register and the POEN, ROS, IOS, ISOx and ISOxN bits in the TIMERx\_CCHP and TIMERx\_CTL1 registers. The outputs polarity is determined by CHxP and CHxNP bits in the TIMERx\_CHCTL2 register.

**Table 15-8. Complementary outputs controlled by parameters**

Complementary Parameters					Output Status			
POEN	ROS	IOS	CHxEN	CHxNEN	CHx_O	CHx_ON		
0	0/1	0	0	0	CHx_O / CHx_ON = LOW CHx_O / CHx_ON output disable.			
				1	CHx_O = CHxP CHx_ON = CHxNP			
			1	0	CHx_O/CHx_ON output disable.			
				1	If clock is enable: CHx_O = ISOx CHx_ON = ISOxN			
		1	0	0	0	CHx_O = CHxP CHx_ON = CHxNP CHx_O/CHx_ON output disable.		
					1	CHx_O = CHxP CHx_ON = CHxNP		
			1	0	CHx_O/CHx_ON output enable.			
				1	If clock is enable: CHx_O = ISOx CHx_ON = ISOxN			
1	0	0/1	0	0	CHx_O/CHx_ON = LOW CHx_O/CHx_ON output disable.			
				1	CHx_O = LOW CHx_O output disable.	CHx_ON=OxCPRE ⊕ CHxNP CHx_ON output enable		
			1	0	CHx_O=OxCPRE⊕CHxP CHx_O output enable		CHx_ON = LOW CHx_ON output disable.	
				1	CHx_O=OxCPRE⊕CHxP CHx_O output enable		CHx_ON=OxCPRE⊕CHxNP CHx_ON output enable	
			1	0	0	CHx_O = CHxP CHx_O output disable.		CHx_ON = CHxNP CHx_ON output disable.
					1	CHx_O = CHxP CHx_O output enable		CHx_ON=OxCPRE⊕CHxNP CHx_ON output enable
	1	0		CHx_O=OxCPRE⊕CHxP CHx_O output enable		CHx_ON = CHxNP CHx_ON output enable.		
		1		CHx_O=OxCPRE⊕CHxP CHx_O output enable		CHx_ON=OxCPRE⊕CHxNP CHx_ON output enable.		

### Dead time insertion

The dead time insertion is enabled when both CHxEN and CHxNEN are 1'b1, and set POEN is also necessary. The field named DTCFG defines the dead time delay that can be used for channel 0. The detail about the delay time, refer to the register TIMERx\_CCHP.

The dead time delay insertion ensures that no two complementary signals drive the active state at the same time.

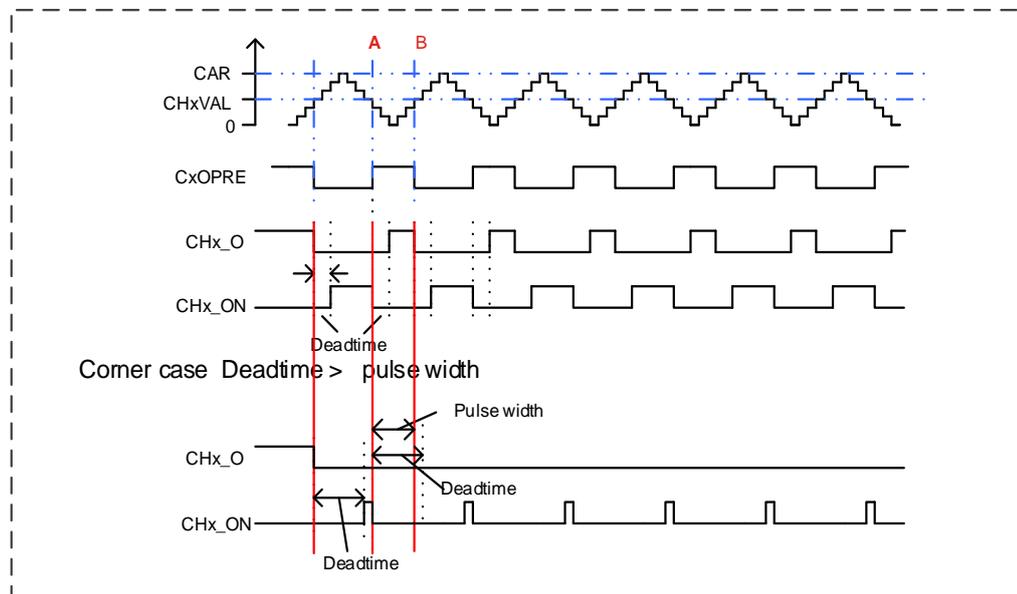
When the channel (x) match (TIMERx counter = CHxVAL) occurs, OxCPRE will be toggled because under PWM0 mode. At point A in the [Figure 15-67. Complementary output with dead-time insertion](#), CHx\_O signal remains at the low value until the end of the deadtime delay, while CHx\_ON will be cleared at once. Similarly, At point B when counter match (counter = CHxVAL) occurs again, OxCPRE is cleared, CHx\_O signal will be cleared at once, while CHx\_ON signal remains at the low value until the end of the dead time delay.

Sometimes, we can see corner cases about the dead time insertion. For example:

The dead time delay is greater than or equal to the CHx\_O duty cycle, then the CHx\_O signal is always the inactive value. (as show in the [Figure 15-67. Complementary output with dead-time insertion](#).)

The dead time delay is greater than or equal to the CHx\_ON duty cycle, then the CHx\_ON signal is always the inactive value.

**Figure 15-67. Complementary output with dead-time insertion.**



### Break function

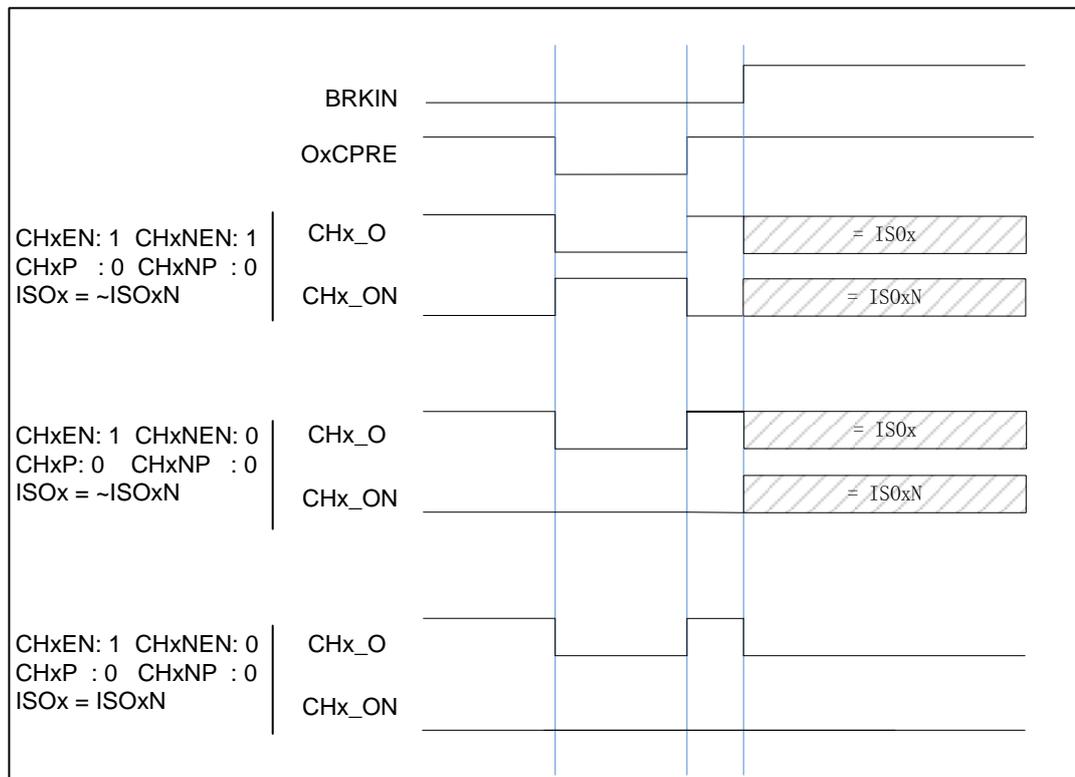
In this function, the output CHx\_O and CHx\_ON are controlled by the POEN, IOS and ROS bits in the TIMERx\_CCHP register, ISOx and ISOxN bits in the TIMERx\_CTL1 register and

cannot be set both to active level when break occurs. The break sources are input break pin and HXTAL stuck event by Clock Monitor (CKM) in RCU. The break function enabled by setting the BRKEN bit in the TIMERx\_CCHP register. The break input polarity is setting by the BRKP bit in TIMERx\_CCHP.

When a break occurs, the POEN bit is cleared asynchronously, the output CHx\_O and CHx\_ON are driven with the level programmed in the ISOx bit and ISOxN in the TIMERx\_CTL1 register as soon as POEN is 0. If IOS is 0 then the timer releases the enable output else the enable output remains high. The complementary outputs are first put in reset state, and then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the ISOx and ISOxN bits after a dead-time.

When a break occurs, the BRKIF bit in the TIMERx\_INTF register is set. If BRKIE is 1, an interrupt generated.

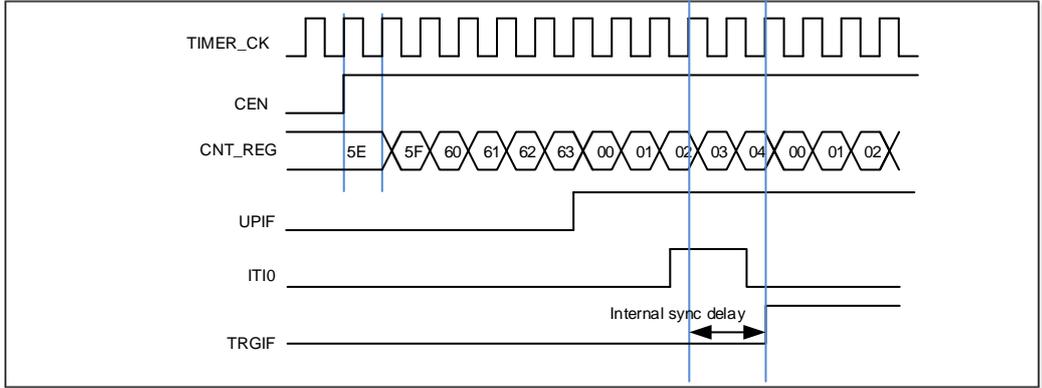
**Figure 15-68. Output behavior in response to a break(The break high active)**

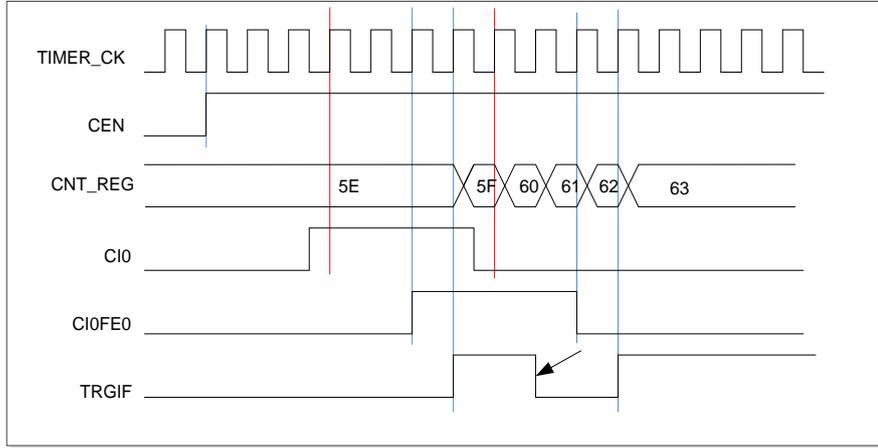
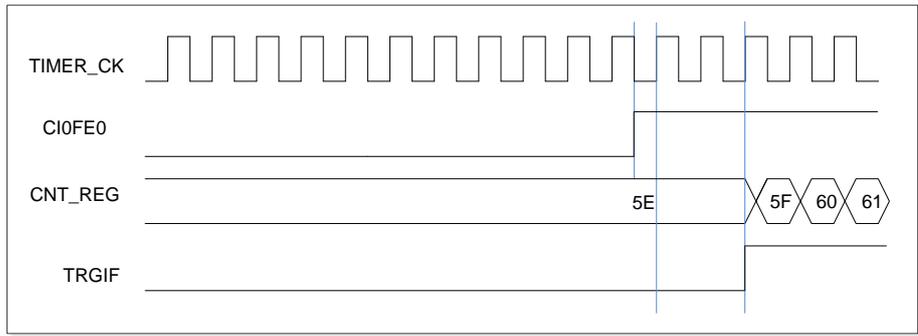


## Slave controller

The TIMERx can be synchronized with a trigger in several modes including the restart mode, the pause mode and the event mode which is selected by the SMC [2:0] in the TIMERx\_SMCFG register. The trigger input of these modes can be selected by the TRGS [2:0] in the TIMERx\_SMCFG register.

**Table 15-9. Slave mode example table**

	Mode Selection	Source Selection	Polarity Selection	Filter and Prescaler
LIST	SMC[2:0]  3'b100 (restart mode)  3'b101 (pause mode)  3'b110 (event mode)	TRGS[2:0] 000: ITI0 001: ITI1 010: ITI2 011: ITI3 100: CI0F_ED 101: CI0FE0 110: CI1FE1 111: Reserved	If you choose the CI0FE0 or CI1FE1, configure the CHxP and CHxNP for the polarity selection and inversion.	For the ITIx no filter and prescaler can be used.  For the CIx, configure Filter by CHxCAPFLT, no prescaler can be used.
Exam1	Restart mode  The counter can be clear and restart when a rising trigger input.	TRGS[2:0]=3'b000  ITI0 is the selection.	-  For ITI0, no polarity selector can be used.	-  For the ITI0, no filter and prescaler can be used.
<b>Figure 15-69. Restart mode</b>				
				
Exam2	Pause mode  The counter can be paused when the trigger input is low.	TRGS[2:0]=3'b101  CI0FE0 is the selection.	TI0S=0.(Non-xor) [CH0NP==0, CH0P==0] no inverted. Capture will be sensitive to the rising edge only.	Filter is bypass in this example.

	Mode Selection	Source Selection	Polarity Selection	Filter and Prescaler
	<p><b>Figure 15-70. Pause mode</b></p> 			
Exam3	<p>Event mode The counter will start to count when a rising trigger input.</p>	<p>TRGS[2:0]=3 'b101 CIOFE0 is the selection.</p>	<p>TIOS=0.(Non-xor) [CH0NP==0, CH0P==0] no inverted.</p>	<p>Filter is bypass in this example.</p>
	<p><b>Figure 15-71. Event mode</b></p> 			

## Single pulse mode

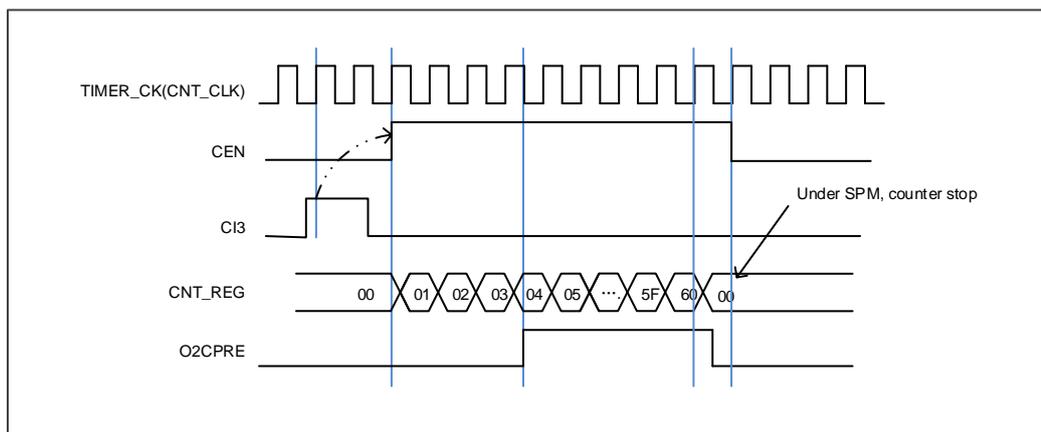
Single pulse mode is opposite to the repetitive mode, which can be enabled by setting SPM in TIMERx\_CTL0. When you set SPM, the counter will be clear and stop when the next update event automatically. In order to get pulse waveform, you can set the TIMERx to PWM mode or compare by CHxCOMCTL.

Once the timer is set to operate in the single pulse mode, it is not necessary to set the timer enable bit CEN in the TIMERx\_CTL0 register to 1 to enable the counter. The trigger to generate a pulse can be sourced from the trigger signals edge or by setting the CEN bit to 1

using software. Setting the CEN bit to 1 or a trigger from the trigger signals edge can generate a pulse and then keep the CEN bit at a high state until the update event occurs or the CEN bit is written to 0 by software. If the CEN bit is cleared to 0 using software, the counter will be stopped and its value held. If the CEN bit is automatically cleared to 0 by a hardware update event, the counter will be reinitialized.

In the single pulse mode, the trigger active edge which sets the CEN bit to 1 will enable the counter. However, there exist several clock delays to perform the comparison result between the counter value and the `TIMERx_CHxCV` value. In order to reduce the delay to a minimum value, the user can set the `CHxCOMFEN` bit in each `TIMERx_CHCTL0/1` register. After a trigger rising occurs in the single pulse mode, the `OxCPRE` signal will immediately be forced to the state which the `OxCPRE` signal will change to, as the compare match event occurs without taking the comparison result into account. The `CHxCOMFEN` bit is available only when the output channel is configured to operate in the PWM0 or PWM1 output mode and the trigger source is derived from the trigger signal.

**Figure 15-72. Single pulse mode `TIMERx_CHxCV = 0x04` `TIMERx_CAR=0x60`**



## Timers interconnection

Refer to [Advanced timer \(TIMERx, x=0\)](#).

**Table 15-10. TIMERx(x=14) interconnection**

Slave TIMER	IT10(TRGS = 000)	IT11(TRGS = 001)	IT12(TRGS = 010)	IT13(TRGS = 011)
TIMER14	TIMER1	TIMER2	Reserved	Reserved

## Timer DMA mode

Timer's DMA mode is the function that configures timer's register by DMA module. The relative registers are `TIMERx_DMACFG` and `TIMERx_DMATB`. Of course, you have to enable a DMA request which will be asserted by some internal event. When the interrupt

event was asserted, TIMERx will send a request to DMA, which is configured to M2P mode and PADDR is TIMERx\_DMATB, then DMA will access the TIMERx\_DMATB. In fact, register TIMERx\_DMATB is only a buffer; timer will map the TIMERx\_DMATB to an internal register, appointed by the field of DMATA in TIMERx\_DMACFG . If the field of DMATC in TIMERx\_DMACFG is 0(1 transfer), then the timer's DMA request is finished. While if TIMERx\_DMATC is not 0, such as 3( 4 transfers), then timer will send 3 more requests to DMA, and DMA will access timer's registers DMATA+0x4, DMATA+0x8, DMATA+0xc at the next 3 accesses to TIMERx\_DMATB. In one word, one time DMA internal interrupt event assert, DMATC+1 times request will be send by TIMERx.

If one more time DMA request event coming, TIMERx will repeat the process as above.

### **Timer debug mode**

When the Cortex™-M3 halted, and the TIMERx\_HOLD configuration bit in DBG\_CTL1 register set to 1, the TIMERx counter stops.

### 15.4.5. TIMERx registers(x=14)

#### Control register 0 (TIMERx\_CTL0)

Address offset: 0x00

Reset value: 0x0000

This register can be accessed by half-word(16-bit) or word(32-bit)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				CKDIV[1:0]		ARSE		Reserved				SPM	UPS	UPDIS	CEN
				rw		rw						rw	rw	rw	rw

Bits	Fields	Descriptions
15:10	Reserved	Must be kept at reset value
9:8	CKDIV[1:0]	Clock division The CKDIV bits can be configured by software to specify division ratio between the timer clock (TIMER_CK) and the dead-time and sampling clock (DTS), which is used by the dead-time generators and the digital filters. 00: $f_{DTS} = f_{TIMER\_CK}$ 01: $f_{DTS} = f_{TIMER\_CK} / 2$ 10: $f_{DTS} = f_{TIMER\_CK} / 4$ 11: Reserved
7	ARSE	Auto-reload shadow enable 0: The shadow register for TIMERx_CAR register is disabled 1: The shadow register for TIMERx_CAR register is enabled
6:4	Reserved	Must be kept at reset value
3	SPM	Single pulse mode. 0: Counter continues after update event. 1: The CEN is cleared by hardware and the counter stops at next update event.
2	UPS	Update source This bit is used to select the update event sources by software. 0: Any of the following events generate an update interrupt or DMA request: <ul style="list-style-type: none"> <li>- The UPG bit is set</li> <li>- The counter generates an overflow or underflow event</li> <li>- The slave mode controller generates an update event.</li> </ul> 1: Only counter overflow/underflow generates an update interrupt or DMA request.
1	UPDIS	Update disable. This bit is used to enable or disable the update event generation. 0: update event enable. The update event is generate and the buffered registers are loaded with their preloaded values when one of the following events occurs: <ul style="list-style-type: none"> <li>- The UPG bit is set</li> </ul>

- The counter generates an overflow or underflow event
- The slave mode controller generates an update event.

1: update event disable. The buffered registers keep their value, while the counter and the prescaler are reinitialized if the UG bit is set or if the slave mode controller generates a hardware reset event.

0            CEN            Counter enable  
 0: Counter disable  
 1: Counter enable

The CEN bit must be set by software when timer works in external clock, pause mode and encoder mode. While in event mode, the hardware can set the CEN bit automatically.

## Control register 1 (TIMERx\_CTL1)

Address offset: 0x04

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved				ISO1	ISO0N	ISO0	Reserved	MMC[2:0]			DMAS	CCUC	Reserved	CCSE	
					rw	rw	rw			rw		rw	rw		rw	

Bits	Fields	Descriptions
15:11	Reserved	Must be kept at reset value
10	ISO1	Idle state of channel 1 output Refer to ISO0 bit
9	ISO0N	Idle state of channel 0 complementary output 0: When POEN bit is reset, CH0_ON is set low. 1: When POEN bit is reset, CH0_ON is set high This bit can be modified only when PROT [1:0] bits in TIMERx_CCHP register is 00.
8	ISO0	Idle state of channel 0 output 0: When POEN bit is reset, CH0_O is set low. 1: When POEN bit is reset, CH0_O is set high The CH0_O output changes after a dead-time if CH0_ON is implemented. This bit can be modified only when PROT [1:0] bits in TIMERx_CCHP register is 00.
7	Reserved	Must be kept at reset value
6:4	MMC[2:0]	Master mode control These bits control the selection of TRGO signal, which is sent in master mode to slave timers for synchronization function. 000: Reset. When the UPG bit in the TIMERx_SWEVG register is set or a reset is

generated by the slave mode controller, a TRGO pulse occurs. And in the latter case, the signal on TRGO is delayed compared to the actual reset.

001: Enable. This mode is useful to start several timers at the same time or to control a window in which a slave timer is enabled. In this mode the master mode controller selects the counter enable signal as TRGO. The counter enable signal is set when CEN control bit is set or the trigger input in pause mode is high. There is a delay between the trigger input in pause mode and the TRGO output, except if the master-slave mode is selected.

010: Update. In this mode the master mode controller selects the update event as TRGO.

011: Capture/compare pulse. In this mode the master mode controller generates a TRGO pulse when a capture or a compare match occurred in channel0.

100: Compare. In this mode the master mode controller selects the O0CPRE signal is used as TRGO

101: Compare. In this mode the master mode controller selects the O1CPRE signal is used as TRGO

110: Compare. In this mode the master mode controller selects the O2CPRE signal is used as TRGO

111: Compare. In this mode the master mode controller selects the O3CPRE signal is used as TRGO

3 DMAS

DMA request source selection

0: DMA request of channel x is sent when capture/compare event occurs.

1: DMA request of channel x is sent when update event occurs.

2 CCUC

Commutation control shadow register update control

When the commutation control shadow enable (for CHxEN, CHxNEN and CHxCOMCTL bits) are set (CCSE=1), these shadow registers update are controlled as below:

0: The shadow registers update by when CMTG bit is set.

1: The shadow registers update by when CMTG bit is set or a rising edge of TRGI occurs.

When a channel does not have a complementary output, this bit has no effect.

1 Reserved

Must be kept at reset value.

0 CCSE

Commutation control shadow enable

0: The shadow registers for CHxEN, CHxNEN and CHxCOMCTL bits are disabled.

1: The shadow registers for CHxEN, CHxNEN and CHxCOMCTL bits are enabled.

After these bits have been written, they are updated based when commutation event coming.

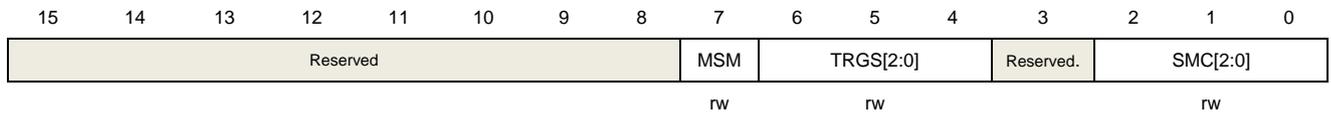
When a channel does not have a complementary output, this bit has no effect.

## Slave mode configuration register (TIMERx\_SMCFG)

Address offset: 0x08

Reset value: 0x0000

This register can be accessed by half-word(16-bit) or word(32-bit)



Bits	Fields	Descriptions
15:8	Reserved	Must be kept at reset value
7	MSM	<p>Master-slave mode</p> <p>This bit can be used to synchronize selected timers to begin counting at the same time. The TRGI is used as the start event, and through TRGO, timers are connected together.</p> <p>0: Master-slave mode disable 1: Master-slave mode enable</p>
6:4	TRGS[2:0]	<p>Trigger selection</p> <p>This bit-field specifies which signal is selected as the trigger input, which is used to synchronize the counter.</p> <p>000: Internal trigger input 0 (ITIO) TIMER1 001: Internal trigger input 1 (IT11) TIERM2 010: Reserved 011: Reserved 100: CI0 edge flag (CI0F_ED) 101: Channel 0 input filtered output (CI0FE0) 110: Channel 1 input filtered output (CI1FE1) 111: Reserved</p> <p>These bits must not be changed when slave mode is enabled.</p>
3	Reserved	Must be kept at reset value
2:0	SMC[2:0]	<p>Slave mode control</p> <p>000: Disable mode. The slave mode is disabled; The prescaler is clocked directly by the internal clock (TIMER_CK) when CEN bit is set high. 001: Reserved 010: Reserved 011: Reserved 100: Restart Mode. The counter is reinitialized and the shadow registers are updated on the rising edge of the selected trigger input. 101: Pause Mode. The trigger input enables the counter clock when it is high and disables the counter when it is low. 110: Event Mode. A rising edge of the trigger input enables the counter. The counter cannot be disabled by the slave mode controller.</p>

111: External Clock Mode 0. The counter counts on the rising edges of the selected trigger.

Because CIOF\_ED outputs 1 pulse for each transition on CIOF, and the pause mode checks the level of the trigger signal, when CIOF\_ED is selected as the trigger input, the pause mode must not be used.

## DMA and interrupt enable register (TIMERx\_DMAINTEN)

Address offset: 0x0C

Reset value: 0x0000

This register can be accessed by half-word(16-bit) or word(32-bit)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	TRGDEN	Reserved	Reserved	CH1DEN	CH0DEN	UPDEN	BRKIE	TRGIE	CMTIE	Reserved	Reserved	CH1IE	CH0IE	UPIE	
	rw	rw		rw	rw	rw	rw	rw	rw			rw	rw	rw	

Bits	Fields	Descriptions
15	Reserved	Must be kept at reset value
14	TRGDEN	Trigger DMA request enable 0: disabled 1: enabled
13:11	Reserved	Must be kept at reset value
10	CH1DEN	Channel 1 capture/compare DMA request enable 0: disabled 1: enabled
9	CH0DEN	Channel 0 capture/compare DMA request enable 0: disabled 1: enabled
8	UPDEN	Update DMA request enable 0: disabled 1: enabled
7	BRKIE	Break interrupt enable 0: disabled 1: enabled
6	TRGIE	Trigger interrupt enable 0: disabled 1: enabled
5	CMTIE	commutation interrupt enable 0: disabled 1: enabled

4:3	Reserved	Must be kept at reset value
2	CH1IE	Channel 1 capture/compare interrupt enable 0: disabled 1: enabled
1	CH0IE	Channel 0 capture/compare interrupt enable 0: disabled 1: enabled
0	UPIE	Update interrupt enable 0: disabled 1: enabled

### Interrupt flag register (TIMERx\_INTF)

Address offset: 0x10

Reset value: 0x0000

This register can be accessed by half-word(16-bit) or word(32-bit)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				CH1OF	CH0OF	Reserved.	BRKIF	TRGIF	CMTIF	Reserved.	CH1IF	CH0IF	UPIF		
				rc_w0	rc_w0		rc_w0	rc_w0	rc_w0		rc_w0	rc_w0	rc_w0		

Bits	Fields	Descriptions
15:11	Reserved	Must be kept at reset value
10	CH1OF	Channel 1 over capture flag Refer to CH0OF description
9	CH0OF	Channel 0 over capture flag When channel 0 is configured in input mode, this flag is set by hardware when a capture event occurs while CH0IF flag has already been set. This flag is cleared by software. 0: No over capture interrupt occurred 1: Over capture interrupt occurred
8	Reserved	Must be kept at reset value.
7	BRKIF	Break interrupt flag This flag is set by hardware when the break input goes active, and cleared by software if the break input is not active. 0: No active level break has been detected. 1: An active level has been detected.
6	TRGIF	Trigger interrupt flag This flag is set by hardware on trigger event and cleared by software. When the slave mode controller is enabled in all modes but pause mode, an active edge on trigger

input generates a trigger event. When the slave mode controller is enabled in pause mode both edges on trigger input generates a trigger event.

0: No trigger event occurred.  
1: Trigger interrupt occurred.

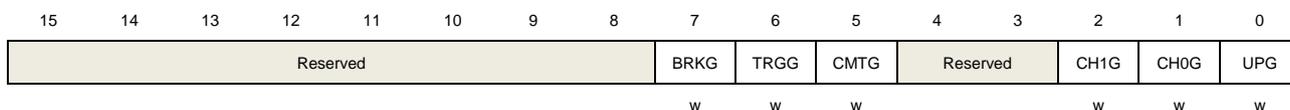
5	CMTIF	<p>Channel commutation interrupt flag</p> <p>This flag is set by hardware when channel's commutation event occurs, and cleared by software</p> <p>0: No channel commutation interrupt occurred 1: Channel commutation interrupt occurred</p>
4:3	Reserved	Must be kept at reset value
2	CH1IF	<p>Channel 1 's capture/compare interrupt flag</p> <p>Refer to CH0IF description</p>
1	CH0IF	<p>Channel 0 's capture/compare interrupt flag</p> <p>This flag is set by hardware and cleared by software. When channel 0 is in input mode, this flag is set when a capture event occurs. When channel 0 is in output mode, this flag is set when a compare event occurs.</p> <p>0: No Channel 0 interrupt occurred 1: Channel 0 interrupt occurred</p>
0	UPIF	<p>Update interrupt flag</p> <p>This bit is set by hardware on an update event and cleared by software.</p> <p>0: No update interrupt occurred 1: Update interrupt occurred</p>

## Software event generation register (TIMERx\_SWEVG)

Address offset: 0x14

Reset value: 0x0000

This register can be accessed by half-word(16-bit) or word(32-bit)



Bits	Fields	Descriptions
15:8	Reserved	Must be kept at reset value
7	BRKG	<p>Break event generation</p> <p>This bit is set by software and cleared by hardware automatically. When this bit is set, the POEN bit is cleared and BRKIF flag is set, related interrupt or DMA transfer can occur if enabled.</p> <p>0: No generate a break event</p>

		1: Generate a break event
6	TRGG	<p>Trigger event generation</p> <p>This bit is set by software and cleared by hardware automatically. When this bit is set, the TRGIF flag in TIMERx_INTF register is set, related interrupt or DMA transfer can occur if enabled.</p> <p>0: No generate a trigger event 1: Generate a trigger event</p>
5	CMTG	<p>Channel commutation event generation</p> <p>This bit is set by software and cleared by hardware automatically. When this bit is set, channel's capture/compare control registers (CHxEN, CHxNEN and CHxCOMCTL bits) are updated based on the value of CCSE (in the TIMERx_CTL1).</p> <p>0: No affect 1: Generate channel's c/c control update event</p>
4:3	Reserved	Must be kept at reset value
2	CH1G	<p>Channel 1's capture or compare event generation</p> <p>Refer to CH0G description</p>
1	CH0G	<p>Channel 0's capture or compare event generation</p> <p>This bit is set by software in order to generate a capture or compare event in channel 0, it is automatically cleared by hardware. When this bit is set, the CH0IF flag is set, the corresponding interrupt or DMA request is sent if enabled. In addition, if channel 1 is configured in input mode, the current value of the counter is captured in TIMERx_CH0CV register, and the CH0OF flag is set if the CH0IF flag was already high.</p> <p>0: No generate a channel 1 capture or compare event 1: Generate a channel 1 capture or compare event</p>
0	UPG	<p>Update event generation</p> <p>This bit can be set by software, and cleared by hardware automatically. When this bit is set, the counter is cleared if the center-aligned or up counting mode is selected, else (down counting) it takes the auto-reload value. The prescaler counter is cleared at the same time.</p> <p>0: No generate an update event 1: Generate an update event</p>

## Channel control register 0 (TIMERx\_CHCTL0)

Address offset: 0x18

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)

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

Reserved	CH1COMCTL[2:0]	CH1COM SEN	CH1COM FEN	CH1MS[1:0]	Reserved	CH0COMCTL[2:0]	CH0COM SEN	CH0COM FEN	CH0MS[1:0]
	CH1CAPFLT[3:0]	CH1CAPPSC[1:0]				CH0CAPFLT[3:0]	CH0CAPPSC[1:0]		
	rw		rw	rw		rw		rw	rw

**Output compare mode:**

Bits	Fields	Descriptions
15	Reserved	Must be kept at reset value
14:12	CH1COMCTL[2:0]	Channel 1 compare output control Refer to CH0COMCTL description
11	CH1COMSEN	Channel 1 output compare shadow enable Refer to CH0COMSEN description
10	CH1COMFEN	Channel 1 output compare fast enable Refer to CH0COMSEN description
9:8	CH1MS[1:0]	Channel 1 mode selection This bit-field specifies the direction of the channel and the input signal selection. This bit-field is writable only when the channel is not active. (CH1EN bit in TIMERx_CHCTL2 register is reset). 00: Channel 1 is configured as output 01: Channel 1 is configured as input, IS1 is connected to CI1FE1 10: Channel 1 is configured as input, IS1 is connected to CI0FE1 11: Channel 1 is configured as input, IS1 is connected to ITS. This mode is working only if an internal trigger input is selected through TRGS bits in TIMERx_SMCFG register.
7	Reserved	Must be kept at reset value
6:4	CH0COMCTL[2:0]	Channel 0 compare output control This bit-field controls the behavior of the output reference signal O0CPRE which drives CH0_O and CH0_ON. O0CPRE is active high, while CH0_O and CH0_ON active level depends on CH0P and CH0NP bits. 000: Frozen. The O0CPRE signal keeps stable, independent of the comparison between the register TIMERx_CH0CV and the counter TIMERx_CNT. 001: Set the channel output. O0CPRE signal is forced high when the counter matches the output compare register TIMERx_CH0CV. 010: Clear the channel output. O0CPRE signal is forced low when the counter matches the output compare register TIMERx_CH0CV. 011: Toggle on match. O0CPRE toggles when the counter matches the output compare register TIMERx_CH0CV. 100: Force low. O0CPRE is forced low level. 101: Force high. O0CPRE is forced high level. 110: PWM mode0. When counting up, O0CPRE is active as long as the counter is smaller than TIMERx_CH0CV else inactive. When counting down, O0CPRE is

inactive as long as the counter is larger than `TIMERx_CH0CV` else active.

111: PWM mode1. When counting up, `O0CPRE` is inactive as long as the counter is smaller than `TIMERx_CH0CV` else active. When counting down, `O0CPRE` is active as long as the counter is larger than `TIMERx_CH0CV` else inactive.

When configured in PWM mode, the `O0CPRE` level changes only when the output compare mode switches from “frozen” mode to “PWM” mode or when the result of the comparison changes.

This bit cannot be modified when `PROT [1:0]` bit-filed in `TIMERx_CCHP` register is 11 and `CH0MS` bit-filed is 00(`COMPARE MODE`).

3	<code>CH0COMSEN</code>	<p>Channel 0 compare output shadow enable</p> <p>When this bit is set, the shadow register of <code>TIMERx_CH0CV</code> register, which updates at each update event, will be enabled.</p> <p>0: Channel 0 output compare shadow disable 1: Channel 0 output compare shadow enable</p> <p>The PWM mode can be used without validating the shadow register only in single pulse mode (<code>SPM</code> bit in <code>TIMERx_CTL0</code> register is set).</p> <p>This bit cannot be modified when <code>PROT [1:0]</code> bit-filed in <code>TIMERx_CCHP</code> register is 11 and <code>CH0MS</code> bit-filed is 00.</p>
2	<code>CH0COMFEN</code>	<p>Channel 0 output compare fast enable</p> <p>When this bit is set, the effect of an event on the trigger in input on the capture/compare output will be accelerated if the channel is configured in <code>PWM0</code> or <code>PWM1</code> mode. The output channel will treat an active edge on the trigger input as a compare match, and <code>CH0_O</code> is set to the compare level independently from the result of the comparison.</p> <p>0: Channel 0 output quickly compare disable. The minimum delay from an edge on the trigger input to activate <code>CH0_O</code> output is 5 clock cycles. 1: Channel 0 output quickly compare enable. The minimum delay from an edge on the trigger input to activate <code>CH0_O</code> output is 3 clock cycles.</p>
1:0	<code>CH0MS[1:0]</code>	<p>Channel 0 I/O mode selection</p> <p>This bit-field specifies the work mode of the channel and the input signal selection. This bit-field is writable only when the channel is not active. (<code>CH0EN</code> bit in <code>TIMERx_CHCTL2</code> register is reset).).</p> <p>00: Channel 0 is configured as output 01: Channel 0 is configured as input, <code>IS0</code> is connected to <code>CI0FE0</code> 10: Channel 0 is configured as input, <code>IS0</code> is connected to <code>CI1FE0</code> 11: Channel 0 is configured as input, <code>IS0</code> is connected to <code>ITS</code>, This mode is working only if an internal trigger input is selected through <code>TRGS</code> bits in <code>TIMERx_SMCFG</code> register.</p>

**Input capture mode:**

Bits	Fields	Descriptions
15:12	<code>CH1CAPFLT[3:0]</code>	Channel 1 input capture filter control

		Refer to CH0CAPFLT description
11:10	CH1CAPPSC[1:0]	Channel 1 input capture prescaler Refer to CH0CAPPSC description
9:8	CH1MS[1:0]	Channel 1 mode selection Same as Output compare mode
7:4	CH0CAPFLT[3:0]	Channel 0 input capture filter control An event counter is used in the digital filter, in which a transition on the output occurs after N input events. This bit-field specifies the frequency used to sample CIO input signal and the length of the digital filter applied to CIO. 0000: Filter disabled, $f_{SAMP}=f_{DTS}$ , $N=1$ 0001: $f_{SAMP}=f_{TIMER\_CK}$ , $N=2$ 0010: $f_{SAMP}=f_{TIMER\_CK}$ , $N=4$ 0011: $f_{SAMP}=f_{TIMER\_CK}$ , $N=8$ 0100: $f_{SAMP}=f_{DTS}/2$ , $N=6$ 0101: $f_{SAMP}=f_{DTS}/2$ , $N=8$ 0110: $f_{SAMP}=f_{DTS}/4$ , $N=6$ 0111: $f_{SAMP}=f_{DTS}/4$ , $N=8$ 1000: $f_{SAMP}=f_{DTS}/8$ , $N=6$ 1001: $f_{SAMP}=f_{DTS}/8$ , $N=8$ 1010: $f_{SAMP}=f_{DTS}/16$ , $N=5$ 1011: $f_{SAMP}=f_{DTS}/16$ , $N=6$ 1100: $f_{SAMP}=f_{DTS}/16$ , $N=8$ 1101: $f_{SAMP}=f_{DTS}/32$ , $N=5$ 1110: $f_{SAMP}=f_{DTS}/32$ , $N=6$ 1111: $f_{SAMP}=f_{DTS}/32$ , $N=8$
3:2	CH0CAPPSC[1:0]	Channel 0 input capture prescaler This bit-field specifies the factor of the prescaler on channel 0 input. The prescaler is reset when CH0EN bit in TIMEx_CHCTL2 register is clear. 00: Prescaler disable, capture is done on each channel input edge 01: Capture is done every 2 channel input edges 10: Capture is done every 4 channel input edges 11: Capture is done every 8 channel input edges
1:0	CH0MS[1:0]	Channel 0 mode selection Same as Output compare mode

### Channel control register 2 (TIMEx\_CHCTL2)

Address offset: 0x20

Reset value: 0x0000

This register can be accessed by half-word(16-bit) or word(32-bit)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							CH1NP	Reserved.	CH1P	CH1EN	CH0NP	CH0NEN	CH0P	CH0EN	
							rw		rw	rw	rw	rw	rw	rw	

Bits	Fields	Descriptions
15:8	Reserved	Must be kept at reset value
7	CH1NP	Channel 1 complementary output polarity Refer to CH0NP description
6	Reserved	Must be kept at reset value
5	CH1P	Channel 1 capture/compare function polarity Refer to CH0P description
4	CH1EN	Channel 1 capture/compare function enable Refer to CH0EN description
3	CH0NP	Channel 0 complementary output polarity When channel 0 is configured in output mode, this bit specifies the complementary output signal polarity. 0: Channel 0 active high 1: Channel 0 active low When channel 0 is configured in input mode, In conjunction with CH0P, this bit is used to define the polarity of CI0. This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 or 10.
2	CH0NEN	Channel 0 complementary output enable When channel 0 is configured in output mode, setting this bit enables the complementary output in channel0. 0: Channel 0 complementary output disabled 1: Channel 0 complementary output enabled
1	CH0P	Channel 0 capture/compare function polarity When channel 0 is configured in output mode, this bit specifies the output signal polarity. 0: Channel 0 active high 1: Channel 0 active low When channel 0 is configured in input mode, this bit specifies the CI0 signal polarity. [CH0NP, CH0P] will select the active trigger or capture polarity for CI0FE0 or CI1FE0. [CH0NP==0, CH0P==0]: ClxFE0's rising edge is the active signal for capture or trigger operation in slave mode. And ClxFE0 will not be inverted. [CH0NP==0, CH0P==1]: ClxFE0's falling edge is the active signal for capture or trigger operation in slave mode. And ClxFE0 will be inverted. [CH0NP==1, CH0P==0]: Reserved. [CH0NP==1, CH0P==1]: ClxFE0's falling and rising edge are both the active signal for capture or trigger operation in slave mode. And ClxFE0 will be not inverted.

This bit cannot be modified when PROT [1:0] bit-field in TIMERx\_CCHP register is 11 or 10.

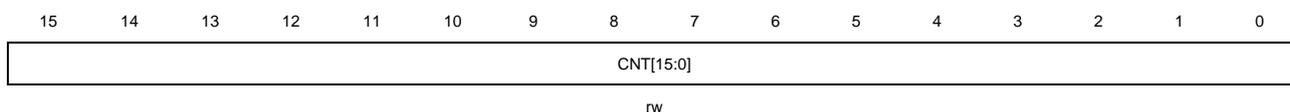
0	CH0EN	<p>Channel 0 capture/compare function enable</p> <p>When channel 0 is configured in output mode, setting this bit enables CH0_O signal in active state. When channel 0 is configured in input mode, setting this bit enables the capture event in channel0.</p> <p>0: Channel 0 disabled 1: Channel 0 enabled</p>
---	-------	---

### Counter register (TIMERx\_CNT)

Address offset: 0x24

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



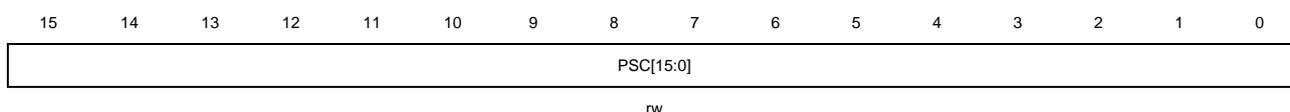
Bits	Fields	Descriptions
15:0	CNT[15:0]	This bit-field indicates the current counter value. Writing to this bit-field can change the value of the counter.

### Prescaler register (TIMERx\_PSC)

Address offset: 0x28

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



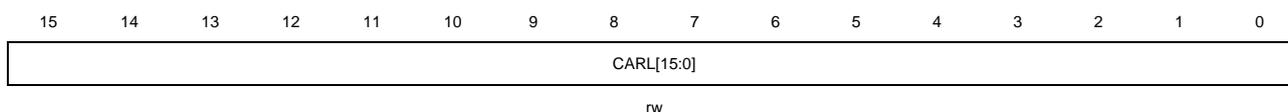
Bits	Fields	Descriptions
15:0	PSC[15:0]	<p>Prescaler value of the counter clock</p> <p>The PSC clock is divided by (PSC+1) to generate the counter clock. The value of this bit-field will be loaded to the corresponding shadow register at every update event.</p>

### Counter auto reload register (TIMERx\_CAR)

Address offset: 0x2C

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



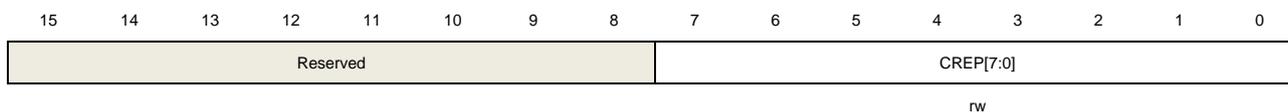
Bits	Fields	Descriptions
15:0	CARL[15:0]	Counter auto reload value This bit-filed specifies the auto reload value of the counter.

## Counter repetition register (TIMERx\_CREP)

Address offset: 0x30

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



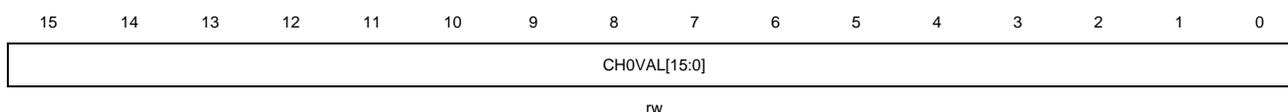
Bits	Fields	Descriptions
15:8	Reserved	Must be kept at reset value.
7:0	CREP[7:0]	Counter repetition value This bit-filed specifies the update event generation rate. Each time the repetition counter counting down to zero, an update event is generated. The update rate of the shadow registers is also affected by this bit-filed when these shadow registers are enabled.

## Channel 0 capture/compare value register (TIMERx\_CH0CV)

Address offset: 0x34

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



Bits	Fields	Descriptions
15:0	CH0VAL[15:0]	Capture or compare value of channel0 When channel 0 is configured in input mode, this bit-filed indicates the counter value corresponding to the last capture event. And this bit-filed is read-only. When channel 0 is configured in output mode, this bit-filed contains value to be

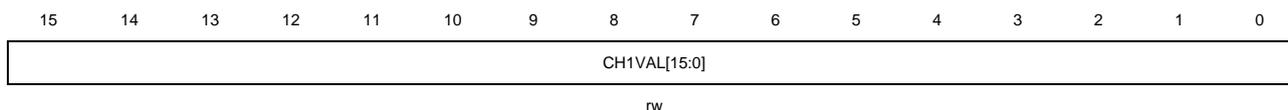
compared to the counter. When the corresponding shadow register is enabled, the shadow register updates every update event.

### Channel 1 capture/compare value register (TIMERx\_CH1CV)

Address offset: 0x38

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



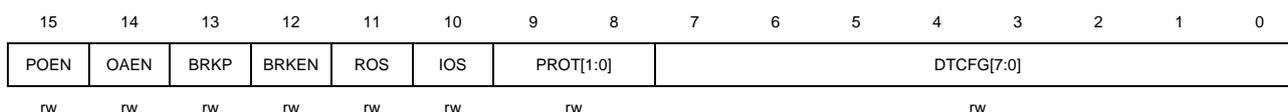
Bits	Fields	Descriptions
15:0	CH1VAL[15:0]	Capture or compare value of channel1  When channel 1 is configured in input mode, this bit-filed indicates the counter value corresponding to the last capture event. And this bit-filed is read-only.  When channel 1 is configured in output mode, this bit-filed contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates every update event.

### Channel complementary protection register (TIMERx\_CCHP)

Address offset: 0x44

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



Bits	Fields	Descriptions
15	POEN	Primary output enable  This bit s set by software or automatically by hardware depending on the OAEN bit. It is cleared asynchronously by hardware as soon as the break input is active. When one of channels is configured in output mode, setting this bit enables the channel outputs (CHx_O and CHx_ON) if the corresponding enable bits (CHxEN, CHxNEN in TIMERx_CHCTL2 register) have been set.  0: Channel outputs are disabled or forced to idle state. 1: Channel outputs are enabled.
14	OAEN	Output automatic enable  This bit specifies whether the POEN bit can be set automatically by hardware.  0: POEN can be not set by hardware.

		<p>1: POEN can be set by hardware automatically at the next update event, if the break input is not active.</p> <p>This bit can be modified only when PROT [1:0] bit-filed in TIMERx_CCHP register is 00.</p>
13	BRKP	<p>Break polarity</p> <p>This bit specifies the polarity of the BRKIN input signal.</p> <p>0: BRKIN input active low</p> <p>1; BRKIN input active high</p>
12	BRKEN	<p>Break enable</p> <p>This bit can be set to enable the BRKIN and CCS clock failure event inputs.</p> <p>0: Break inputs disabled</p> <p>1; Break inputs enabled</p> <p>This bit can be modified only when PROT [1:0] bit-filed in TIMERx_CCHP register is 00.</p>
11	ROS	<p>Run mode off-state configure</p> <p>When POEN bit is set, this bit specifies the output state for the channels which has a complementary output and has been configured in output mode.</p> <p>0: When POEN bit is set, the channel output signals (CHx_O/CHx_ON) are disabled.</p> <p>1: When POEN bit is set, the channel output signals (CHx_O/CHx_ON) are enabled, with relationship to CHxEN/CHxNEN bits in TIMERx_CHCTL2 register.</p> <p>This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 10 or 11.</p>
10	IOS	<p>Idle mode off-state configure</p> <p>When POEN bit is reset, this bit specifies the output state for the channels which has been configured in output mode.</p> <p>0: When POEN bit is reset, the channel output signals (CHx_O/CHx_ON) are disabled.</p> <p>1: When POEN bit is reset, he channel output signals (CHx_O/CHx_ON) are enabled, with relationship to CHxEN/CHxNEN bits in TIMERx_CHCTL2 register.</p> <p>This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 10 or 11.</p>
9:8	PROT[1:0]	<p>Complementary register protect control</p> <p>This bit-filed specifies the write protection property of registers.</p> <p>00: protect disable. No write protection.</p> <p>01: PROT mode 0.The ISOx/ISOxN bits in TIMERx_CTL1 register and the BRKEN/BRKP/OAEN/DTCFG bits in TIMERx_CCHP register are writing protected.</p> <p>10: PROT mode 1. In addition of the registers in PROT mode 0, the CHxP/CHxNP bits in TIMERx_CHCTL2 register (if related channel is configured in output mode) and the ROS/IOS bits in TIMERx_CCHP register are writing protected.</p> <p>11: PROT mode 2. In addition of the registers in PROT mode 1, the CHxCOMCTL/CHxCOMSEN bits in TIMERx_CHCTL0/1 registers (if the related channel is</p>

configured in output) are writing protected.

This bit-field can be written only once after the reset. Once the `TIMERx_CCHP` register has been written, this bit-field will be writing protected.

**7:0 DTCFG[7:0]** Dead time configure

This bit-field controls the value of the dead-time, which is inserted before the output transitions. The relationship between DTCFG value and the duration of dead-time is as follow:

DTCFG [7:5] =3'b0xx:  $DTvalue = DTCFG [7:0] \times t_{DT}, t_{DT} = t_{DTS}$ .

DTCFG [7:5] =3'b 10x:  $DTvalue = (64 + DTCFG [5:0]) \times t_{DT}, t_{DT} = t_{DTS} * 2$ .

DTCFG [7:5] =3'b 110:  $DTvalue = (32 + DTCFG [4:0]) \times t_{DT}, t_{DT} = t_{DTS} * 8$ .

DTCFG [7:5] =3'b 111:  $DTvalue = (32 + DTCFG [4:0]) \times t_{DT}, t_{DT} = t_{DTS} * 16$ .

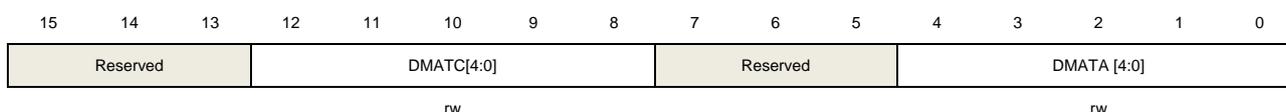
This bit can be modified only when `PROT [1:0]` bit-field in `TIMERx_CCHP` register is 00.

## DMA configuration register (TIMERx\_DMACFG)

Address offset: 0x48

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



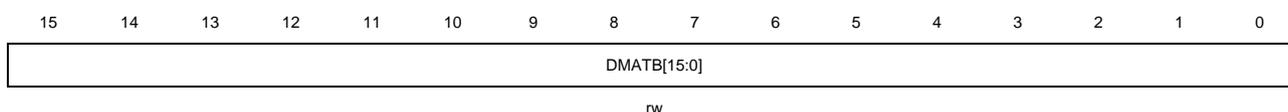
Bits	Fields	Descriptions
15:13	Reserved	Must be kept at reset value.
12:8	DMATC [4:0]	DMA transfer count This filed is defined the number of DMA will access(R/W) the register of <code>TIMERx_DMATB</code>
7:5	Reserved	Must be kept at reset value.
4:0	DMATA [4:0]	DMA transfer access start address This filed define the first address for the DMA access the <code>TIMERx_DMATB</code> . When access is done through the <code>TIMERx_DMA</code> address first time, this bit-field specifies the address you just access. And then the second access to the <code>TIMERx_DMATB</code> , you will access the address of start address + 0x4.  5'b0_0000: <code>TIMERx_CTL0</code> 5'b0_0001: <code>TIMERx_CTL1</code> ... In a word: Start Address = <code>TIMERx_CTL0</code> + <code>DMATA*4</code>

## DMA transfer buffer register (TIMERx\_DMATB)

Address offset: 0x4C

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



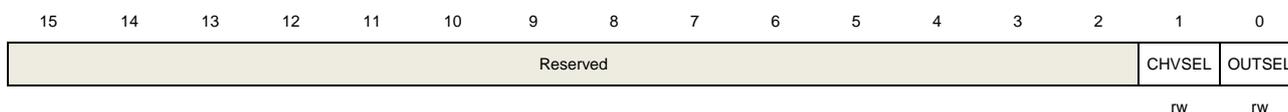
Bits	Fields	Descriptions
15:0	DMATB[15:0]	<p>DMA transfer buffer</p> <p>When a read or write operation is assigned to this register, the register located at the address range (Start Addr + Transfer Timer* 4) will be accessed.</p> <p>The transfer Timer is calculated by hardware, and ranges from 0 to DMATC.</p>

## Configuration register (TIMERx\_CFG) of GD32F170xx and GD32F190xx devices

Address offset: 0xFC

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



Bits	Fields	Descriptions
15:2	Reserved	Must be kept at reset value
1	CHVSEL	<p>Write CHxVAL register selection</p> <p>This bit-field set and reset by software.</p> <p>1: If write the CHxVAL register, the write value is same as the CHxVAL value, the write access ignored</p> <p>0: No effect</p>
0	OUTSEL	<p>The output value selection</p> <p>This bit-field set and reset by software</p> <p>1: If POEN and IOS is 0, the output disabled</p> <p>0: No effect</p>

## 15.5. General level4 timer (TIMERx, x=15,16)

### 15.5.1. Overview

The general level4 timer module (TIMER15,TIMER16) is a one-channel timer that supports both input capture and output compare. They can generate PWM signals to control motor or be used for power management applications. The general level4 timer has a 16-bit counter that can be used as an unsigned counter.

In addition, the general level4 timers can be programmed and be used for counting, their external events can be used to drive other timers.

Timer also includes a dead-time Insertion module which issuitable for motor control applications.

### 15.5.2. Characteristics

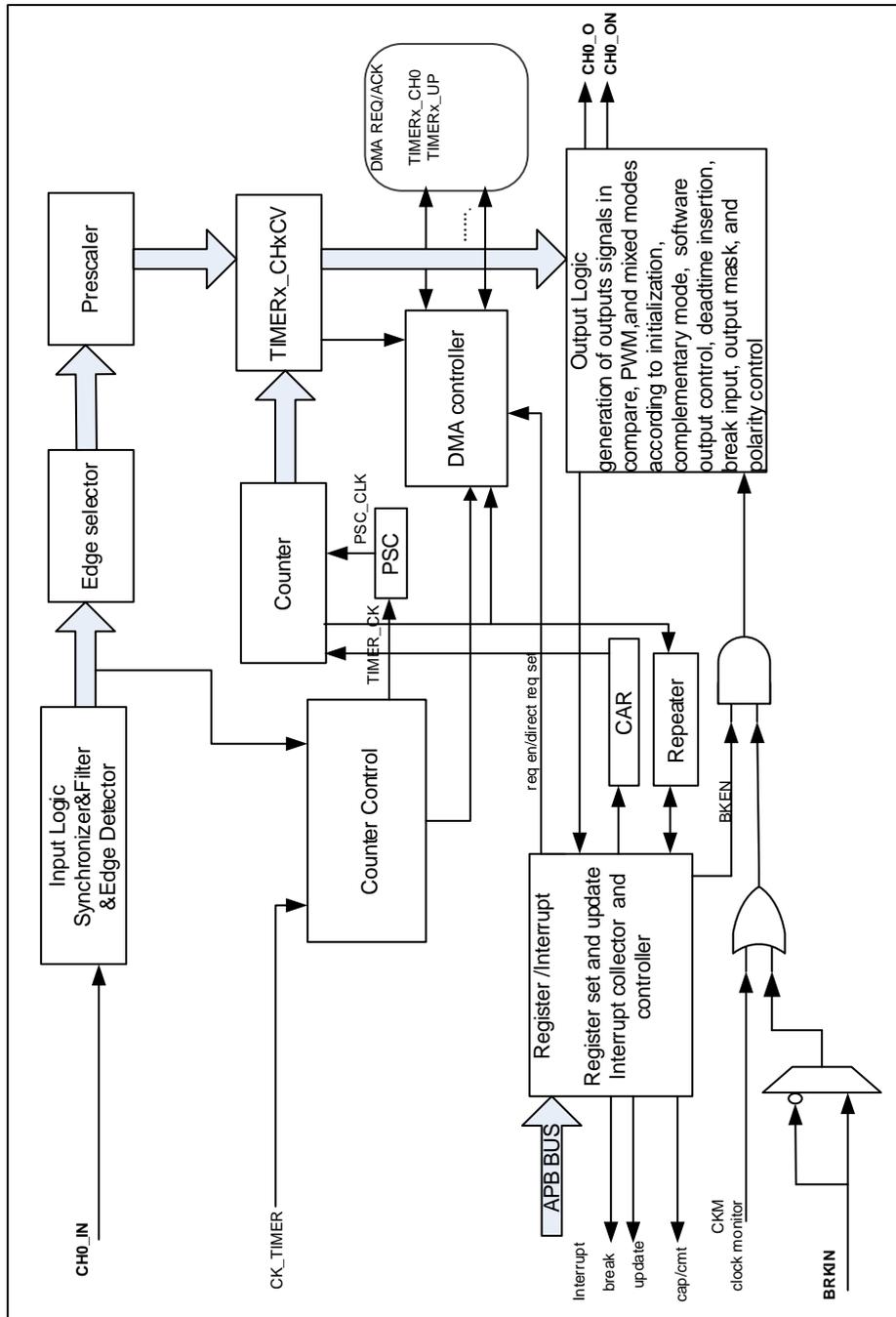
- Total channel num: 1.
- Counter width: 16 bit.
- Source of counter clock: internal clock.
- Counter modes: count up only.
- Programmable prescaler: 16 bit.The factor can be changed on the go.
- Each channel is user-configurable:  
input capture mode, output compare mode, programmable PWM mode, single pulse mode
- Programmable dead time insertion.
- Auto reload function.
- Programmable counter repetition function.
- Break input.
- Interrupt output or DMA request on: update, compare/capture event, and break input.

### 15.5.3. Block diagram

[Figure 15-73. General level4 timer block diagram](#) provides details of the internal

configuration of the general level4 timer.

**Figure 15-73. General level4 timer block diagram**



### 15.5.4. Function overview

#### Clock selection

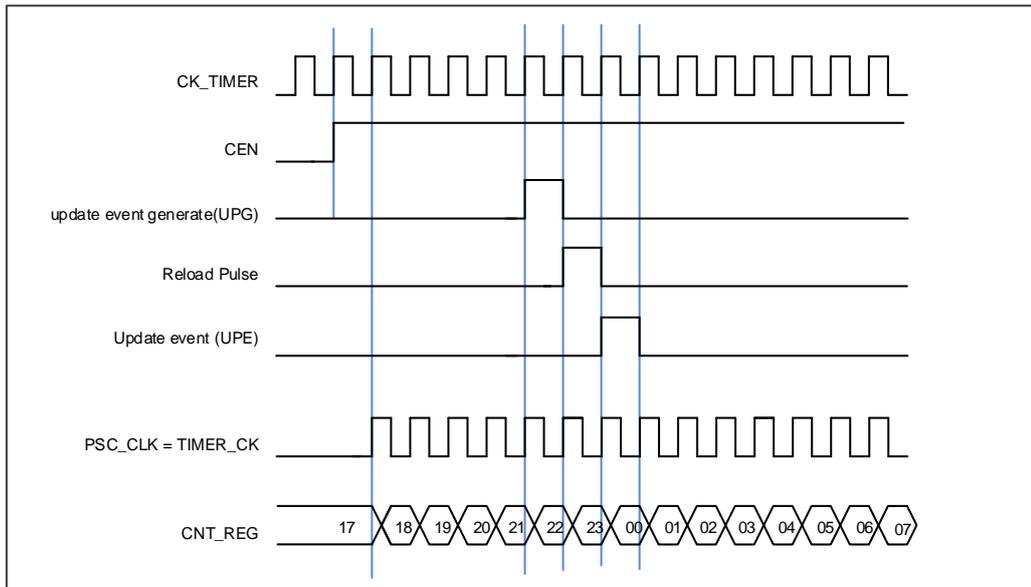
The general level4 TIMER can only being clocked by the CK\_TIMER.

■ Internal timer clock CK\_TIMER which is from module RCU

The general level4 TIMER has only one clock source which is the internal CK\_TIMER, used to drive the counter prescaler. When the CEN is set, the CK\_TIMER will be divided by PSC value to generate PSC\_CLK.

The TIMER\_CK, driven counter's prescaler to count, is equal to CK\_TIMER which is from RCU

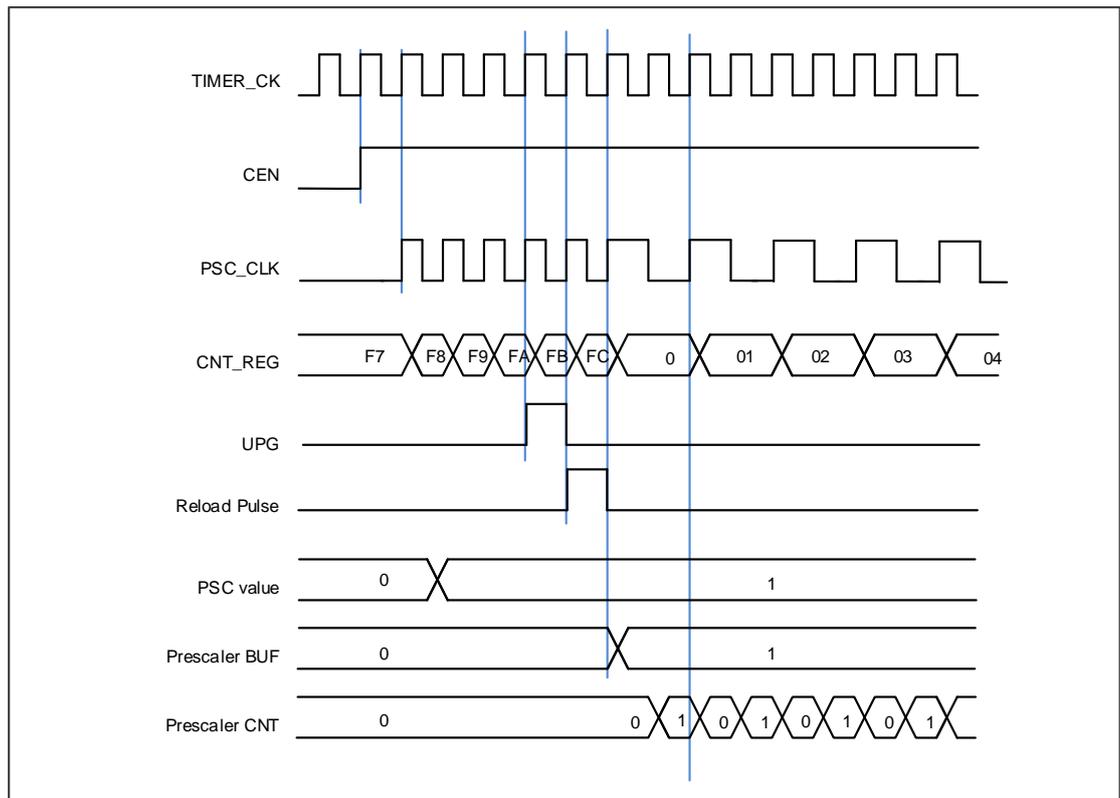
**Figure 15-74. Normal mode, internal clock divided by 1**



## Prescaler

The prescaler can divide the timer clock (TIMER\_CK) to a counter clock (PSC\_CLK) by any factor between 1 and 65536. It is controlled by prescaler register (TIMERx\_PSC) which can be changed on the go but is taken into account at the next update event.

**Figure 15-75. Counter timing diagram with prescaler division change from 1 to 2**



### Up counting mode

In this mode, the counter counts up continuously from 0 to the counter-reload value, which is defined in the `TIMERx_CAR` register, in a count-up direction. Once the counter reaches the counter reload value, the counter restarts from 0. If the repetition counter is set, the update events will be generated after  $(\text{TIMERx\_CREP}+1)$  times of overflow. Otherwise the update event is generated each time when overflows. The counting direction bit `DIR` in the `TIMERx_CTL0` register should be set to 0 for the up counting mode.

Whenever, if the update event software trigger is enabled by setting the `UPG` bit in the `TIMERx_SWEVG` register, the counter value will be initialized to 0 and generates an update event.

If set the `UPDIS` bit in `TIMERx_CTL0` register, the update event is disabled.

When an update event occurs, all the registers (repetition counter, auto reload register, prescaler register) are updated.

[Figure 15-76. Up-counter timechart, PSC=0/1](#) show some examples of the counter

behavior for different clock prescaler factor when `TIMERx_CAR=0x63`.

**Figure 15-76. Up-counter timechart, PSC=0/1**

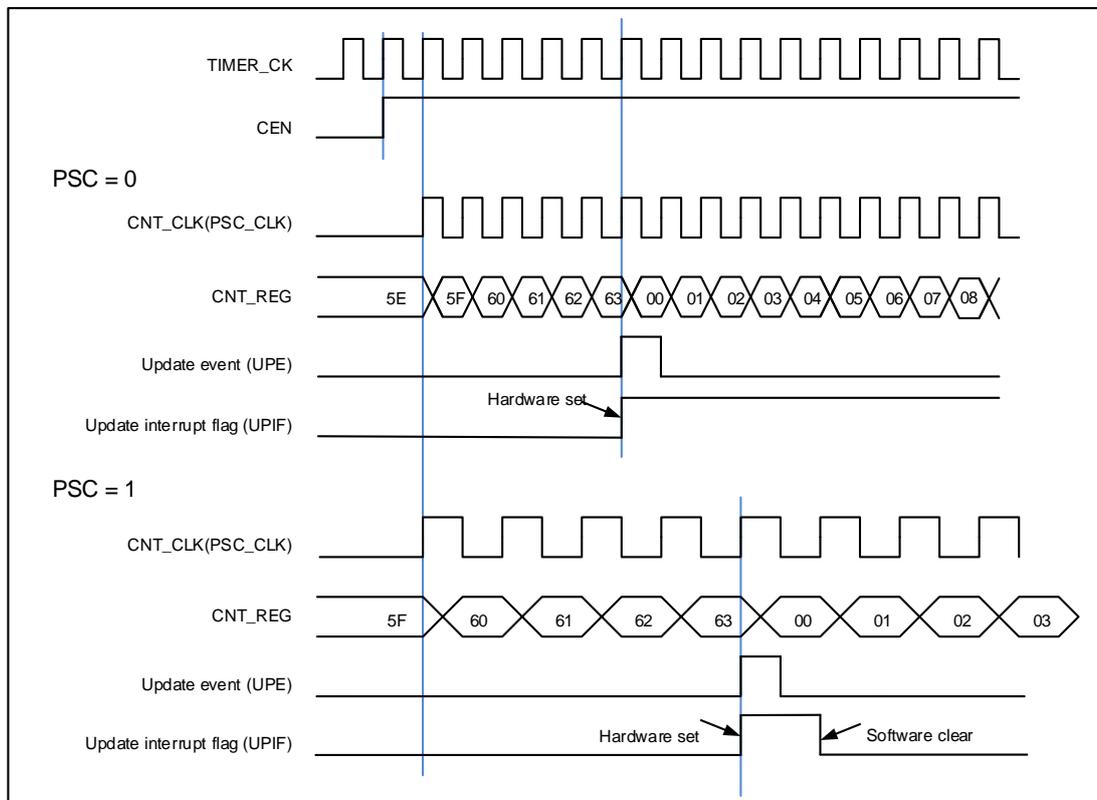
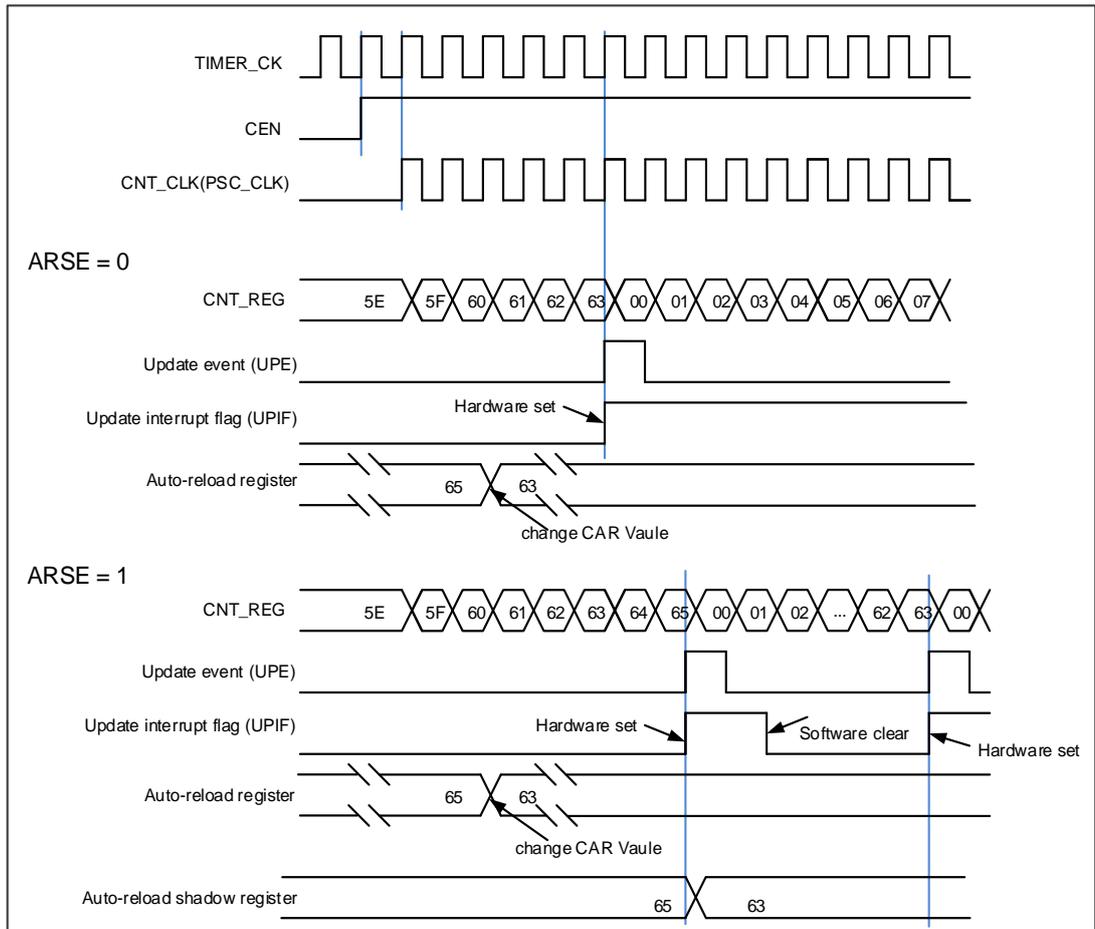


Figure 15-77. Up-counter timechart, change `TIMERx_CAR` on the go

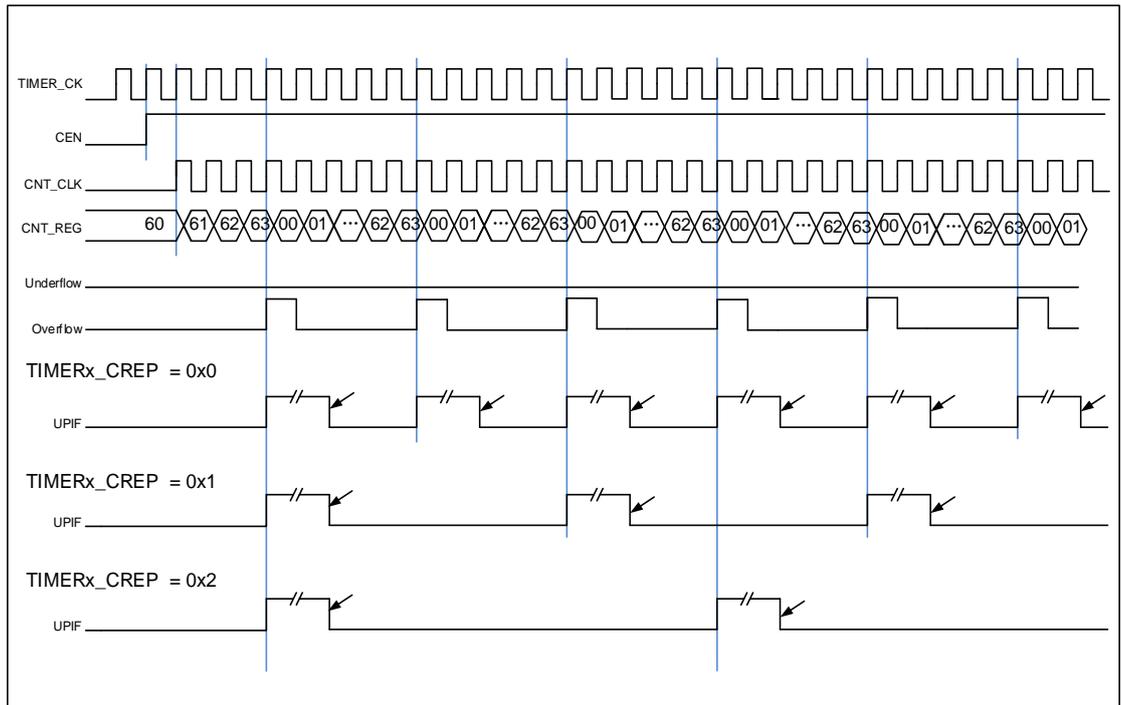


### Counter repetition

Counter repetition is used to generate update event or update the timer registers only after a given number (N+1) of cycles of the counter, where N is CREP in `TIMERx_CREP` register. The repetition counter is decremented at each counter overflow in up-counting mode.

Setting the UPG bit in the `TIMERx_SWEVG` register will reload the content of CREP in `TIMERx_CREP` register and generate an update event.

Figure 15-78. Repetition timechart for up-counter



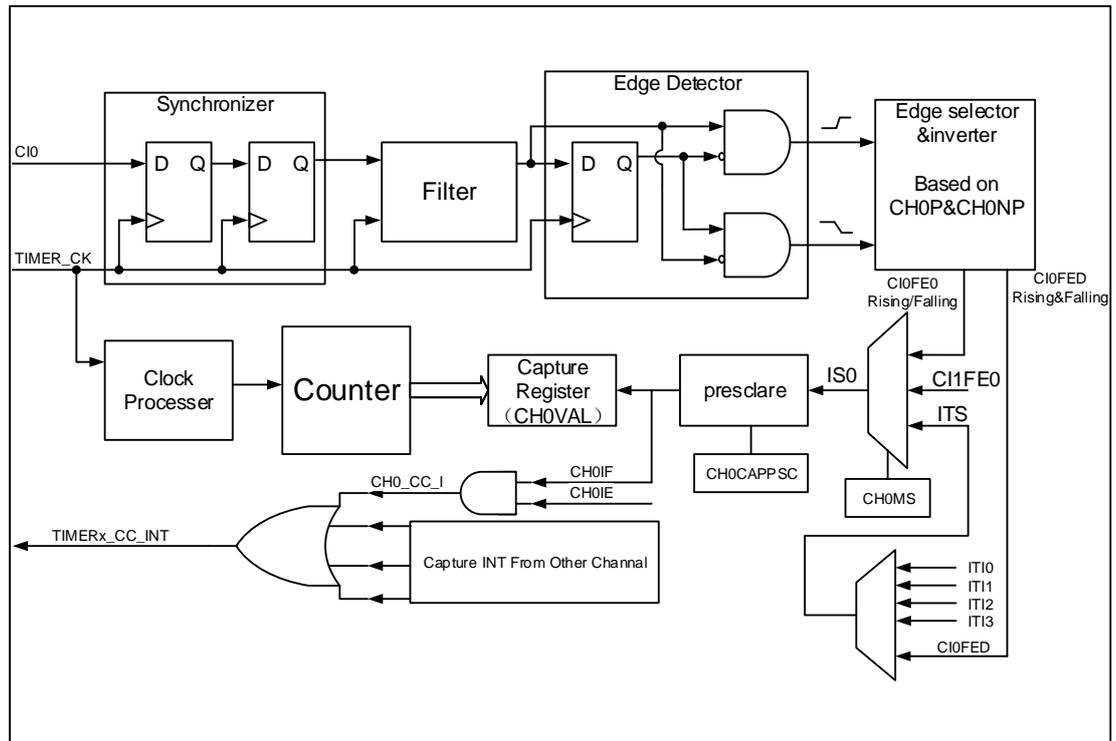
### Capture/compare channels

The general level4 timer has one independent channels which can be used as capture inputs or compare match outputs. Each channel is built around a channel capture compare register including an input stage, channel controller and an output stage.

- #### Input capture mode

Capture mode allows the channel to perform measurements such as pulse timing, frequency, period, duty cycle and so on. The input stage consists of a digital filter, a channel polarity selection, edge detection and a channel prescaler. When a selected edge occurs on the channel input, the current value of the counter is captured into the **TIMERx\_CHxCV** register, at the same time the **CHxIF** bit is set and the channel interrupt is generated if enabled by **CHxIE = 1**.

Figure 15-79. Input capture logic



Channels' input signals (C1x) is the TIMERx\_CHx signal. First, the channel input signal (C1x) is synchronized to TIMER\_CK domain, and then sampled by a digital filter to generate a filtered input signal. Then through the edge detector, the rising and falling edge are detected. You can select one of them by CHxP. One more selector is for the other channel and trig, controlled by CHxMS. The IC\_prescaler make several the input event generate one effective capture event. On the capture event, CHxVAL will restore the value of Counter.

So the process can be divided to several steps as below:

**Step1:** Filter configuration. (CHxCAPFLT in TIMERx\_CHCTL0)

Based on the input signal and requested signal quality, configure compatible CHxCAPFLT.

**Step2:** Edge selection. (CHxP/CHxNP in TIMERx\_CHCTL2)

Rising or falling edge, choose one by CHxP/CHxNP.

**Step3:** Capture source selection. (CHxMS in TIMERx\_CHCTL0)

As soon as you select one input capture source by CHxMS, you have set the channel to input mode (CHxMS!=0x0) and TIMERx\_CHxCV cannot be written any more.

**Step4:** Interrupt enable. (CHxIE and CHxDEN in TIMERx\_DMAINTEN)

Enable the related interrupt enable; you can got the interrupt and DMA request.

**Step5:** Capture enables. (CHxEN in TIMERx\_CHCTL2)

**Result:** when you wanted input signal is got, TIMERx\_CHxCV will be set by counter's value.

And CHxIF is asserted. If the CHxIF is high, the CHxOF will be asserted also. The interrupt and DMA request will be asserted based on the configuration of CHxIE and CHxDEN in TIMERx\_DMAINTEN

**Direct generation:** if you want to generate a DMA request or Interrupt, you can set CHxG by software directly.

### ■ Output compare mode

In output compare mode, the TIMERx can generate timed pulses with programmable position, polarity, duration and frequency. When the counter matches the value in the CHxVAL register of an output compare channel, the channel (n) output can be set, cleared, or toggled based on CHxCOMCTL. When the counter reaches the value in the CHxVAL register, the CHxIF bit is set and the channel (n) interrupt is generated if CHxIE = 1. And the DMA request will be assert, if CHxDEN =1.

So the process can be divided to several steps as below:

**Step1:** Clock Configuration. Such as clock source, clock prescaler and so on.

**Step2:** Compare mode configuration.

- \* Set the shadow enable mode by CHxCOMSEN
- \* Set the output mode (Set/Clear/Toggle) by CHxCOMCTL.
- \* Select the active high polarity by CHxP/CHxNP
- \* Enable the output by CHxEN

**Step3:** Interrupt/DMA-request enables configuration by CHxIE/CHxDEN

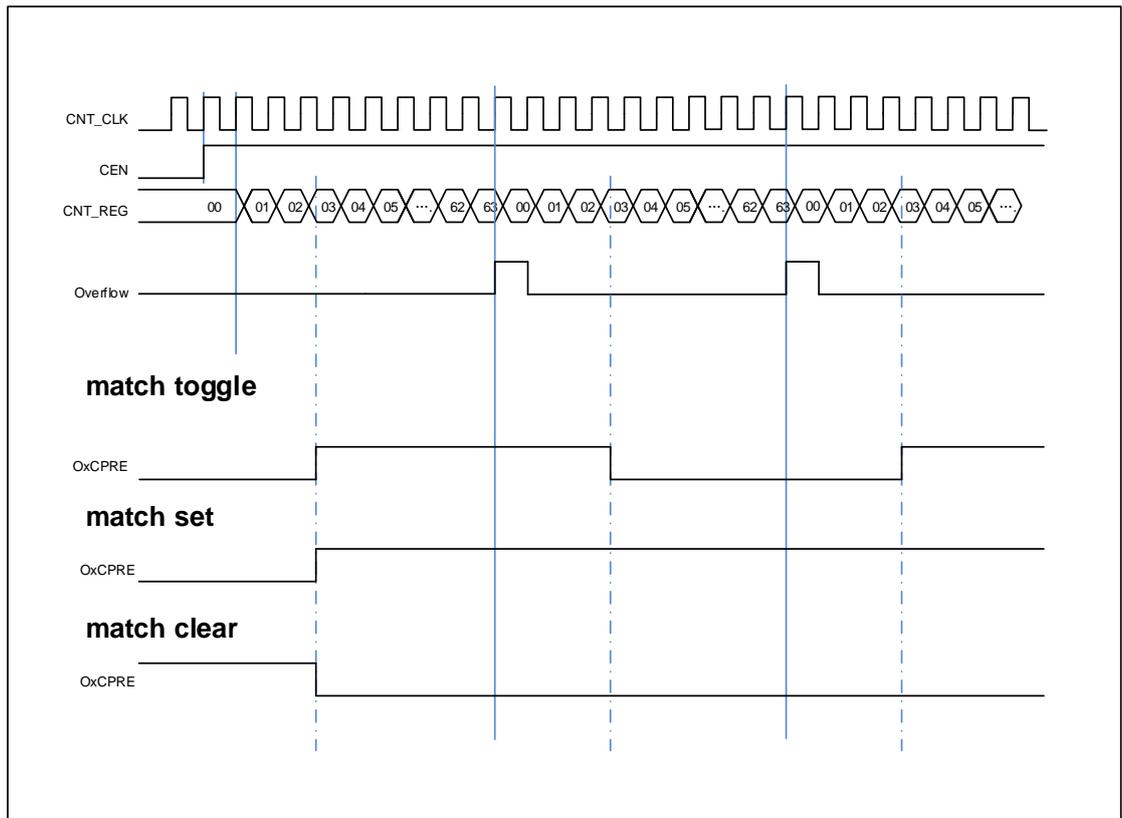
**Step4:** Compare output timing configuration by TIMERx\_CAR and TIMERx\_CHxCV

About the CHxVAL; you can change it on the go to meet the waveform you expected.

**Step5:** Start the counter by CEN.

The timechart below show the three compare modes toggle/set/clear. CAR=0x63, CHxVAL=0x3

Figure 15-80. Output-compare under three modes



### PWM mode

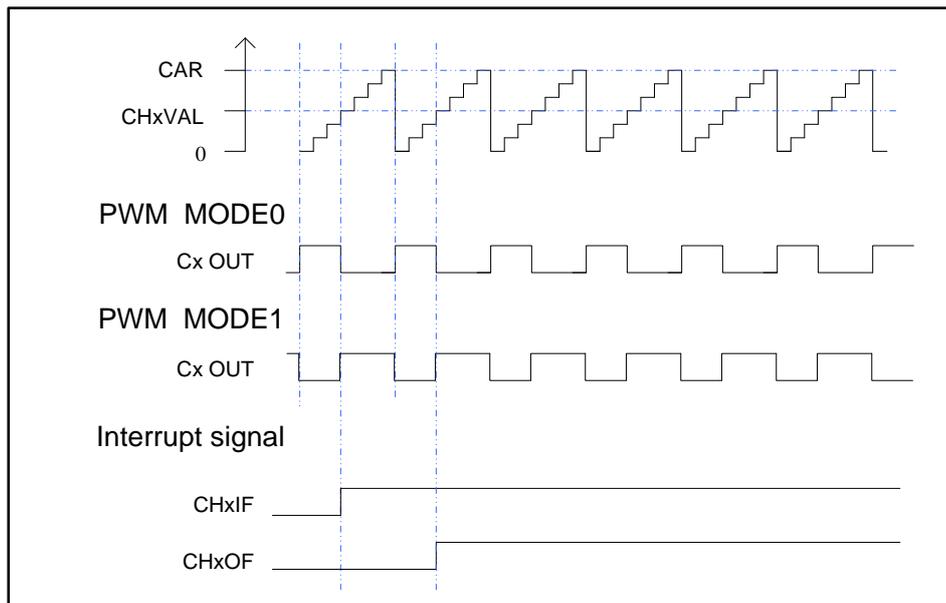
In the output PWM mode (by setting the CHxCOMCTL bits to 3'b110 (PWM mode0) or to 3'b111(PWM mode1), the channel can generate PWM waveform according to the TIMERx\_CAR registers and TIMERx\_CHxCV registers.

The period is determined by TIMERx\_CAR and duty cycle is determined by TIMERx\_CHxCV. [Figure 15-81. PWM mode timechart](#) shows the PWM output mode and interrupts waveform.

If TIMERx\_CHxCV is greater than TIMERx\_CAR, the output will be always active under PWM mode0 (CHxCOMCTL==3'b110).

And if TIMERx\_CHxCV is equal to zero, the output will be always inactive under PWM mode0 (CHxCOMCTL==3'b110).

**Figure 15-81. PWM mode timechart**



### Channel output reference signal

When the TIMERx is used in the compare match output mode, the OxCPRE signal (Channel x Output prepare signal) is defined by setting the CHxCOMCTL field. The OxCPRE signal has several types of output function. These include, keeping the original level by setting the CHxCOMCTL field to 0x00, set to 1 by setting the CHxCOMCTL field to 0x01, set to 0 by setting the CHxCOMCTL field to 0x02 or signal toggle by setting the CHxCOMCTL field to 0x03 when the counter value matches the content of the TIMERx\_CHxCV register.

The PWM mode 0 and PWM mode 1 outputs are also another kind of OxCPRE output which is setup by setting the CHxCOMCTL field to 0x06/0x07. In these modes, the OxCPRE signal level is changed according to the counting direction and the relationship between the counter value and the TIMERx\_CHxCV content. With regard to a more detail description refer to the relative bit definition.

Another special function of the OxCPRE signal is a forced output which can be achieved by setting the CHxCOMCTL field to 0x04/0x05. Here the output can be forced to an inactive/active level irrespective of the comparison condition between the counter and the TIMERx\_CHxCV values.

### Outputs complementary

Function of complementary is for a pair of CHx\_O and CHx\_ON. Those two output signals cannot be active at the same time. The TIMERx has only 1 channel have this function. The complementary signals CHx\_O and CHx\_ON are controlled by a group of parameters: the CHxEN and CHxNEN bits in the TIMERx\_CHCTL2 register and the POEN, ROS, IOS, ISOx and ISOxN bits in the TIMERx\_CCHP and TIMERx\_CTL1 registers. The outputs polarity is determined by CHxP and CHxNP bits in the TIMERx\_CHCTL2 register.

**Table 15-11. Complementary outputs controlled by parameters**

Complementary Parameters					Output Status			
POEN	ROS	IOS	CHxEN	CHxNEN	CHx_O	CHx_ON		
0	0/1	0	0	0	CHx_O / CHx_ON = LOW CHx_O / CHx_ON output disable.			
				1	CHx_O = CHxP CHx_ON = CHxNP			
			1	0	CHx_O/CHx_ON output disable.			
				1	If clock is enable: CHx_O = ISOx CHx_ON = ISOxN			
		1	0	0	0	CHx_O = CHxP CHx_ON = CHxNP CHx_O/CHx_ON output disable.		
					1	CHx_O = CHxP CHx_ON = CHxNP		
			1	0	CHx_O/CHx_ON output enable.			
				1	If clock is enable: CHx_O = ISOx CHx_ON = ISOxN			
1	0	0/1	0	0	CHx_O/CHx_ON = LOW CHx_O/CHx_ON output disable.			
				1	CHx_O = LOW CHx_O output disable.	CHx_ON=OxCPRE ⊕ CHxNP CHx_ON output enable		
			1	0	CHx_O=OxCPRE⊕CHxP CHx_O output enable		CHx_ON = LOW CHx_ON output disable.	
				1	CHx_O=OxCPRE⊕CHxP CHx_O output enable		CHx_ON=OxCPRE⊕CHxNP CHx_ON output enable	
			1	0	0	CHx_O = CHxP CHx_O output disable.		CHx_ON = CHxNP CHx_ON output disable.
					1	CHx_O = CHxP CHx_O output enable		CHx_ON=OxCPRE⊕CHxNP CHx_ON output enable
	1	0		CHx_O=OxCPRE⊕CHxP CHx_O output enable		CHx_ON = CHxNP CHx_ON output enable.		
		1		CHx_O=OxCPRE⊕CHxP CHx_O output enable		CHx_ON=OxCPRE⊕CHxNP CHx_ON output enable.		

### Dead time insertion

The dead time insertion is enabled when both CHxEN and CHxNEN are 1'b1, and set POEN is also necessary. The field named DTCFG defines the dead time delay that can be used for channel 1. The detail about the delay time, refer to the register TIMERx\_CCHP.

The dead time delay insertion ensures that no two complementary signals drive the active state at the same time.

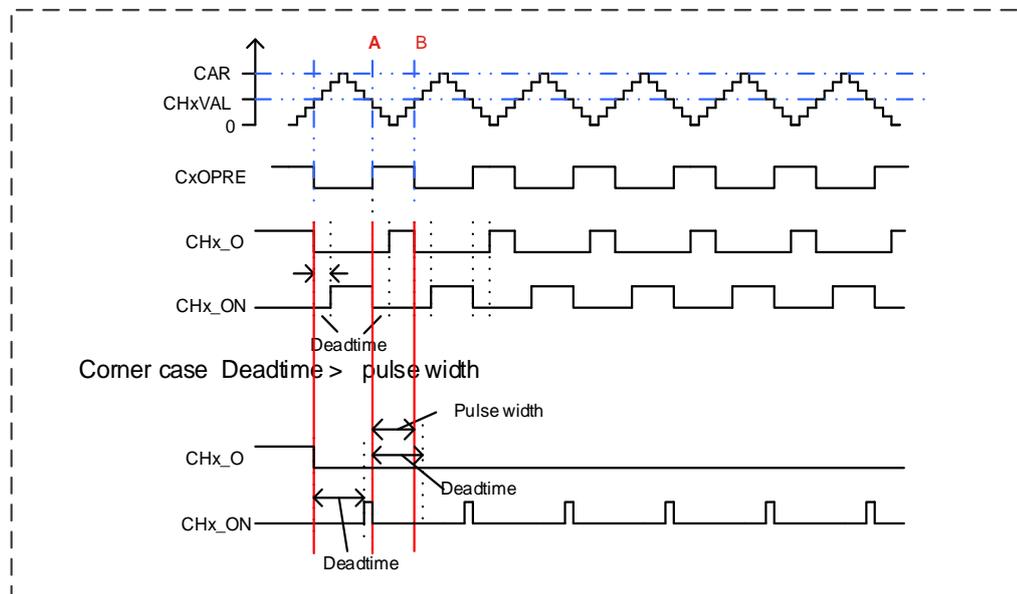
When the channel (x) match (TIMERx counter = CHxVAL) occurs, OxCPRE will be toggled because under PWM0 mode. At point A in the [Figure 15-82. Complementary output with dead-time insertion](#), CHx\_O signal remains at the low value until the end of the deadtime delay, while CHx\_ON will be cleared at once. Similarly, At point B when counter match (counter = CHxVAL) occurs again, OxCPRE is cleared, CHx\_O signal will be cleared at once, while CHx\_ON signal remains at the low value until the end of the dead time delay.

Sometimes, we can see corner cases about the dead time insertion. For example:

The dead time delay is greater than or equal to the CHx\_O duty cycle, then the CHx\_O signal is always the inactive value. (as show in the [Figure 15-82. Complementary output with dead-time insertion](#).)

The dead time delay is greater than or equal to the CHx\_ON duty cycle, then the CHx\_ON signal is always the inactive value.

**Figure 15-82. Complementary output with dead-time insertion.**



### Break function

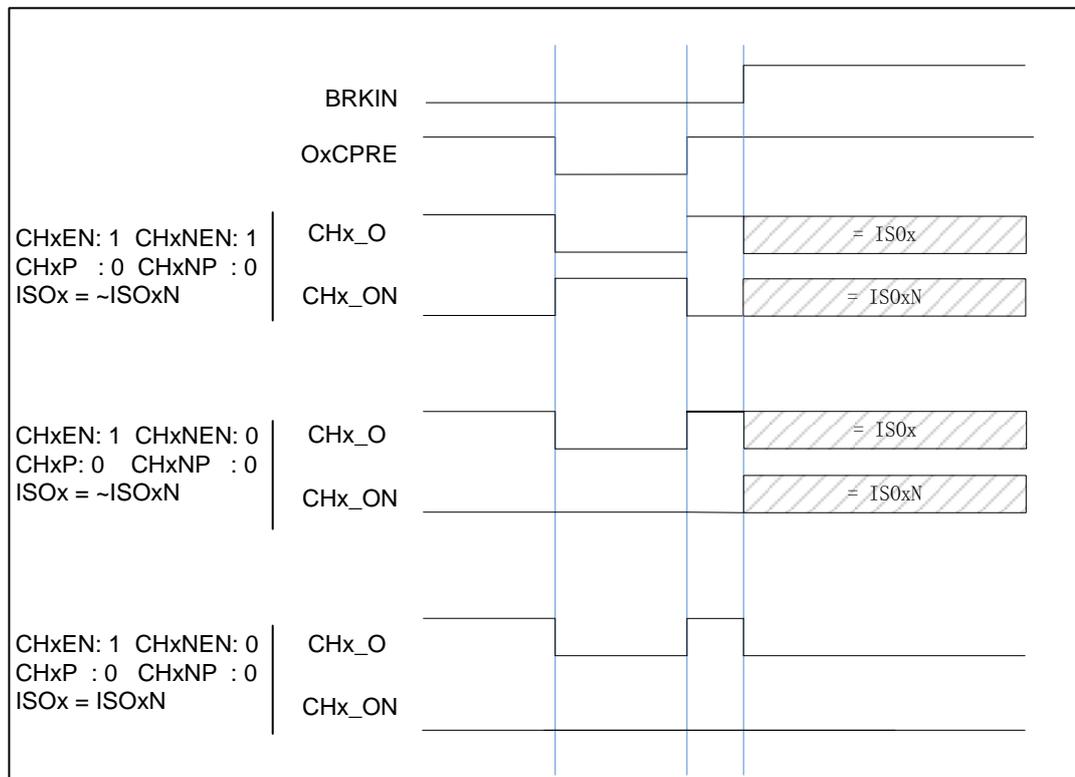
In this function, the output CHx\_O and CHx\_ON are controlled by the POEN, IOS and ROS bits in the TIMERx\_CCHP register, ISOx and ISOxN bits in the TIMERx\_CTL1 register and

cannot be set both to active level when break occurs. The break sources are input break pin and HXTAL stuck event by Clock Monitor (CKM) in RCU. The break function enabled by setting the BRKEN bit in the TIMERx\_CCHP register. The break input polarity is setting by the BRKP bit in TIMERx\_CCHP.

When a break occurs, the POEN bit is cleared asynchronously, the output CHx\_O and CHx\_ON are driven with the level programmed in the ISOx bit and ISOxN in the TIMERx\_CTL1 register as soon as POEN is 0. If IOS is 0 then the timer releases the enable output else the enable output remains high. The complementary outputs are first put in reset state, and then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the ISOx and ISOxN bits after a dead-time.

When a break occurs, the BRKIF bit in the TIMERx\_INTF register is set. If BRKIE is 1, an interrupt generated.

**Figure 15-83. Output behavior in response to a break(The break high active)**



## Single pulse mode

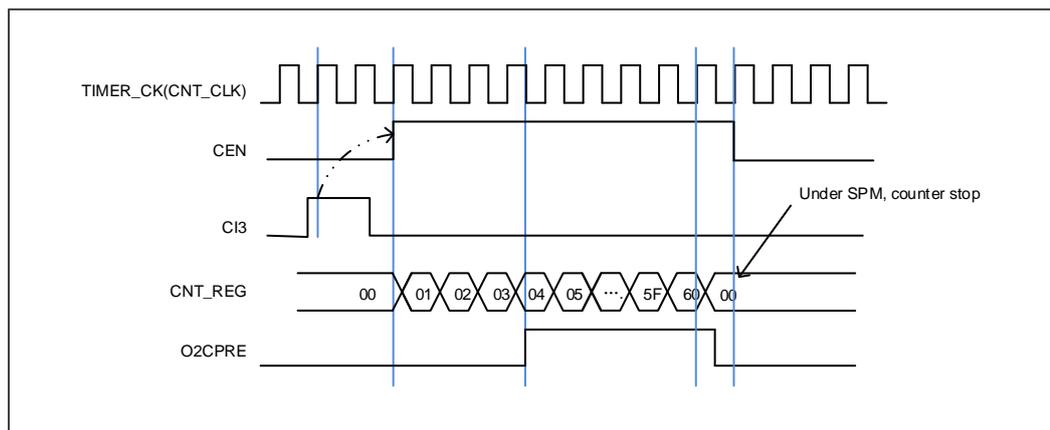
Single pulse mode is opposite to the repetitive mode, which can be enabled by setting SPM in TIMERx\_CTL0. When you set SPM, the counter will be clear and stop when the next update event automatically. In order to get pulse waveform, you can set the TIMERx to PWM mode or compare by CHxCOMCTL.

Once the timer is set to operate in the single pulse mode, it is not necessary to set the timer enable bit CEN in the TIMERx\_CTL0 register to 1 to enable the counter. The trigger to

generate a pulse can be sourced from the trigger signals edge or by setting the CEN bit to 1 using software. Setting the CEN bit to 1 or a trigger from the trigger signals edge can generate a pulse and then keep the CEN bit at a high state until the update event occurs or the CEN bit is written to 0 by software. If the CEN bit is cleared to 0 using software, the counter will be stopped and its value held. If the CEN bit is automatically cleared to 0 by a hardware update event, the counter will be reinitialized.

In the single pulse mode, the trigger active edge which sets the CEN bit to 1 will enable the counter. However, there exist several clock delays to perform the comparison result between the counter value and the `TIMERx_CHxCV` value. In order to reduce the delay to a minimum value, the user can set the `CHxCOMFEN` bit in each `TIMERx_CHCTL0/1` register. After a trigger rising occurs in the single pulse mode, the `OxCPRE` signal will immediately be forced to the state which the `OxCPRE` signal will change to, as the compare match event occurs without taking the comparison result into account. The `CHxCOMFEN` bit is available only when the output channel is configured to operate in the PWM0 or PWM1 output mode and the trigger source is derived from the trigger signal.

**Figure 15-84. Single pulse mode `TIMERx_CHxCV = 0x04` `TIMERx_CAR=0x60`**



## Timer DMA mode

Timer's DMA mode is the function that configures timer's register by DMA module. The relative registers are `TIMERx_DMACFG` and `TIMERx_DMATB`. Of course, you have to enable a DMA request which will be asserted by some internal event. When the interrupt event was asserted, `TIMERx` will send a request to DMA, which is configured to M2P mode and PADDR is `TIMERx_DMATB`, then DMA will access the `TIMERx_DMATB`. In fact, register `TIMERx_DMATB` is only a buffer; timer will map the `TIMERx_DMATB` to an internal register, appointed by the field of `DMATA` in `TIMERx_DMACFG`. If the field of `DMATC` in `TIMERx_DMACFG` is 0(1 transfer), then the timer's DMA request is finished. While if `TIMERx_DMATC` is not 0, such as 3( 4 transfers), then timer will send 3 more requests to DMA, and DMA will access timer's registers `DMATA+0x4`, `DMATA+0x8`, `DMATA+0xc` at the next 3 accesses to `TIMERx_DMATB`. In one word, one time DMA internal interrupt event assert, `DMATC+1` times request will be send by `TIMERx`.

If one more time DMA request event coming, TIMERx will repeat the process as above.

### **Timer debug mode**

When the Cortex™-M3 halted, and the TIMERx\_HOLD configuration bit in DBG\_CTL1 register set to 1, the TIMERx counter stops.

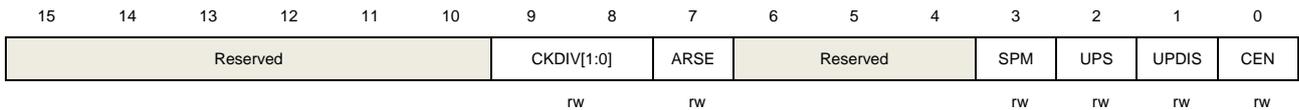
## 15.5.5. TIMERx registers(x=15,16)

### Control register 0 (TIMERx\_CTL0)

Address offset: 0x00

Reset value: 0x0000

This register can be accessed by half-word(16-bit) or word(32-bit)



Bits	Fields	Descriptions
15:10	Reserved	Must be kept at reset value
9:8	CKDIV[1:0]	<p>Clock division</p> <p>The CKDIV bits can be configured by software to specify division ratio between the timer clock (TIMER_CK) and the dead-time and sampling clock (DTS), which is used by the dead-time generators and the digital filters.</p> <p>00: <math>f_{DTS}=f_{TIMER\_CK}</math>            01: <math>f_{DTS}= f_{TIMER\_CK} /2</math>            10: <math>f_{DTS}= f_{TIMER\_CK} /4</math>            11: Reserved</p>
7	ARSE	<p>Auto-reload shadow enable</p> <p>0: The shadow register for TIMERx_CAR register is disabled            1: The shadow register for TIMERx_CAR register is enabled</p>
6:4	Reserved	Must be kept at reset value
3	SPM	<p>Single pulse mode.</p> <p>0: Counter continues after update event.            1: The CEN is cleared by hardware and the counter stops at next update event.</p>
2	UPS	<p>Update source</p> <p>This bit is used to select the update event sources by software.</p> <p>0: Any of the following events generate an update interrupt or DMA request:           <ul style="list-style-type: none"> <li>- The UPG bit is set</li> <li>- The counter generates an overflow or underflow event</li> <li>- The slave mode controller generates an update event.</li> </ul>           1: Only counter overflow/underflow generates an update interrupt or DMA request.</p>
1	UPDIS	<p>Update disable.</p> <p>This bit is used to enable or disable the update event generation.</p> <p>0: update event enable. The update event is generate and the buffered registers are loaded with their preloaded values when one of the following events occurs:</p>

- The UPG bit is set
- The counter generates an overflow or underflow event
- The slave mode controller generates an update event.

1: update event disable. The buffered registers keep their value, while the counter and the prescaler are reinitialized if the UG bit is set or if the slave mode controller generates a hardware reset event.

0            CEN

Counter enable

0: Counter disable

1: Counter enable

The CEN bit must be set by software when timer works in external clock, pause mode and encoder mode. While in event mode, the hardware can set the CEN bit automatically.

## Control register 1 (TIMERx\_CTL1)

Address offset: 0x04

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						ISO0N	ISO0	Reserved				DMAS	CCUC	Reserved	CCSE
						rw	rw					rw	rw		

Bits	Fields	Descriptions
15:10	Reserved	Must be kept at reset value
9	ISO0N	Idle state of channel 0 complementary output 0: When POEN bit is reset, CH0_ON is set low. 1: When POEN bit is reset, CH0_ON is set high This bit can be modified only when PROT [1:0] bits in TIMERx_CCHP register is 00.
8	ISO0	Idle state of channel 0 output 0: When POEN bit is reset, CH0_O is set low. 1: When POEN bit is reset, CH0_O is set high The CH0_O output changes after a dead-time if CH0_ON is implemented. This bit can be modified only when PROT [1:0] bits in TIMERx_CCHP register is 00.
7:4	Reserved	Must be kept at reset value
3	DMAS	DMA request source selection 0: DMA request of channel x is sent when capture/compare event occurs. 1: DMA request of channel x is sent when update event occurs.
2	CCUC	Commutation control shadow register update control When the commutation control shadow enable (for CHxEN, CHxNEN and

CHxCOMCTL bits) are set (CCSE=1), these shadow registers update are controlled as below:

0: The shadow registers update by when CMTG bit is set.

1: The shadow registers update by when CMTG bit is set or a rising edge of TRGI occurs.

When a channel does not have a complementary output, this bit has no effect.

1 Reserved

Must be kept at reset value.

0 CCSE

Commutation control shadow enable

0: The shadow registers for CHxEN, CHxNEN and CHxCOMCTL bits are disabled.

1: The shadow registers for CHxEN, CHxNEN and CHxCOMCTL bits are enabled.

After these bits have been written, they are updated based when commutation event coming.

When a channel does not have a complementary output, this bit has no effect.

## DMA and interrupt enable register (TIMERx\_DMAINTEN)

Address offset: 0x0C

Reset value: 0x0000

This register can be accessed by half-word(16-bit) or word(32-bit)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CH0DEN	UPDEN	BRKIE	Reserved	CMTIE	Reserved			CHOIE	UPIE
						rw	rw	rw	rw			rw	rw		

Bits	Fields	Descriptions
15:10	Reserved	Must be kept at reset value
9	CH0DEN	Channel 0 capture/compare DMA request enable 0: disabled 1: enabled
8	UPDEN	Update DMA request enable 0: disabled 1: enabled
7	BRKIE	Break interrupt enable 0: disabled 1: enabled
6	Reserved	Must be kept at reset value
5	CMTIE	Commutation interrupt enable 0: disabled 1: enabled

4:2	Reserved	Must be kept at reset value
1	CH0IE	Channel 0 capture/compare interrupt enable 0: disabled 1: enabled
0	UPIE	Update interrupt enable 0: disabled 1: enabled

### Interrupt flag register (TIMERx\_INTF)

Address offset: 0x10

Reset value: 0x0000

This register can be accessed by half-word(16-bit) or word(32-bit)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CH0OF	Reserved.	BRKIF	Reserved	CMTIF	Reserved.			CH0IF	UPIF
						rc_w0		rc_w0		rc_w0				rc_w0	rc_w0

Bits	Fields	Descriptions
15:10	Reserved	Must be kept at reset value
9	CH0OF	Channel 0 over capture flag When channel 0 is configured in input mode, this flag is set by hardware when a capture event occurs while CH0IF flag has already been set. This flag is cleared by software. 0: No over capture interrupt occurred 1: Over capture interrupt occurred
8	Reserved	Must be kept at reset value.
7	BRKIF	Break interrupt flag This flag is set by hardware when the break input goes active, and cleared by software if the break input is not active. 0: No active level break has been detected. 1: An active level has been detected.
6	Reserved	Must be kept at reset value
5	CMTIF	Channel commutation interrupt flag This flag is set by hardware when channel's commutation event occurs, and cleared by software 0: No channel commutation interrupt occurred 1: Channel commutation interrupt occurred
4:2	Reserved	Must be kept at reset value
1	CH0IF	Channel 0 's capture/compare interrupt flag

This flag is set by hardware and cleared by software. When channel 0 is in input mode, this flag is set when a capture event occurs. When channel 0 is in output mode, this flag is set when a compare event occurs.

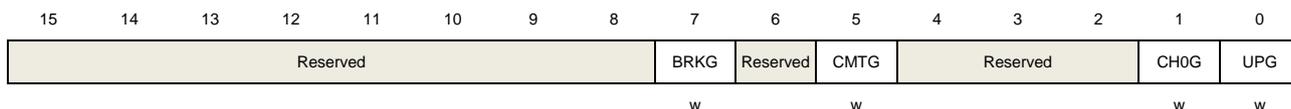
- 0: No Channel 0 interrupt occurred
- 1: Channel 0 interrupt occurred

0 UPIF Update interrupt flag  
 This bit is set by hardware on an update event and cleared by software.  
 0: No update interrupt occurred  
 1: Update interrupt occurred

### Software event generation register (TIMERx\_SWEVG)

Address offset: 0x14  
 Reset value: 0x0000

This register can be accessed by half-word(16-bit) or word(32-bit)



Bits	Fields	Descriptions
15:8	Reserved	Must be kept at reset value
7	BRKG	Break event generation This bit is set by software and cleared by hardware automatically. When this bit is set, the POEN bit is cleared and BRKIF flag is set, related interrupt or DMA transfer can occur if enabled. 0: No generate a break event 1: Generate a break event
6	Reserved	Must be kept at reset value
5	CMTG	Channel commutation event generation This bit is set by software and cleared by hardware automatically. When this bit is set, channel's capture/compare control registers (CHxEN, CHxNEN and CHxCOMCTL bits) are updated based on the value of CCSE (in the TIMERx_CTL1). 0: No affect 1: Generate channel's c/c control update event
4:2	Reserved	Must be kept at reset value
1	CH0G	Channel 0's capture or compare event generation This bit is set by software in order to generate a capture or compare event in channel 0, it is automatically cleared by hardware. When this bit is set, the CH0IF flag is set, the corresponding interrupt or DMA request is sent if enabled. In addition, if channel 1

is configured in input mode, the current value of the counter is captured in `TIMERx_CH0CV` register, and the `CH0OF` flag is set if the `CH0IF` flag was already high.

- 0: No generate a channel 1 capture or compare event
- 1: Generate a channel 1 capture or compare event

**0 UPG** Update event generation

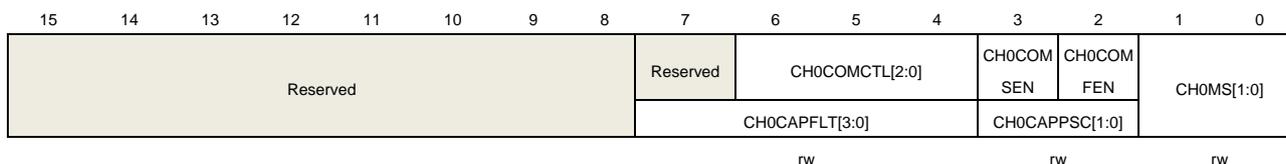
This bit can be set by software, and cleared by hardware automatically. When this bit is set, the counter is cleared if the center-aligned or up counting mode is selected, else (down counting) it takes the auto-reload value. The prescaler counter is cleared at the same time.

- 0: No generate an update event
- 1: Generate an update event

## Channel control register 0 (TIMERx\_CHCTL0)

Address offset: 0x18  
Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



### Output compare mode:

Bits	Fields	Descriptions
15:7	Reserved	Must be kept at reset value
6:4	CH0COMCTL[2:0]	<p>Channel 0 compare output control</p> <p>This bit-field controls the behavior of the output reference signal <code>O0CPRE</code> which drives <code>CH0_O</code> and <code>CH0_ON</code>. <code>O0CPRE</code> is active high, while <code>CH0_O</code> and <code>CH0_ON</code> active level depends on <code>CH0P</code> and <code>CH0NP</code> bits.</p> <p>000: Frozen. The <code>O0CPRE</code> signal keeps stable, independent of the comparison between the register <code>TIMERx_CH0CV</code> and the counter <code>TIMERx_CNT</code>.</p> <p>001: Set the channel output. <code>O0CPRE</code> signal is forced high when the counter matches the output compare register <code>TIMERx_CH0CV</code>.</p> <p>010: Clear the channel output. <code>O0CPRE</code> signal is forced low when the counter matches the output compare register <code>TIMERx_CH0CV</code>.</p> <p>011: Toggle on match. <code>O0CPRE</code> toggles when the counter matches the output compare register <code>TIMERx_CH0CV</code>.</p> <p>100: Force low. <code>O0CPRE</code> is forced low level.</p> <p>101: Force high. <code>O0CPRE</code> is forced high level.</p> <p>110: PWM mode0. When counting up, <code>O0CPRE</code> is active as long as the counter is</p>

smaller than `TIMERx_CH0CV` else inactive. When counting down, `O0CPRE` is inactive as long as the counter is larger than `TIMERx_CH0CV` else active.

111: PWM mode1. When counting up, `O0CPRE` is inactive as long as the counter is smaller than `TIMERx_CH0CV` else active. When counting down, `O0CPRE` is active as long as the counter is larger than `TIMERx_CH0CV` else inactive.

When configured in PWM mode, the `O0CPRE` level changes only when the output compare mode switches from “frozen” mode to “PWM” mode or when the result of the comparison changes.

This bit cannot be modified when `PROT [1:0]` bit-filed in `TIMERx_CCHP` register is 11 and `CH0MS` bit-filed is 00(`COMPARE MODE`).

3	<code>CH0COMSEN</code>	<p>Channel 0 compare output shadow enable</p> <p>When this bit is set, the shadow register of <code>TIMERx_CH0CV</code> register, which updates at each update event, will be enabled.</p> <p>0: Channel 0 output compare shadow disable 1: Channel 0 output compare shadow enable</p> <p>The PWM mode can be used without validating the shadow register only in single pulse mode (<code>SPM</code> bit in <code>TIMERx_CTL0</code> register is set).</p> <p>This bit cannot be modified when <code>PROT [1:0]</code> bit-filed in <code>TIMERx_CCHP</code> register is 11 and <code>CH0MS</code> bit-filed is 00.</p>
2	<code>CH0COMFEN</code>	<p>Channel 0 output compare fast enable</p> <p>When this bit is set, the effect of an event on the trigger in input on the capture/compare output will be accelerated if the channel is configured in <code>PWM0</code> or <code>PWM1</code> mode. The output channel will treat an active edge on the trigger input as a compare match, and <code>CH0_O</code> is set to the compare level independently from the result of the comparison.</p> <p>0: Channel 0 output quickly compare disable. The minimum delay from an edge on the trigger input to activate <code>CH0_O</code> output is 5 clock cycles. 1: Channel 0 output quickly compare enable. The minimum delay from an edge on the trigger input to activate <code>CH0_O</code> output is 3 clock cycles.</p>
1:0	<code>CH0MS[1:0]</code>	<p>Channel 0 I/O mode selection</p> <p>This bit-field specifies the work mode of the channel and the input signal selection. This bit-field is writable only when the channel is not active. (<code>CH0EN</code> bit in <code>TIMERx_CHCTL2</code> register is reset).</p> <p>00: Channel 0 is configured as output 01: Channel 0 is configured as input, <code>IS0</code> is connected to <code>CI0FE0</code> 10: Channel 0 is configured as input, <code>IS0</code> is connected to <code>CI1FE0</code> 11: Channel 0 is configured as input, <code>IS0</code> is connected to <code>ITS</code>, This mode is working only if an internal trigger input is selected through <code>TRGS</code> bits in <code>TIMERx_SMCFG</code> register.</p>

**Input capture mode:**

Bits	Fields	Descriptions
------	--------	--------------

15:8	Reserved	Must be kept at reset value
7:4	CH0CAPFLT[3:0]	<p>Channel 0 input capture filter control</p> <p>An event counter is used in the digital filter, in which a transition on the output occurs after N input events. This bit-field specifies the frequency used to sample CIO input signal and the length of the digital filter applied to CIO.</p> <p>0000: Filter disabled, <math>f_{SAMP}=f_{DTS}</math>, N=1</p> <p>0001: <math>f_{SAMP}=f_{TIMER\_CK}</math>, N=2</p> <p>0010: <math>f_{SAMP}=f_{TIMER\_CK}</math>, N=4</p> <p>0011: <math>f_{SAMP}=f_{TIMER\_CK}</math>, N=8</p> <p>0100: <math>f_{SAMP}=f_{DTS}/2</math>, N=6</p> <p>0101: <math>f_{SAMP}=f_{DTS}/2</math>, N=8</p> <p>0110: <math>f_{SAMP}=f_{DTS}/4</math>, N=6</p> <p>0111: <math>f_{SAMP}=f_{DTS}/4</math>, N=8</p> <p>1000: <math>f_{SAMP}=f_{DTS}/8</math>, N=6</p> <p>1001: <math>f_{SAMP}=f_{DTS}/8</math>, N=8</p> <p>1010: <math>f_{SAMP}=f_{DTS}/16</math>, N=5</p> <p>1011: <math>f_{SAMP}=f_{DTS}/16</math>, N=6</p> <p>1100: <math>f_{SAMP}=f_{DTS}/16</math>, N=8</p> <p>1101: <math>f_{SAMP}=f_{DTS}/32</math>, N=5</p> <p>1110: <math>f_{SAMP}=f_{DTS}/32</math>, N=6</p> <p>1111: <math>f_{SAMP}=f_{DTS}/32</math>, N=8</p>
3:2	CH0APPSC[1:0]	<p>Channel 0 input capture prescaler</p> <p>This bit-field specifies the factor of the prescaler on channel 0 input. The prescaler is reset when CH0EN bit in TIMERx_CHCTL2 register is clear.</p> <p>00: Prescaler disable, capture is done on each channel input edge</p> <p>01: Capture is done every 2 channel input edges</p> <p>10: Capture is done every 4 channel input edges</p> <p>11: Capture is done every 8 channel input edges</p>
1:0	CH0MS[1:0]	<p>Channel 0 mode selection</p> <p>Same as Output compare mode</p>

### Channel control register 2 (TIMERx\_CHCTL2)

Address offset: 0x20

Reset value: 0x0000

This register can be accessed by half-word(16-bit) or word(32-bit)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												CH0NP	CH0NEN	CH0P	CH0EN
												rw	rw	rw	rw

Bits	Fields	Descriptions
------	--------	--------------

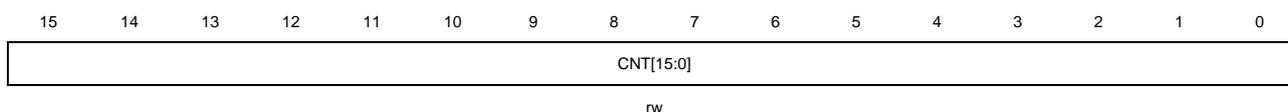
15:4	Reserved	Must be kept at reset value
3	CH0NP	<p>Channel 0 complementary output polarity</p> <p>When channel 0 is configured in output mode, this bit specifies the complementary output signal polarity.</p> <p>0: Channel 0 active high</p> <p>1: Channel 0 active low</p> <p>When channel 0 is configured in input mode, In conjunction with CH0P, this bit is used to define the polarity of C10.</p> <p>This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 or 10.</p>
2	CH0NEN	<p>Channel 0 complementary output enable</p> <p>When channel 0 is configured in output mode, setting this bit enables the complementary output in channel0.</p> <p>0: Channel 0 complementary output disabled</p> <p>1: Channel 0 complementary output enabled</p>
1	CH0P	<p>Channel 0 capture/compare function polarity</p> <p>When channel 0 is configured in output mode, this bit specifies the output signal polarity.</p> <p>0: Channel 0 active high</p> <p>1: Channel 0 active low</p> <p>When channel 0 is configured in input mode, this bit specifies the C10 signal polarity. [CH0NP, CH0P] will select the active trigger or capture polarity for C10FE0 or C11FE0. [CH0NP==0, CH0P==0]: C1xFE0's rising edge is the active signal for capture or trigger operation in slave mode. And C1xFE0 will not be inverted. [CH0NP==0, CH0P==1]: C1xFE0's falling edge is the active signal for capture or trigger operation in slave mode. And C1xFE0 will be inverted. [CH0NP==1, CH0P==0]: Reserved. [CH0NP==1, CH0P==1]: C1xFE0's falling and rising edge are both the active signal for capture or trigger operation in slave mode. And C1xFE0 will be not inverted.</p> <p>This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 or 10.</p>
0	CH0EN	<p>Channel 0 capture/compare function enable</p> <p>When channel 0 is configured in output mode, setting this bit enables CH0_O signal in active state. When channel 0 is configured in input mode, setting this bit enables the capture event in channel0.</p> <p>0: Channel 0 disabled</p> <p>1: Channel 0 enabled</p>

### Counter register (TIMERx\_CNT)

Address offset: 0x24

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



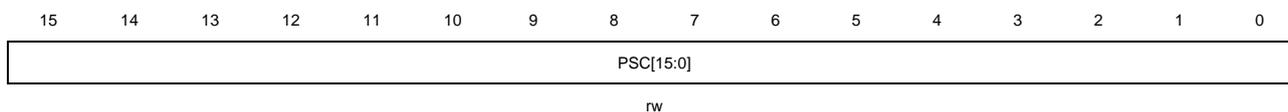
Bits	Fields	Descriptions
15:0	CNT[15:0]	This bit-filed indicates the current counter value. Writing to this bit-filed can change the value of the counter.

### Prescaler register (TIMERx\_PSC)

Address offset: 0x28

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



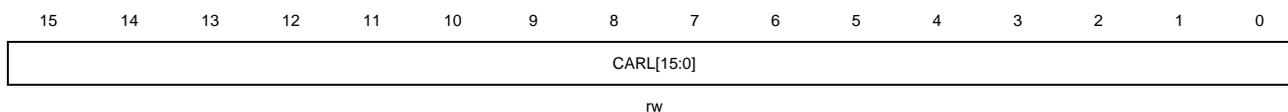
Bits	Fields	Descriptions
15:0	PSC[15:0]	Prescaler value of the counter clock The PSC clock is divided by (PSC+1) to generate the counter clock. The value of this bit-filed will be loaded to the corresponding shadow register at every update event.

### Counter auto reload register (TIMERx\_CAR)

Address offset: 0x2C

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



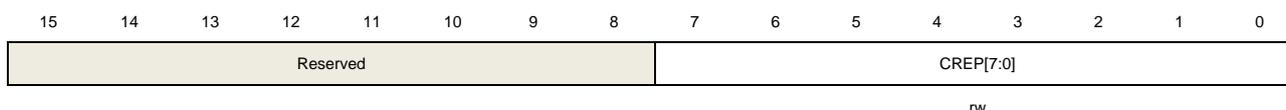
Bits	Fields	Descriptions
15:0	CARL[15:0]	Counter auto reload value This bit-filed specifies the auto reload value of the counter.

### Counter repetition register (TIMERx\_CREP)

Address offset: 0x30

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



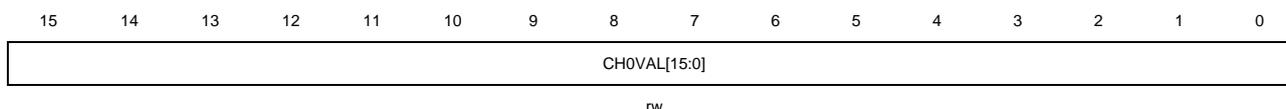
Bits	Fields	Descriptions
15:8	Reserved	Must be kept at reset value.
7:0	CREP[7:0]	Counter repetition value This bit-filed specifies the update event generation rate. Each time the repetition counter counting down to zero, an update event is generated. The update rate of the shadow registers is also affected by this bit-filed when these shadow registers are enabled.

### Channel 0 capture/compare value register (TIMERx\_CH0CV)

Address offset: 0x34

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



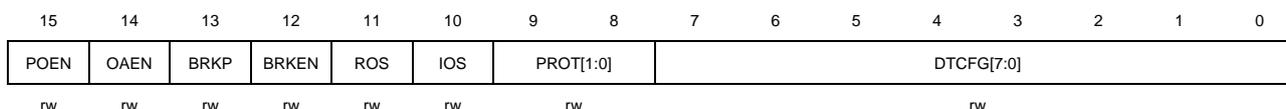
Bits	Fields	Descriptions
15:0	CH0VAL[15:0]	Capture or compare value of channel0 When channel 0 is configured in input mode, this bit-filed indicates the counter value corresponding to the last capture event. And this bit-filed is read-only. When channel 0 is configured in output mode, this bit-filed contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates every update event.

### Channel complementary protection register (TIMERx\_CCHP)

Address offset: 0x44

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



Bits	Fields	Descriptions
------	--------	--------------

15	POEN	<p>Primary output enable</p> <p>This bit is set by software or automatically by hardware depending on the OAEN bit. It is cleared asynchronously by hardware as soon as the break input is active. When one of channels is configured in output mode, setting this bit enables the channel outputs (CHx_O and CHx_ON) if the corresponding enable bits (CHxEN, CHxNEN in TIMERx_CHCTL2 register) have been set.</p> <p>0: Channel outputs are disabled or forced to idle state. 1: Channel outputs are enabled.</p>
14	OAEN	<p>Output automatic enable</p> <p>This bit specifies whether the POEN bit can be set automatically by hardware.</p> <p>0: POEN can be not set by hardware. 1: POEN can be set by hardware automatically at the next update event, if the break input is not active.</p> <p>This bit can be modified only when PROT [1:0] bit-field in TIMERx_CCHP register is 00.</p>
13	BRKP	<p>Break polarity</p> <p>This bit specifies the polarity of the BRKIN input signal.</p> <p>0: BRKIN input active low 1: BRKIN input active high</p>
12	BRKEN	<p>Break enable</p> <p>This bit can be set to enable the BRKIN and CCS clock failure event inputs.</p> <p>0: Break inputs disabled 1: Break inputs enabled</p> <p>This bit can be modified only when PROT [1:0] bit-field in TIMERx_CCHP register is 00.</p>
11	ROS	<p>Run mode off-state configure</p> <p>When POEN bit is set, this bit specifies the output state for the channels which has a complementary output and has been configured in output mode.</p> <p>0: When POEN bit is set, the channel output signals (CHx_O/CHx_ON) are disabled. 1: When POEN bit is set, the channel output signals (CHx_O/CHx_ON) are enabled, with relationship to CHxEN/CHxNEN bits in TIMERx_CHCTL2 register.</p> <p>This bit cannot be modified when PROT [1:0] bit-field in TIMERx_CCHP register is 10 or 11.</p>
10	IOS	<p>Idle mode off-state configure</p> <p>When POEN bit is reset, this bit specifies the output state for the channels which has been configured in output mode.</p> <p>0: When POEN bit is reset, the channel output signals (CHx_O/CHx_ON) are disabled. 1: When POEN bit is reset, the channel output signals (CHx_O/CHx_ON) are enabled, with relationship to CHxEN/CHxNEN bits in TIMERx_CHCTL2 register.</p> <p>This bit cannot be modified when PROT [1:0] bit-field in TIMERx_CCHP register is 10</p>

or 11.

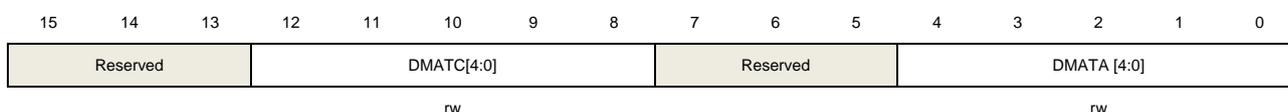
- 9:8      PROT[1:0]      Complementary register protect control  
 This bit-field specifies the write protection property of registers.  
 00: protect disable. No write protection.  
 01: PROT mode 0. The ISOx/ISOxN bits in TIMERx\_CTL1 register and the BRKEN/BRKP/OAEN/DTCFG bits in TIMERx\_CCHP register are writing protected.  
 10: PROT mode 1. In addition of the registers in PROT mode 0, the CHxP/CHxNP bits in TIMERx\_CHCTL2 register (if related channel is configured in output mode) and the ROS/IOS bits in TIMERx\_CCHP register are writing protected.  
 11: PROT mode 2. In addition of the registers in PROT mode 1, the CHxCOMCTL/CHxCOMSEN bits in TIMERx\_CHCTL0/1 registers (if the related channel is configured in output) are writing protected.  
 This bit-field can be written only once after the reset. Once the TIMERx\_CCHP register has been written, this bit-field will be writing protected.
- 7:0      DTCFG[7:0]      Dead time configure  
 This bit-field controls the value of the dead-time, which is inserted before the output transitions. The relationship between DTCFG value and the duration of dead-time is as follow:  
 $DTCFG [7:5] = 3'b0xx: DTvalue = DTCFG [7:0] \times t_{DT}, t_{DT} = t_{DTS}$   
 $DTCFG [7:5] = 3'b 10x: DTvalue = (64 + DTCFG [5:0]) \times t_{DT}, t_{DT} = t_{DTS} * 2$   
 $DTCFG [7:5] = 3'b 110: DTvalue = (32 + DTCFG [4:0]) \times t_{DT}, t_{DT} = t_{DTS} * 8$   
 $DTCFG [7:5] = 3'b 111: DTvalue = (32 + DTCFG [4:0]) \times t_{DT}, t_{DT} = t_{DTS} * 16$   
 This bit can be modified only when PROT [1:0] bit-field in TIMERx\_CCHP register is 00.

## DMA configuration register (TIMERx\_DMACFG)

Address offset: 0x48

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



Bits	Fields	Descriptions
15:14	Reserved	Must be kept at reset value.
12:8	DMATC [4:0]	DMA transfer count This field is defined the number of DMA will access(R/W) the register of TIMERx_DMATB
7:5	Reserved	Must be kept at reset value.

4:0      DMATA [4:0]      DMA transfer access start address

This field defines the first address for the DMA access to the TIMERx\_DMATB. When access is done through the TIMERx\_DMA address first time, this bit-field specifies the address you just access. And then the second access to the TIMERx\_DMATB, you will access the address of start address + 0x4.

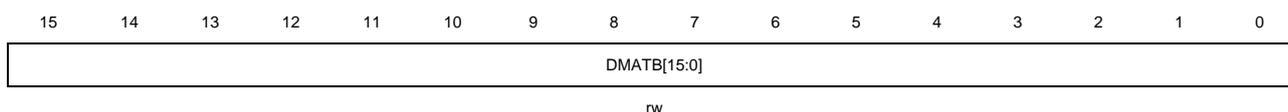
5'b0\_0000: TIMERx\_CTL0  
 5'b0\_0001: TIMERx\_CTL1  
 ...

In a word: Start Address = TIMERx\_CTL0 + DMATA\*4

## DMA transfer buffer register (TIMERx\_DMATB)

Address offset: 0x4C  
 Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)

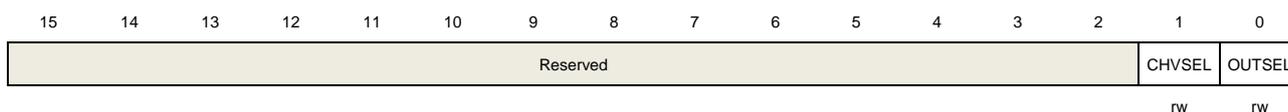


Bits	Fields	Descriptions
15:0	DMATB[15:0]	DMA transfer buffer When a read or write operation is assigned to this register, the register located at the address range (Start Addr + Transfer Timer* 4) will be accessed. The transfer Timer is calculated by hardware, and ranges from 0 to DMATC.

## Configuration register (TIMERx\_CFG) of GD32F170xx and GD32F190xx devices

Address offset: 0xFC  
 Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



Bits	Fields	Descriptions
15:2	Reserved	Must be kept at reset value
1	CHVSEL	Write CHxVAL register selection This bit-field set and reset by software.

		1: If write the CHxVAL register, the write value is same as the CHxVAL value, the write access ignored 0: No effect
0	OUTSEL	The output value selection This bit-field set and reset by software 1: If POEN and IOS is 0, the output disabled 0: No effect

## 15.6. Basic timer (TIMERx, x=5)

### 15.6.1. Overview

The basic timer module (TIMER5) reference is a 16-bit counter that can be used as an unsigned counter. The basic timer can be configured to generate DMA request and TRGO to DAC.

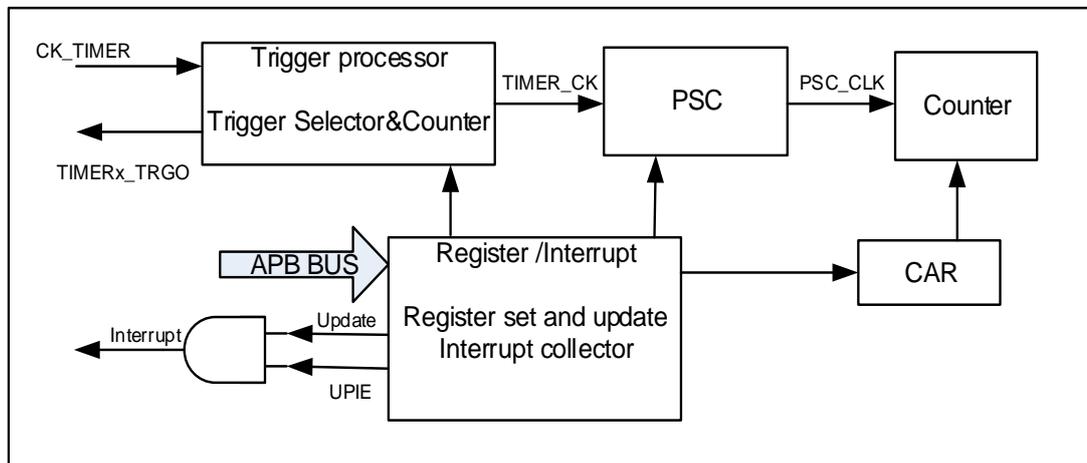
### 15.6.2. Characteristics

- Counter width: 16bit.
- Source of count clock is internal clock only.
- Counter modes: only count up.
- Programmable prescaler: 16 bit. Factor can be changed on the go.
- Auto-reload function.
- Interrupt output or DMA request on update event.

### 15.6.3. Block diagram

[Figure 15-85. Basic timer block diagram](#) provides details on the internal configuration of the basic timer.

**Figure 15-85. Basic timer block diagram**



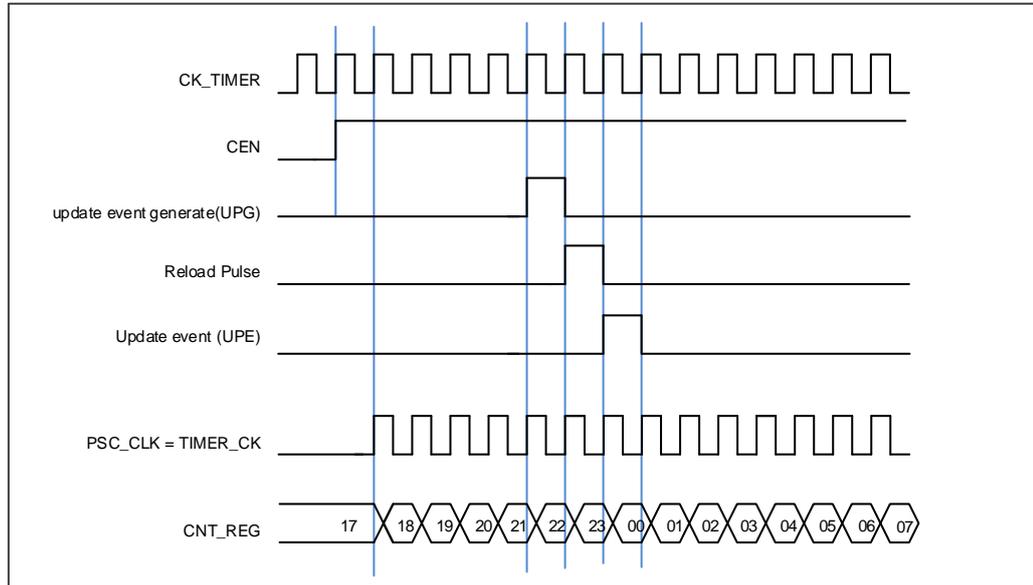
### 15.6.4. Function overview

#### Clock selection

The basic TIMER can only being clocked by the internal timer clock CK\_TIMER, which is from the source named CK\_TIMER in RCU

The TIMER\_CK, driven counter's prescaler to count, is equal to CK\_TIMER used to drive the counter prescaler. When the CEN is set, the CK\_TIMER will be divided by PSC value to generate PSC\_CLK.

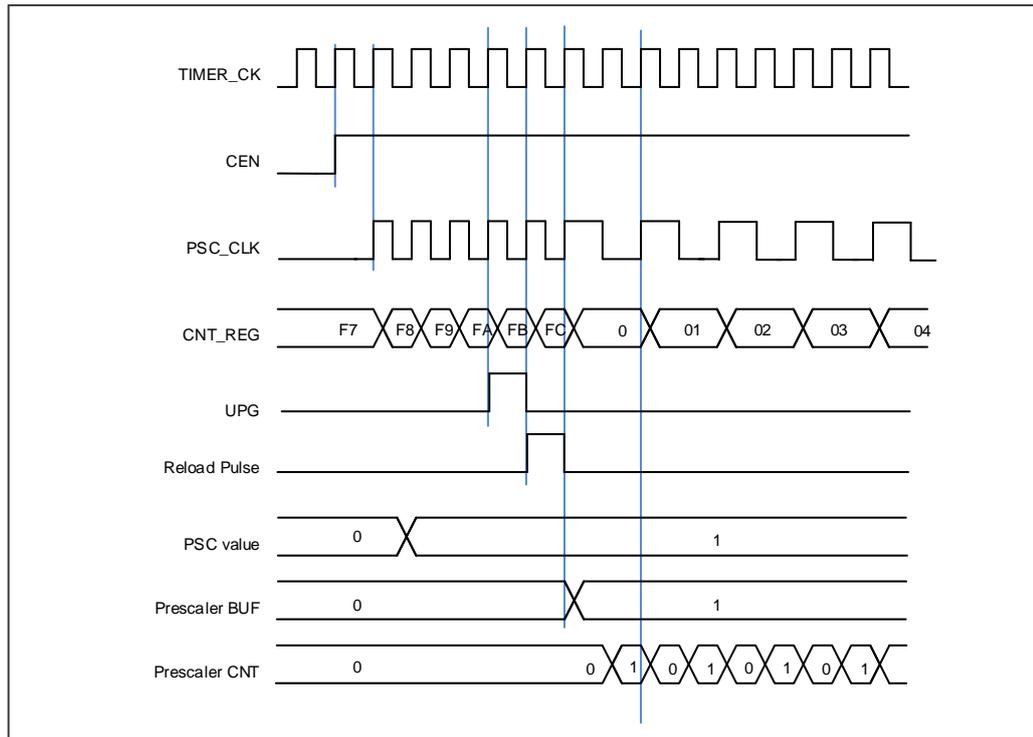
**Figure 15-86. Normal mode, internal clock divided by 1**



## Prescaler

The prescaler can divide the timer clock (TIMER\_CK) to the counter clock (PSC\_CLK) by any factor between 1 and 65536. It is controlled through prescaler register (TIMERx\_PSC) which can be changed on the go but be taken into account at the next update event.

**Figure 15-87. Counter timing diagram with prescaler division change from 1 to 2**



### Up counting mode

In this mode, the counter counts up continuously from 0 to the counter-reload value, which is defined in the `TIMERx_CAR` register, in a count-up direction. Once the counter reaches the counter reload value, the counter restarts to count once again from 0. The update event is generated at each counter overflow. The counting direction bit `DIR` in the `TIMERx_CTL1` register should be set to 0 for the up counting mode.

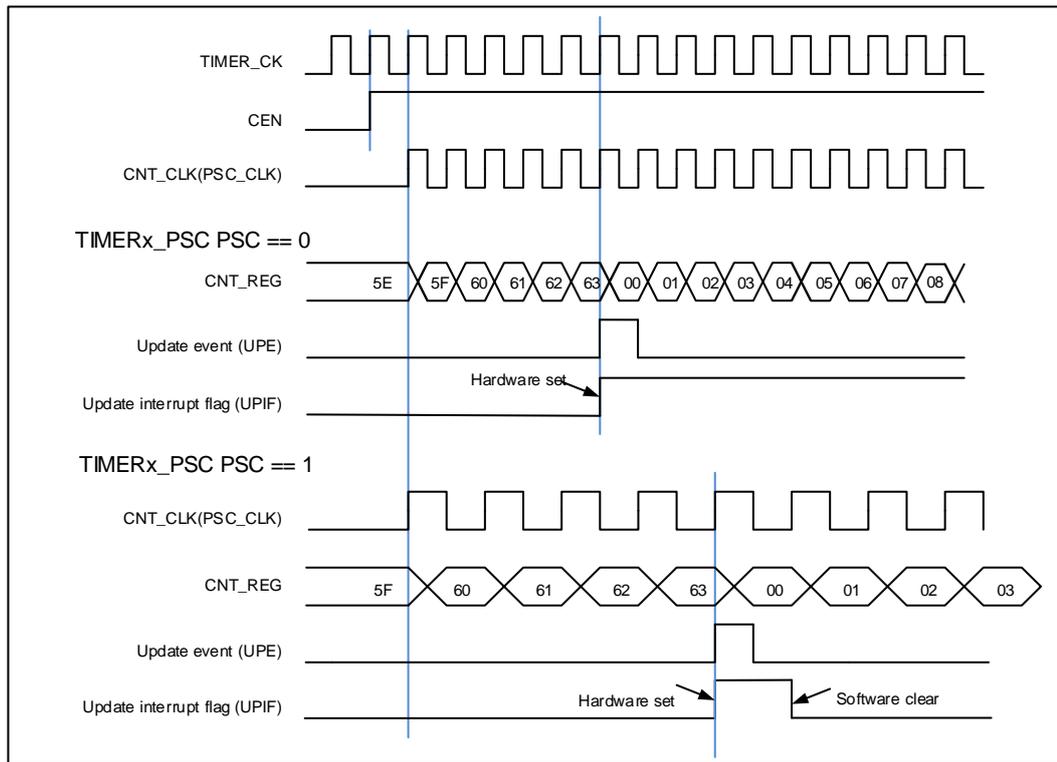
When the update event is set by the `UPG` bit in the `TIMERx_SWEVG` register, the counter value will be initialized to 0 and generates an update event.

If set the `UPDIS` bit in `TIMERx_CTL0` register, the update event is disabled.

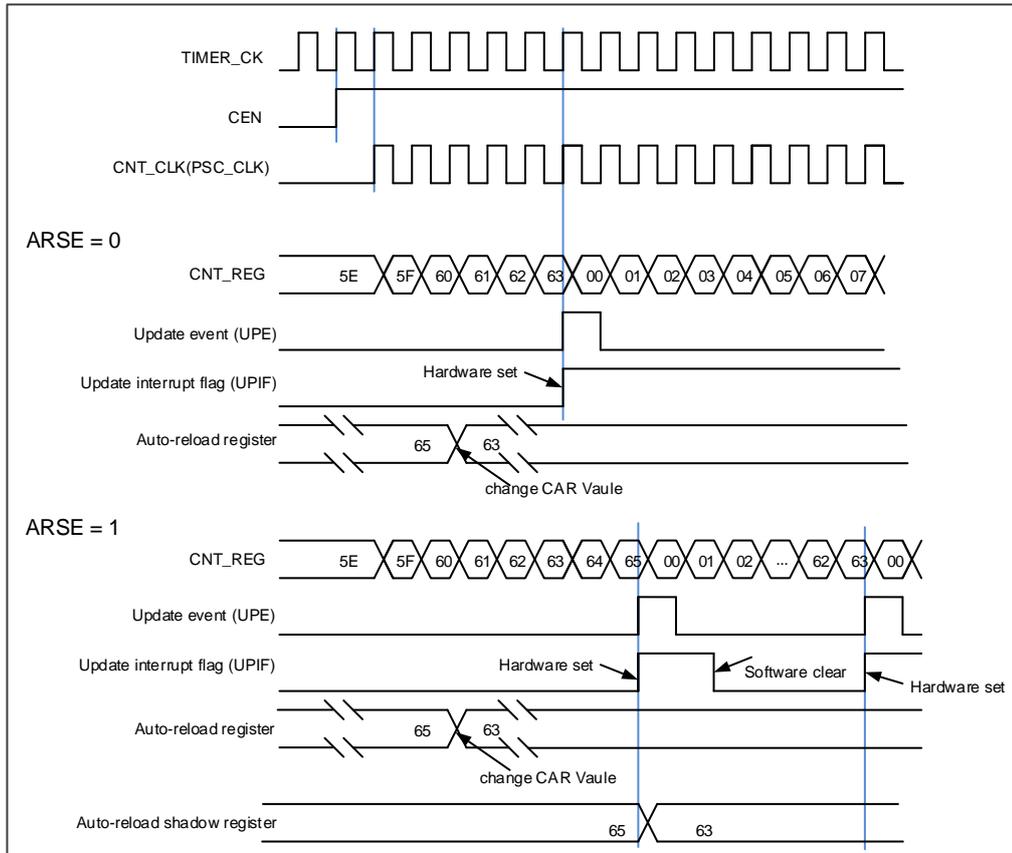
When an update event occurs, all the registers (repetition counter, auto reload register, prescaler register) are updated.

The following figures show some examples of the counter behavior for different clock prescaler factor when `TIMERx_CAR=0x63`.

Figure 15-88. Up-counter timechart, PSC=0/1



**Figure 15-89. Up-counter timechart, change TIMERx\_CAR on the go**



### Timer debug mode

When the Cortex™-M3 halted, and the TIMERx\_HOLD configuration bit in DBG\_CTL0 register set to 1, the TIMERx counter stops.

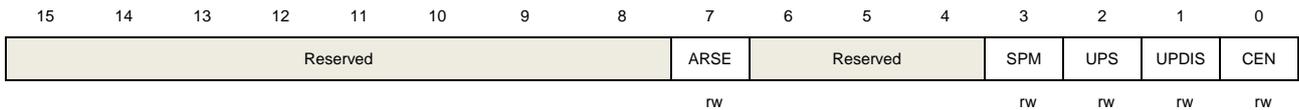
## 15.6.5. TIMERx registers(x=5)

### Control register 0 (TIMERx\_CTL0)

Address offset: 0x00

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



Bits	Fields	Descriptions
15:8	Reserved	Must be kept at reset value
7	ARSE	Auto-reload shadow enable 0: The shadow register for TIMERx_CAR register is disabled 1: The shadow register for TIMERx_CAR register is enabled
6:4	Reserved	Must be kept at reset value
3	SPM	Single pulse mode. 0: Counter continues after update event. 1: The CEN is cleared by hardware and the counter stops at next update event.
2	UPS	Update source This bit is used to select the update event sources by software. 0: When enabled, any of the following events generate an update interrupt or DMA request: <ul style="list-style-type: none"> <li>- The UPG bit is set</li> <li>- The counter generates an overflow or underflow event</li> <li>- The slave mode controller generates an update event.</li> </ul> 1: When enabled, only counter overflow/underflow generates an update interrupt or DMA request.
1	UPDIS	Update disable. This bit is used to enable or disable the update event generation. 0: update event enable. The update event is generate and the buffered registers are loaded with their preloaded values when one of the following events occurs: <ul style="list-style-type: none"> <li>- The UPG bit is set</li> <li>- The counter generates an overflow or underflow event</li> <li>- The slave mode controller generates an update event.</li> </ul> 1: update event disable. The buffered registers keep their value, while the counter and the prescaler are reinitialized if the UG bit is set or if the slave mode controller generates a hardware reset event.

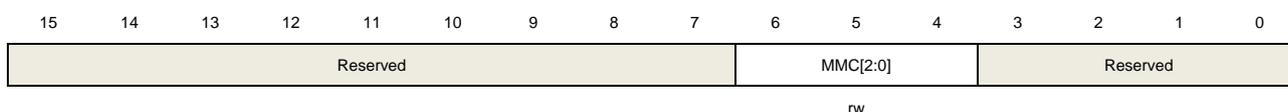
0	CEN	Counter enable 0: Counter disable 1: Counter enable The CEN bit must be set by software when timer works in external clock, pause mode and encoder mode. While in event mode, the hardware can set the CEN bit automatically.
---	-----	--

### Control register 1 (TIMERx\_CTL1)

Address offset: 0x04

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



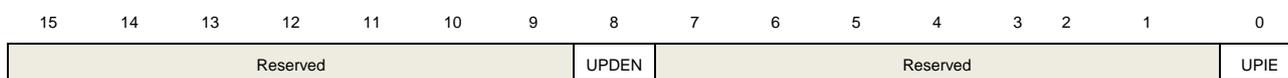
Bits	Fields	Descriptions
15:7	Reserved	Must be kept at reset value
6:4	MMC[2:0]	Master mode control These bits control the selection of TRGO signal, which is sent in master mode to slave timers for synchronization function. 000: Reset. When the UPG bit in the TIMERx_SWEVG register is set or a reset is generated by the slave mode controller, a TRGO pulse occurs. And in the latter case, the signal on TRGO is delayed compared to the actual reset. 001: Enable. This mode is useful to start several timers at the same time or to control a window in which a slave timer is enabled. In this mode the master mode controller selects the counter enable signal TIMERx_EN as TRGO. The counter enable signal is set when CEN control bit is set or the trigger input in pause mode is high. There is a delay between the trigger input in pause mode and the TRGO output, except if the master-slave mode is selected. 010: Update. In this mode the master mode controller selects the update event as TRGO.
3:0	Reserved	Must be kept at reset value.

### Interrupt enable register (TIMERx\_DMAINTEN)

Address offset: 0x0C

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



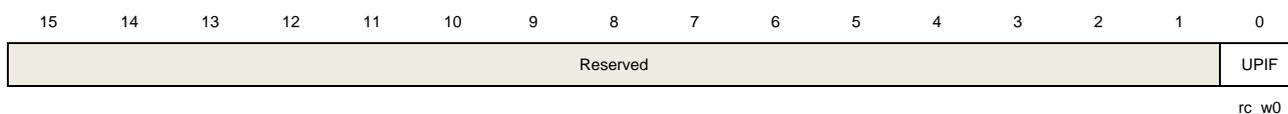
Bits	Fields	Descriptions
15:9	Reserved	Must be kept at reset value.
8	UPDEN	Update DMA request enable 0: disabled 1: enabled
7:1	Reserved	Must be kept at reset value.
0	UPIE	Update interrupt enable 0: disabled 1: enabled

### Interrupt flag register (TIMERx\_INTF)

Address offset: 0x10

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



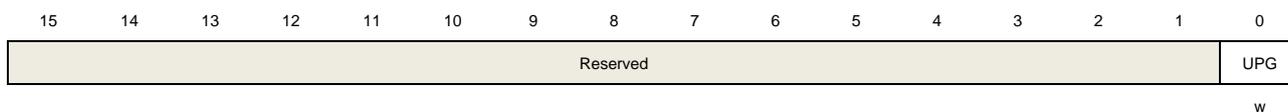
Bits	Fields	Descriptions
15:1	Reserved	Must be kept at reset value.
0	UPIF	Update interrupt flag This bit is set by hardware on an update event and cleared by software. 0: No update interrupt occurred 1: Update interrupt occurred

### Software event generation register (TIMERx\_SWEVG)

Address offset: 0x14

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



Bits	Fields	Descriptions
15:1	Reserved	Must be kept at reset value.

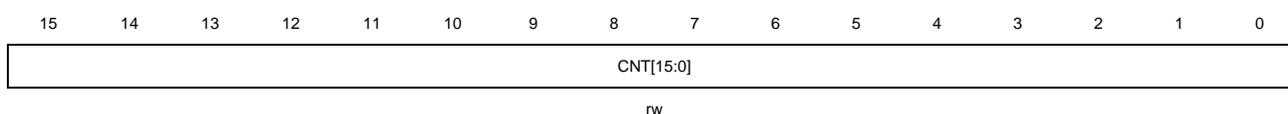
0	UPG	<p>This bit can be set by software, and cleared by hardware automatically. When this bit is set, the counter is cleared. The prescaler counter is cleared at the same time.</p> <p>0: No generate an update event 1: Generate an update event</p>
---	-----	---

### Counter register (TIMERx\_CNT)

Address offset: 0x24

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



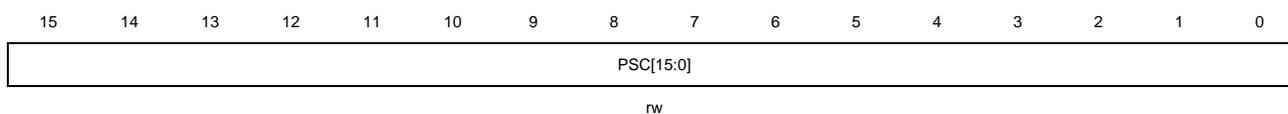
Bits	Fields	Descriptions
15:0	CNT[15:0]	This bit-filed indicates the current counter value. Writing to this bit-filed can change the value of the counter.

### Prescaler register (TIMERx\_PSC)

Address offset: 0x28

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



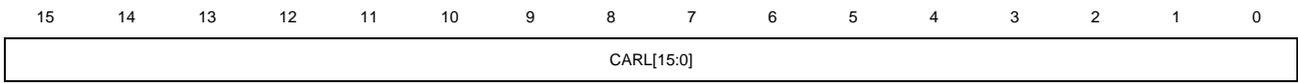
Bits	Fields	Descriptions
15:0	PSC[15:0]	<p>Prescaler value of the counter clock</p> <p>The PSC clock is divided by (PSC+1) to generate the counter clock. The value of this bit-filed will be loaded to the corresponding shadow register at every update event.</p>

### Counter auto reload register (TIMERx\_CAR)

Address offset: 0x2C

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



rw

Bits	Fields	Descriptions
15:0	CARL[15:0]	Counter auto reload value This bit-field specifies the auto reload value of the counter.

## 16. Infrared ray port (IFRP)

### 16.1. Overview

Infrared ray port (IFRP) is used to control infrared light LED, and send out infrared data to implement infrared ray remote control.

There is no register in this module, which is controlled by TIMER15 and TIMER16. You can improve the module's output to high current capacity by set the GPIO pin to Fast Mode.

### 16.2. Characteristics

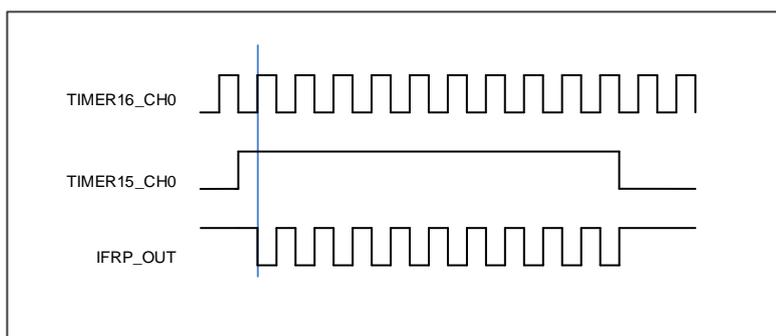
- The IFRP output signal is decided by TIMER15\_CH0 and TIMER16\_CH0
- To get correct infrared ray signal, TIMER15 should generate low frequency modulation envelope signal, and TIMER16 should generate high frequency carrier signal
- The IFRP output (PB9) can provide high current to control LED interface by setting PB9\_HCCE in SYSCFG\_CFG0

### 16.3. Function overview

IFRP is a module which is able to integrate the output of TIMER15 and TIMER16 to generate an infrared ray signal.

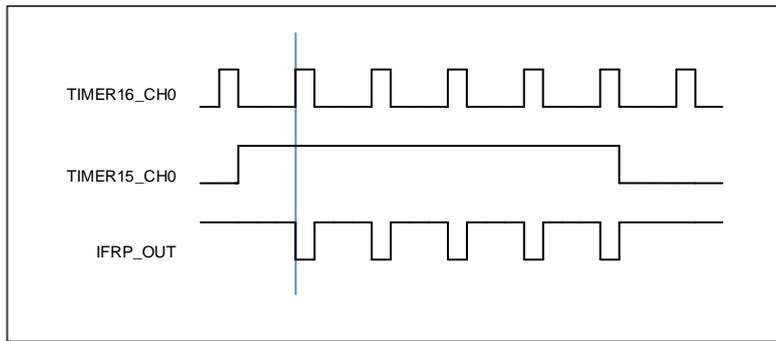
1. The TIMER15's CH0 is programed to generate the low frequency PWM signal which is the modulation evalope signal. The TIMER16's CH0 is programed to generate the high frquence PWM signal which is the carrier signal. And the channel need to be enabled before generating these signals.
2. Program the GPIO remap regisger and enable the pin.
3. If you want to get the high current capacity of output, remapping IFRP\_OUT to PB9 and setting the PB9 to Fast Mode by the register in SYS\_CFG module are required.

**Figure 16-1. IFRP output timechart 1**



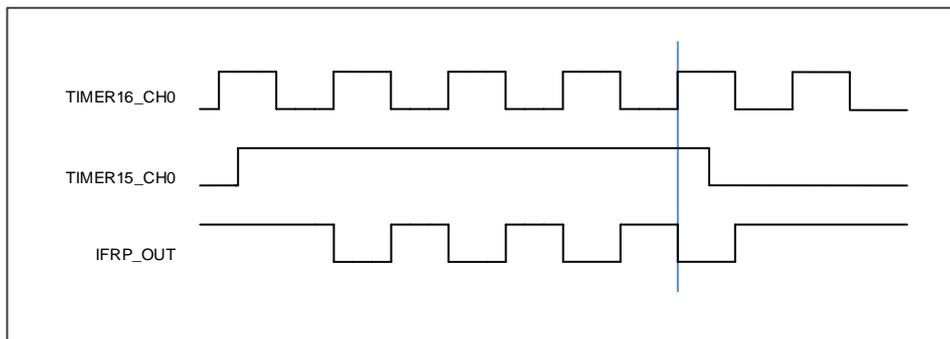
**Note:** IFRP\_OUT has one APB clock delay from TIMER16\_CH0.

**Figure 16-2. IFRP output timechart 2**



**Note:** Carrier (TIMER15\_CH0)'s duty cycle can be changed, and IFRP\_OUT has inverted relationship with TIMER16\_CH0 when TIMER15\_CH0 is high.

**Figure 16-3. IFRP output timechart 3**



**Note:** IFRP\_OUT will keep the integrity of TIMER16\_CH0, even if envelope signal (TIMER15\_CH0) is no active.

## 17. Universal synchronous asynchronous receiver transmitter (USART)

### 17.1. Overview

The Universal Synchronous Asynchronous Receiver Transmitter (USART) provides a flexible serial data exchange interface. Data frames can be transferred in full duplex or half duplex mode, synchronously or asynchronously through this interface. A programmable baud rate generator produces a dedicated wide range baudrate clock for the USART transmitter and receiver.

Besides the standard asynchronous receiver and transmitter mode, the USART implements several other types of serial data exchange modes, such as IrDA (infrared data association) SIR mode, smartcard mode, LIN (local interconnection network) mode and half-duplex synchronous mode. It also supports multiprocessor communication mode, and hardware flow control protocol (CTS/RTS). The data frame can be transferred from LSB or MSB bit. The polarity of the data bits and the TX/RX pins can be configured flexibly.

ALL USARTs support DMA function for high-speed data communication.

### 17.2. Characteristics

- NRZ standard format (Mark/Space)
- Asynchronous, full duplex communication
- Half duplex single wire communications
- Dual clock domain
  - Asynchronous pclk and USART clock
  - Baud rate programming independent from the PCLK reprogramming
- Programmable baud-rate generator allowing speed up to 9 Mbits/s when the clock frequency is 72 MHz and oversampling is by 8.
- Fully programmable serial interface characteristics:
  - A data word (8 or 9 bits) LSB or MSB first
  - Even, odd or no-parity bit generation/detection
  - 1, 1.5 or 2 stop bit generation
- Swappable Tx/Rx pin

- Configurable data polarity
- Auto baud rate detection
- Hardware Modem operations (CTS/RTS) and RS485 drive enable
- Configurable multibuffer communication using centralized DMA
- Separate enable bits for Transmitter and Receiver
- Parity control
  - Transmits parity bit
  - Checks parity of received data byte
- LIN Break generation and detection
- IrDA Support
- Synchronous mode and transmitter clock output for synchronous transmission
- ISO 7816-3 compliant smartcard interface
  - Character mode (T=0)
  - Block mode (T=1)
  - Direct and inverse convention
- Multiprocessor communication
  - Enter into mute mode if address match does not occur
  - Wake up from mute mode by idle line or address mark detection
- Support for ModBus communication
  - Timeout feature
  - CR/LF character recognition
- Wake up from Deep-sleep mode
  - By standard RXNE interrupt
  - By WUF interrupt
- Various status flags
  - Flags for transfer detection: Receive buffer not empty (RBNE), Transmit buffer empty (TBE), transfer complete (TC).
  - Flags for error detection: overrun error (ORERR), noise error (NERR), frame error (FERR) and parity error (PERR)
  - Flag for hardware flow control: CTS changes (CTSF)

- Flag for LIN mode: LIN break detected (LBDF)
- Flag for multiprocessor communication: IDLE frame detected (IDLEF)
- Flag for ModBus communication: Address/character match (AMF) and receiver timeout (RTF)
- Flags for smartcard block mode: end of block (EBF) and receiver timeout (RTF)
- Wakeup from Deep-sleep mode flag
- Interrupt occurs at these events when the corresponding interrupt enable bits are set

While USART0 is fully implemented, USART1 is only partially implemented with the following features not supported.

- Auto baud rate detection
- Smartcard mode
- IrDA SIR ENDEC block
- LIN mode
- Dual clock domain and wakeup from Deep-sleep mode
- Receiver timeout interrupt
- Modbus communication

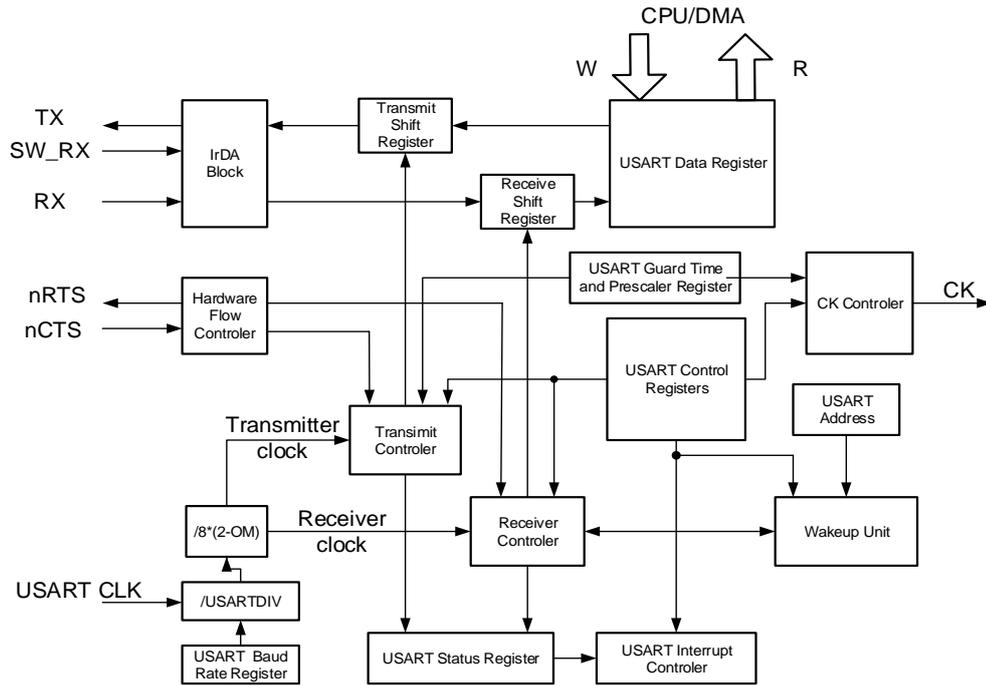
## 17.3. Function overview

The interface is externally connected to another device by the main pins listed as following.

**Table 17-1. USART important pins description**

Pin	Type	Description
RX	Input	Receive Data
TX	Output I/O (single-wire/smartcard mode)	Transmit Data. high level When enabled but nothing to be transmitted
CK	Output	Serial clock for synchronous communication
nCTS	Input	Clear to send in Hardware flow control mode
nRTS	Output	Request to send in Hardware flow control mode

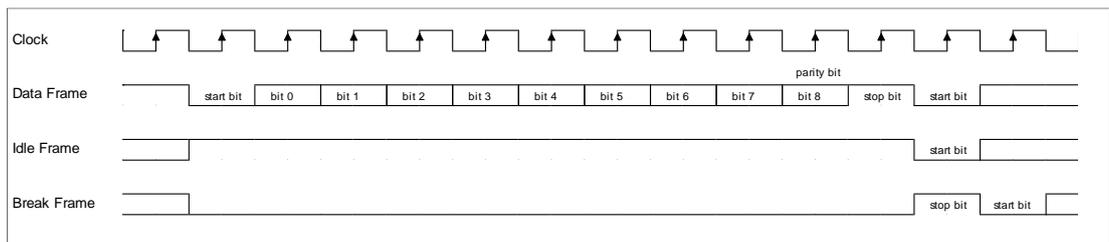
Figure 17-1. USART module block diagram



### 17.3.1. USART frame format

The USART frame starts with a start bit and ends up with a number of stop bits. The length of the data frame is configured by the WL bit in the USART\_CTL0 register. The last data bit can be used as parity check bit by setting the PCEN bit in USART\_CTL0 register. When the WL bit is reset, the parity bit is the 8th bit. When the WL bit is set, the parity bit is the 9th bit. The method of calculating the parity bit is selected by the PM bit in USART\_CTL0 register.

Figure 17-2. USART character frame (9 bits data and 1 stop bit)



In transmission and reception, the number of stop bits can be configured by the STB[1:0] bits in the USART\_CTL1 register.

Table 17-2. Stop bits configuration

STB[1:0]	stop bit length (bit)	usage description
00	1	default value
01	Reserved	Reserved
10	2	normal USART and single-wire modes
11	1.5	Smartcard mode for transmitting and receiving

In an idle frame, all the frame bits are logic 1. The frame length is equal to the normal USART frame.

A break frame is configured number of low bits followed by the configured number of stop bits. The transfer speed of a USART frame depends on the frequency of the PCLK, the configuration of the baud rate generator and the oversampling mode.

### 17.3.2. Baud rate generation

The baud-rate divisor is a 16-bit number consisting of a 12-bit integer and a 4-bit fractional part. The number formed by these two values is used by the baud-rate generator to determine the bit period. Having a fractional baud-rate divider allows the USART to generate all the standard baud rates.

The baud-rate divider (USARTDIV) has the following relationship to the system clock:

In case of oversampling by 16, the equation is:

$$\text{USARTDIV} = \frac{\text{UCLK}}{16 \times \text{Baud Rate}}$$

In case of oversampling by 8, the equation is:

$$\text{USARTDIV} = \frac{\text{UCLK}}{8 \times \text{Baud Rate}}$$

The choice of the USART clock (UCLK) is done through the Clock Control system (see the Reset and clock unit (RCU) section). The clock source must be chosen before enabling the USART (by setting the UEN bit).

### 17.3.3. USART transmitter

If the transmit enable bit (TEN) in USART\_CTL0 register is set, when the transmit data buffer is not empty, the transmitter shifts out the transmit data frame through the TX pin. The polarity of the TX pin can be configured by the TINV bit in the USART\_CTL3 register. Clock pulses can be output through the CK pin.

In case of transmission corruption, the TEN bit should not be disabled when transmission is ongoing.

After power on, the TBE bit is high by default. Data can be written to the USART\_TDATA when the TBE bit of the USART\_STAT register is asserted. The TBE bit is cleared by a writing to the USART\_TDATA register and will be set by hardware after the data is put into the transmit shift register. If a data is written to the USART\_TDATA register while a transmission is ongoing, it will be firstly stored in the transmit buffer, and transferred to the transmit shift register after the current transmission is done. If a data is written to the USART\_TDATA register while no transmission is ongoing, the TBE bit will be cleared and set soon, because the data will be transferred to the transmit shift register immediately.

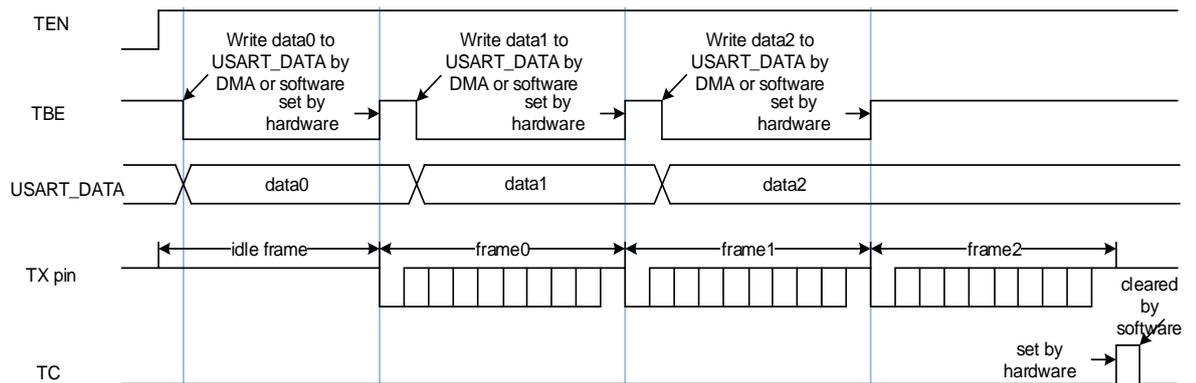
If a frame is transmitted and the TBE bit is asserted, the TC bit of the USART\_STAT register

will be set. An interrupt is generated if the corresponding interrupt enable bit (TCIE) is set in the USART\_CTL0 register.

Refer to the following procedure for the USART transmission:

1. Write the WL bit in USART\_CTL0 to set the data bits length.
2. Set the stop bits length in USART\_CTL1.
3. Enable DMA (DENT bit) in USART\_CTL2 if multibuffer communication is selected.
4. Set the baud rate in USART\_BAUD.
5. Set the UEN bit in USART\_CTL0 to enable the USART.
6. Set the TEN bit in USART\_CTL0.
7. Wait for the TBE being asserted.
8. Write the data to the USART\_TDATA register.
9. Wait until TC=1 to finish.

**Figure 17-3. USART transmit procedure**



It is necessary to wait for the TC bit asserted before disabling the USART or entering the power saving mode.

Reading the USART\_STAT then writing the USART\_TDATA can clear the TC bit. And writing '0' directly to TC bit can also clear the TC bit for multibuffer communication

The break frame is sent when the SBKCMD bit is set, and SBKCMD bit is reset after the transmission.

### 17.3.4. USART receiver

After power on, the USART receiver can be enabled by the following procedure:

1. Set the UEN bit in USART\_CTL0 to enable the USART.
2. Write the WL bit in USART\_CTL0 to set the data bits length.
3. Set the STB[1:0] bits in USART\_CTL1.

4. Enable DMA (DENR bit) in USART\_CTL2 if multibuffer communication is selected.
5. Set the baud rate in USART\_BAUD.
6. Set the REN bit in USART\_CTL0.

After being enabled, the receiver receives a bit stream after a valid start pulse has been detected. Detection on noisy error, parity error, frame error and overrun error are performed during the reception of a frame.

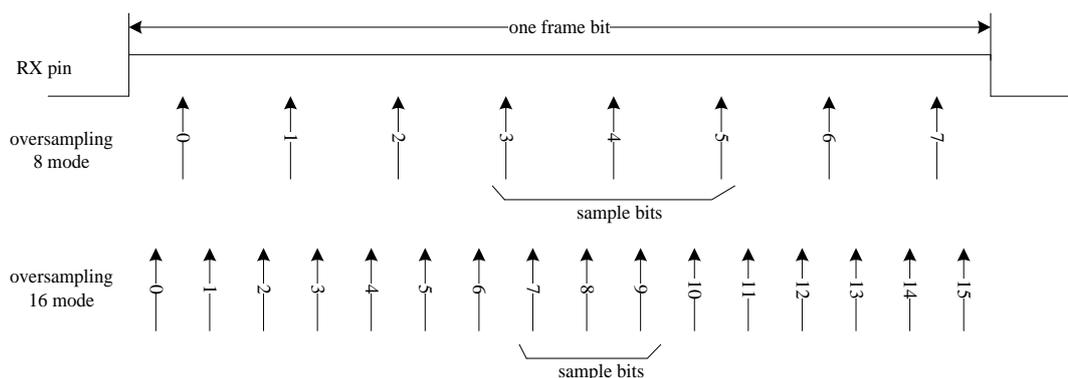
When a frame is received, the RBNE bit in USART\_STAT is asserted, an interrupt is generated if the corresponding interrupt enable bit (RBNEIE) is set in the USART\_CTL0 register. The status bits of the received are stored in the USART\_STAT register.

The software can get the received data by reading the USART\_RDATA register directly, or through DMA. The RBNE status is cleared by a read operation on the USART\_RDATA register, whatever it is performed by software directly, or through DMA.

The REN bit should not be disabled when reception is ongoing, or the current frame will be lost.

By default, the receiver gets three samples to evaluate the value of a frame bit. If the oversampling 8 mode is enabled, the 3rd, 4th and 5th samples are used, while in the oversampling 16 mode, the 7th, 8th, and 9th samples are used. If two or more samples of a frame bit are 0, the frame bit is confirmed as a 0, else 1. If the three samples of any bit of a frame are not the same, whatever it is a start bit, data bit, parity bit or stop bit, a noisy error (NERR) status will be generated for the frame. An interrupt is generated, If the receive DMA is enabled and the ERRIE bit in USART\_CTL2 register is set. If the OSB bit in USART\_CTL2 register is set, the receiver gets only one sample to evaluate a bit value. In this situation, no noisy error will be detected.

**Figure 17-4. Oversampling method of a receive frame bit (OSB=0)**



If the parity check function is enabled by setting the PCEN bit in the USART\_CTL0 register, the receiver calculates the expected parity value while receiving a frame. The received parity bit will be compared with this expected value. If they are not the same, the parity error (PERR) bit in USART\_STAT register will be set. An interrupt is generated, if the PERRIE bit in USART\_CTL0 register is set.

If the RX pin is evaluated as 0 during a stop bit, the frame error (FERR) bit in USART\_STAT register will be set. An interrupt is generated, If the receive DMA is enabled and the ERRIE bit in USART\_CTL2 register is set.

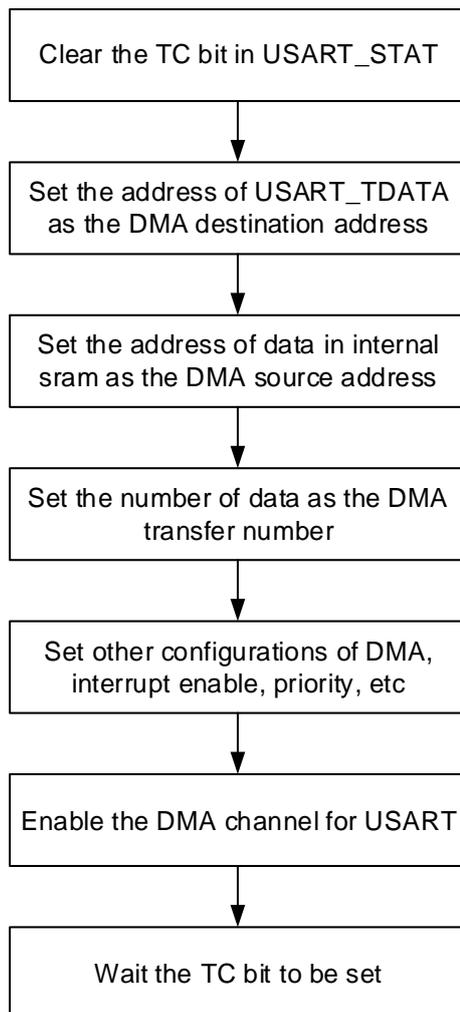
When a frame is received, if the RBNE bit is not cleared yet, the last frame will not be stored in the receive data buffer. The overrun error (ORERR) bit in USART\_STAT register will be set. An interrupt is generated, if the receive DMA is enabled and the ERRIE bit in USART\_CTL2 register is set, or if the RBNEIE is set.

The RBNE, NERR, PERR, FERR and ORERR flags of a reception are always set at the same time. If the receive DMA is not enabled, software can check NERR, PERR, FERR and ORERR flags when serving the RBNE interrupt.

### 17.3.5. Use DMA for data buffer access

To reduce the burden of the processor, DMA can be used to access the transmitting and receiving data buffer. The DENT bit in USART\_CTL2 is used to enable the DMA transmission, and the DENR bit in USART\_CTL2 is used to enable the DMA reception.

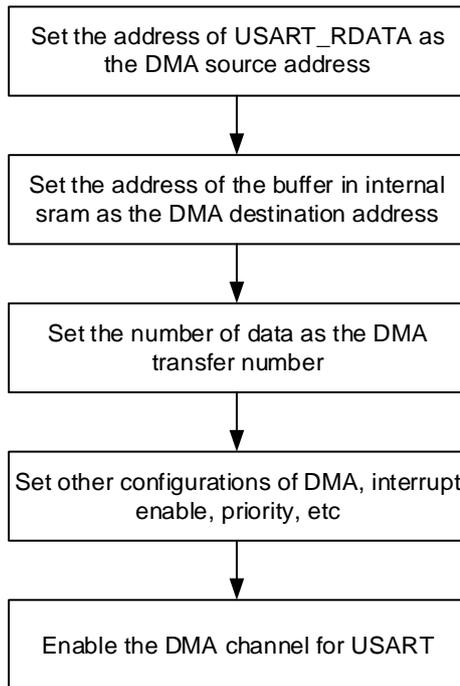
When DMA is used for USART transmission, DMA transfers data from internal sram to the transmit data buffer of the USART. The configuration step is shown in [Figure 17-5. Configuration step when using DMA for USART transmission](#)

**Figure 17-5. Configuration step when using DMA for USART transmission**


After all of the data frames are transmitted, the TC bit in USART\_STAT is set. An interrupt occurs if the TCIE bit in USART\_CTL0 is set.

When DMA is used for USART reception, DMA transfers data from the receive data buffer of the USART to the internal sram. The configuration step is shown in [Figure 17-6. Configuration step when using DMA for USART reception](#). If the ERRIE bit in USART\_CTL2 is set, interrupts can be generated by the Error status bits (FERR, ORERR and NERR) in USART\_STAT.

**Figure 17-6. Configuration step when using DMA for USART reception**

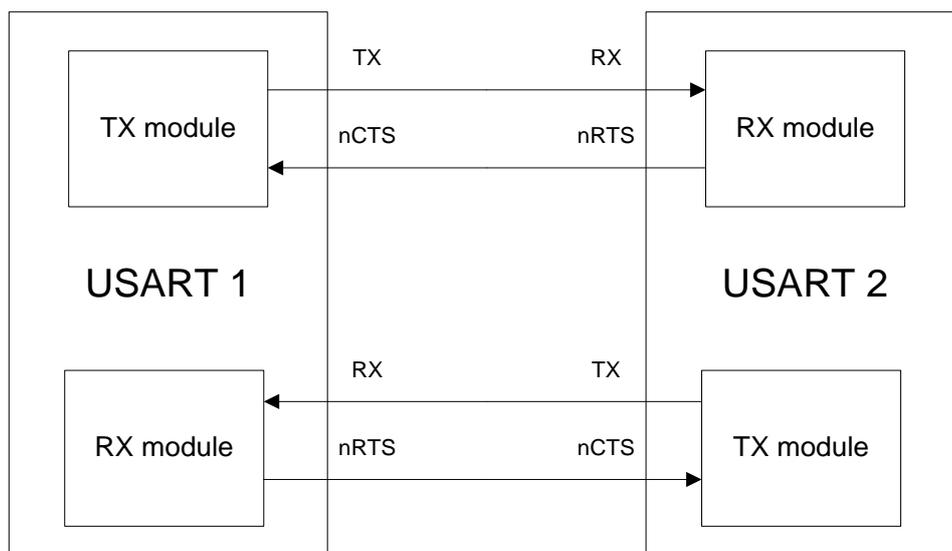


When the number of the data received by USART reaches the DMA transfer number, an end of transfer interrupt can be generated in the DMA module.

### 17.3.6. Hardware flow control

The hardware flow control function is realized by the nCTS and nRTS pins. The RTS flow control is enabled by writing '1' to the RTSEN bit in USART\_CTL2 and the CTS flow control is enabled by writing '1' to the CTSEN bit in USART\_CTL2.

**Figure 17-7. Hardware flow control between two USARTs**



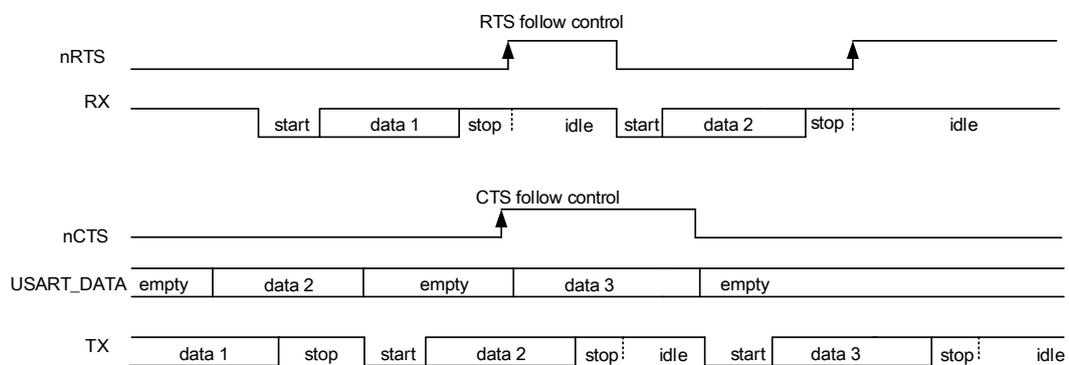
### RTS flow control

USART receiver can receive data only when the nRTS signal is low, and the signal does not go high until the data frame reception is finished. The next reception occurs when the nRTS signal goes low again. The signal keeps high when the receive register is full.

### CTS flow control

If the TBE bit in USART\_STAT is '0' and the nCTS signal is low, the transmitter transmits the data frame. When the nCTS signal goes high during a transmission, the transmitter stops after the current transmission is accomplished.

**Figure 17-8. Hardware flow control**



### RS485 Driver Enable

The driver enable feature, which is enabled by setting bit DEM in the USART\_CTL2 control register, allows the user to activate the external transceiver control, through the DE (Driver Enable) signal. The assertion time, which is programmed using the DEA [4:0] bits field in the USART\_CTL0 control register, is the time between the activation of the DE signal and the beginning of the START bit. The de-assertion time, which is programmed using the DED [4:0] bits field in the USART\_CTL0 control register, is the time between the end of the last stop bit and the de-activation of the DE signal. The polarity of the DE signal can be configured using the DEP bit in the USART\_CTL2 control register.

#### 17.3.7. Multi-processor communication

In multiprocessor communication, several USARTs are connected as a network. It will be a big burden for a device to monitor all of the messages on the RX pin. To reduce the burden of a device, software can put an USART module into a mute mode by writing 1 to the MMCMD bit in USART\_CMD register.

If a USART is in mute mode, all of the receive status bits cannot be set. The USART can also be wake up by hardware by one of the two methods: idle frame method and address match method.

The idle frame wake up method is selected by default. When an idle frame is detected on the RX pin, the hardware clears the RWU bit and exits the mute mode. When wake up at an idle frame, the IDLEF bit in USART\_STAT is not set.

When the WM bit of in USART\_CTL0 register is set, the MSB bit of a frame is detected as the address flag. If the address flag is high, the frame is treated as an address frame. If the address flag is low, the frame is treated as a data frame. If the LSB 4 or 7 bits, which are configured by the ADDM bit of the USART\_CTL1 register, of an address frame is the same as the ADDR bits in the USART\_CTL1 register, the hardware clears the RWU bit and exits the mute mode. The RBNE bit is set for the frame that wakes up the USART. The status bits are available in the USART\_STAT register. If the LSB 4/7 bits of an address frame differs from the ADDR bits in the USART\_CTL1 register, the hardware sets the RWU bit and enters mute mode automatically. In this situation, the RBNE bit is not set.

If the PCEN bit in USART\_CTL0 is set, the MSB bit will be checked as the parity bit, and the bit preceding the MSB bit is detected as the address flag. If the ADDM bit is set and the receive frame is a 7bit data, the LSB 6 bits will be compared with ADDR[5:0]. If the ADDM bit is set and the receive frame is a 9bit data, the LSB 8 bits will be compared with ADDR[7:0].

### 17.3.8. LIN mode

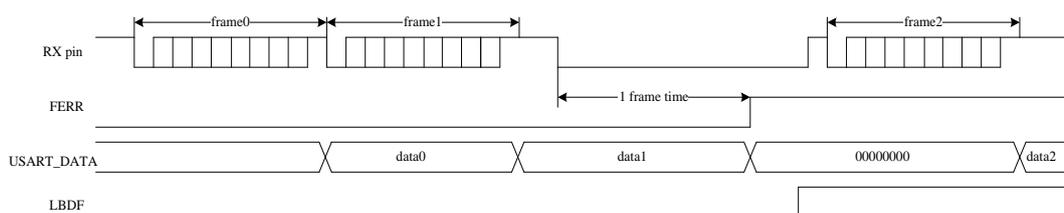
The local interconnection network mode is enabled by setting the LMEN bit in USART\_CTL1. The CKEN, STB[1:0] bit in USART\_CTL1 and the SCEN, HDEN, IREN bits in USART\_CTL2 should be reset in LIN mode.

The LIN transmission procedure is almost the same as the normal transmission procedure. The data bits length can only be 8. And the break frame is 13-bit '0', followed by 1 stop bit.

The break detection function is totally independent from the normal USART receiver. So a break frame can be detected during the idle state or during a frame. The expected length of a break frame can be selected by LBLEN in USART\_CTL1. When the RX pin is detected at low state for a time that is equal to or longer than the expected break frame length (10 bits when LBLEN=0, or 11 bits when LBLEN=1), the LBDF in USART\_STAT is set. An interrupt occurs if the LBDIE bit in USART\_CTL1 is set.

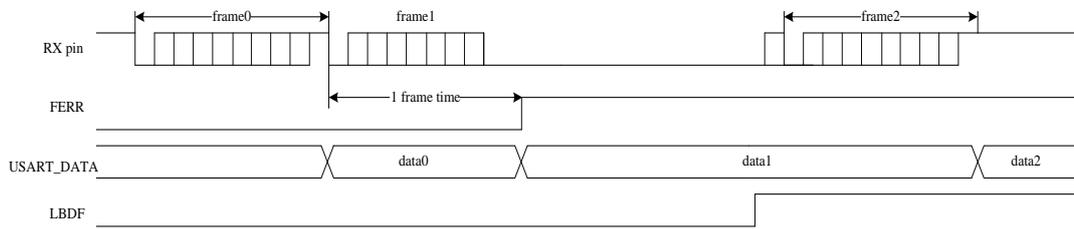
As shown in [Figure 17-9. Break frame occurs during idle state](#), if a break frame occurs during the idle state on the RX pin, the USART receiver will receive an all '0' frame, with an asserted FERR status.

**Figure 17-9. Break frame occurs during idle state**



As shown in [Figure 17-10. Break frame occurs during a frame](#), if a break frame occurs during a frame on the RX pin, the FERR status will be asserted for the current frame.

**Figure 17-10. Break frame occurs during a frame**



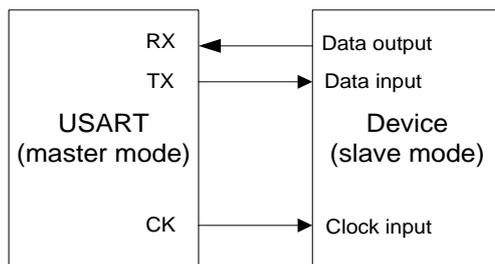
### 17.3.9. Synchronous mode

The USART can be used for full-duplex synchronous serial communications only in master mode, by setting the CKEN bit in USART\_CTL1. The LMEN bit in USART\_CTL1 and SCEN, HDEN, IREN bits in USART\_CTL2 should be reset in synchronous mode. The CK pin is the synchronous USART transmitter clock output, and can be only activated when the TEN bit is enabled. No clock pulse will be sent to the CK pin during the start bit and stop bit transmission. The CLEN bit in USART\_CTL1 can be used to determine whether the clock is output or not during the LSB (address index) bit transmission. The clock output is also not activated during idle and break frame sending. The CPH bit in USART\_CTL1 can be used to determine whether data is captured on the first or the second clock edge. The CPL bit in USART\_CTL1 can be used to configure the clock polarity in the USART Synchronous Mode idle state.

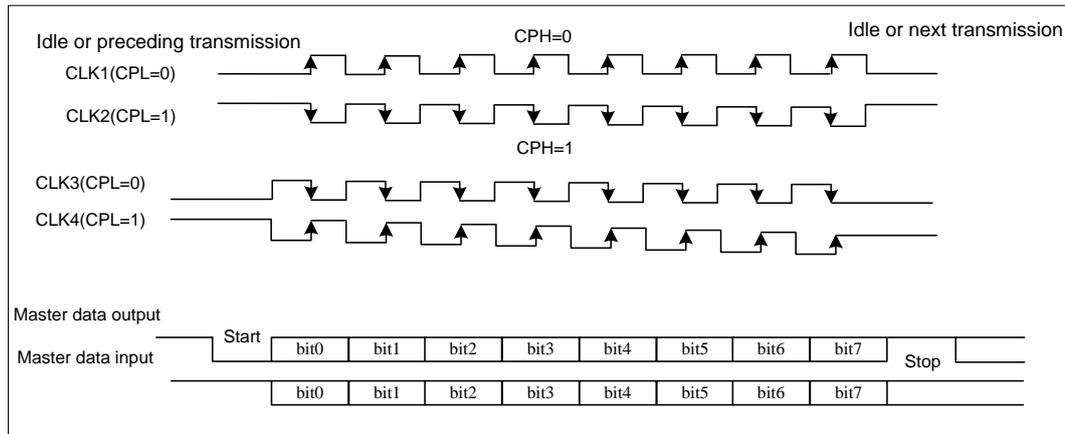
These 3 bits (CPL, CPH, and CLEN) should not be changed while the transmitter or the receiver is enabled

The clock is synchronized with the data transmitted. The receiver in synchronous mode samples the data on the transmitter clock without any oversampling.

**Figure 17-11. Example of USART in synchronous mode**



**Figure 17-12. 8-bit format USART synchronous waveform (CLEN=1)**

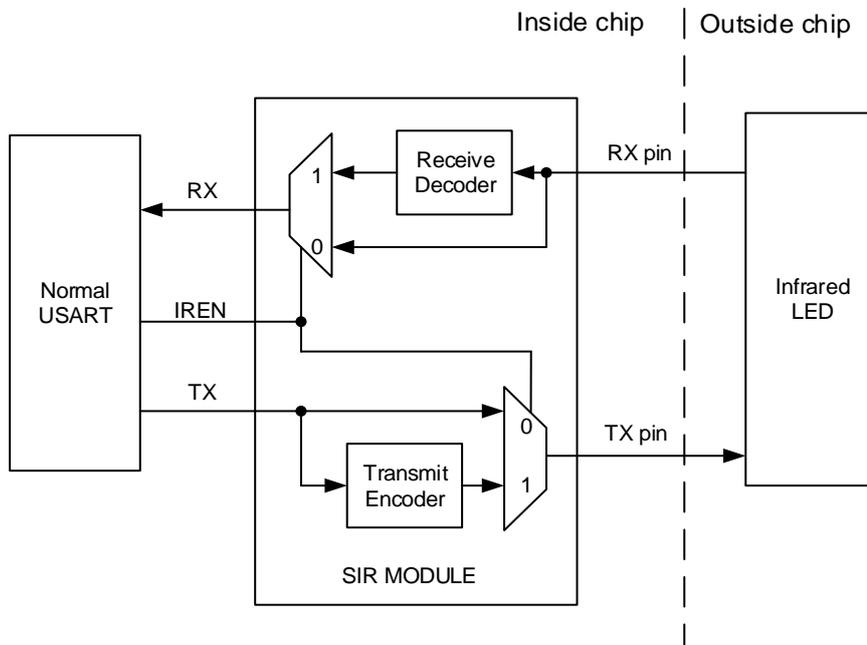


### 17.3.10. IrDA SIR ENDEC mode

The IrDA mode is enabled by setting the IREN bit in USART\_CTL2. The LMEN, STB[1:0], CKEN bits in USART\_CTL1 and HDEN, SCEN bits in USART\_CTL2 should be reset in IrDA mode.

In IrDA mode, the USART transmission data frame is modulated in the SIR transmit encoder and transmitted to the infrared LED through the TX pin. The SIR receive decoder receives the modulated signal from the infrared LED through the RX pin, and puts the demodulated data frame to the USART receiver. The baud rate should not be larger than 115200 for the encoder.

**Figure 17-13. IrDA SIR ENDEC module**

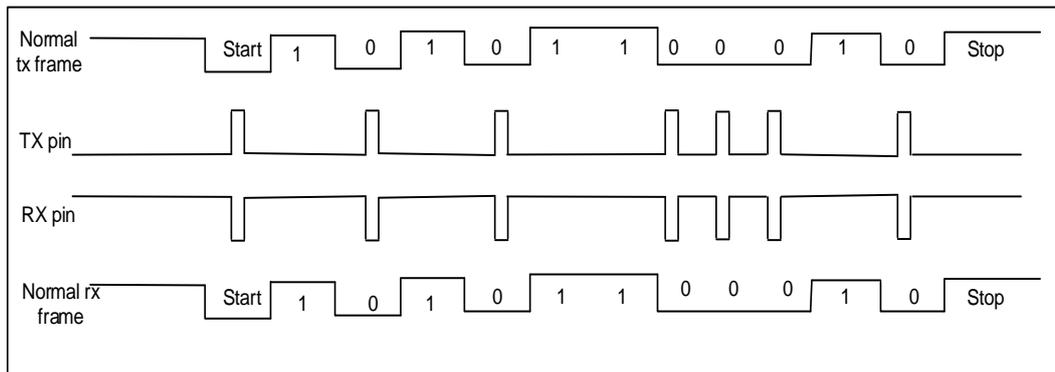


In IrDA mode, the polarity of the TX and RX pins is different. The TX pin is usually at low state, while the RX pin is usually at high state. The IrDA pins keep stable to represent the logic '1', while an infrared light pulse on the IrDA pins (a Return to Zero signal) represents the logic '0'.

The pulse width should be 3/16 of a bit period. The IrDA could not detect any pulse if the pulse width is less than 1 PSC clock. While it can detect a pulse by chance if the pulse width is greater than 1 but smaller than 2 times PSC clock.

Because the IrDA is a half-duplex protocol, the transmission and the reception should not be carried out at the same time in the IrDA SIR ENDEC block.

**Figure 17-14. IrDA data modulation**



The SIR submodule can work in low power mode by setting the IRLP bit in USART\_CTL2. The transmit encoder is driven by a low speed clock, which is divided from the PCLK. The divide ratio is configured by the PSC[7:0] bits in USART\_GP register. The pulse width on the TX pin is 3 cycles of this low speed clock. The receiver decoder works in the same manner as the normal IrDA mode.

### 17.3.11. Half-duplex communication mode

The half-duplex communication mode is enabled by setting the HDEN bit in USART\_CTL2. The LMEN, CKEN bits in USART\_CTL1 and SCEN, IREN bits in USART\_CTL2 should be reset in half-duplex communication mode.

Only one wire is used in half-duplex mode. The TX and RX pins are connected together internally. The TX pin should be configured as IO pin. The conflicts should be controlled by the software. When the TEN bit is set, the data in the data register will be sent.

### 17.3.12. Smartcard (ISO7816) mode

The smartcard mode is an asynchronous mode, which is enabled by setting the SCEN bit in USART\_CTL2. The LMEN bit in USART\_CTL1 and HDEN, IREN bits in USART\_CTL2 should be reset in smartcard mode.

A clock is provided to the smartcard if the CKEN bit is set. The clock can be divided for other use.

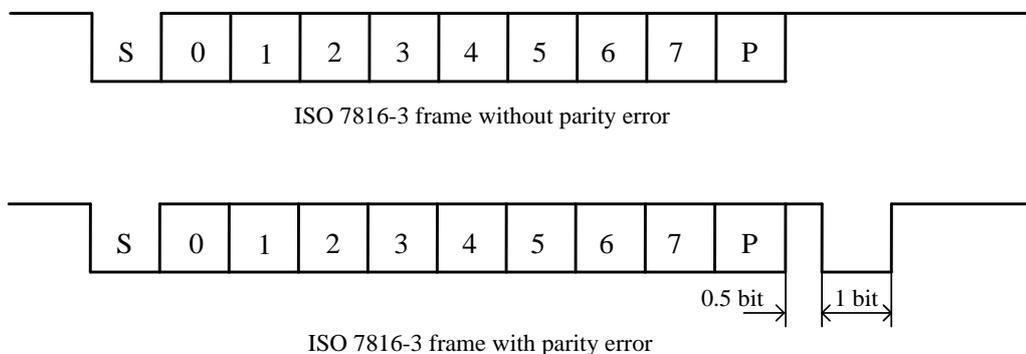
The frame consists of 1 start bit, 9 data bits (1 parity bit included) and 1.5 stop bits.

The smartcard mode is a half-duplex communication protocol. When connected to a

smartcard, the TX pin must be configured as open drain and drives a bidirectional line that is also driven by the smartcard.

## T=0 mode

**Figure 17-15. ISO7816-3 frame format**



Comparing to the time in normal operation, the transmission time from transmit shift register to the TX pin is delayed half baud clock, and the TC flag assertion time delayed a certain value wrote in the guard time register. The USART can automatically re-send data according to the protocol by SCRTNUM times. At the end of reception of the last repeated character the TC bit is set without gardtime immediately. The USART will stop transmitting and signal the error as a framing error if it continues receiving the NACK after the programmed number of retries. The TXFCMD bit in the USART\_CMD register can be used to clear the TBE bit.

During USART reception, the TX line is pulled low for a baud clock after finishing receiving the frame if a parity error is detected. This signal is the 'NACK' signal to smartcard. Then a frame error occurred in smartcard side. The RBNE/receive DMA request is not activated if the received character is erroneous. According to the protocol, the smartcard can resend the data. The USART stops transmitting the NACK and signals the error as a parity error if the received character is still erroneous after the maximum number of retries specified in the SCARNUM bit field.

The 'NACK' signal will be sent to the USART if the NKEN bit in USART\_CTL2 is set. And the USART will not take the 'NACK' signal as the start bit.

The idle frame and break frame do not apply for the smartcard mode.

## T=1 mode (block mode)

In T=1 (block) mode, the NKEN bit in the USART\_CTL2 register should be cleared to deactivate the parity error transmission.

When requesting a read from the smartcard, the USART\_RT register should be programmed with the BWT (block wait time) - 11 value and RBNEIE must be set. A timeout interrupt will be generated, if no answer is received from the card before the expiration of this period. If the first character is received before the expiration of the period, it is signaled by the RBNE interrupt. If DMA is used to read from the smartcard in block mode, the DMA must be enabled

only after the first received byte.

In order to allow the automatic check of the maximum wait time between two consecutive characters, the USART\_RT register must be programmed to the CWT (character wait time) - 11 value, which is expressed in baudtime units, after the reception of the first character (RBNE interrupt). The USART signals to the software through the RT flag and interrupt (when RTIE bit is set), if the smartcard doesn't send a new character in less than the CWT period after the end of the previous character.

The USART uses a block length counter, which is reset when the USART is transmitting (TBE=0), to count all the characters received. The length of the block, which must be programmed to the BL field in the USART\_RT register, is communicated by the smartcard in the third byte of the block (prologue field). This register field must be programmed to the minimum value (0x0), before the start of the block, when using DMA mode. With this value, an interrupt is generated after the 4th received character. The software must read the third byte as block length from the receive buffer.

In interrupt driven receive mode, the length of the block may be checked by software or by programming the BL value. However, before the start of the block, the maximum value of BL (0xFF) may be programmed. The real value will be programmed after the reception of the third character.

The total block length (including prologue, epilogue and information fields) equals BL+4. The end of the block is signaled to the software through the EBF flag and interrupt (when EBIE bit is set). The RT interrupt may occur in case of an error in the block length.

### **Direct and inverse convention**

The smartcard protocol defines two conventions: direct and inverse.

The direct convention is defined as: LSB first, logical bit value of 1 corresponds to H state of the line and parity is even. In this case, the following control bits must be programmed: MSBF=0, DINV=0 (default values).

The inverse convention is defined as: MSB first, logical bit value 1 corresponds to an L state on the signal line and parity is even. In this case, the following control bits must be programmed: MSBF=1, DINV=1.

### **17.3.13. Auto baudrate detection**

The USART is able to detect and automatically set the USART\_BAUD register value based on the reception of one character. There are two methods which can be chosen through the ABDM bits in the USART\_CTL1 register. These methods are:

1. The USART will measure the duration of the start bit (falling edge to rising edge). In this case the receiving pattern should be any character starting with a bit at 1.
2. The USART will measure the duration of the start and of the 1st data bit. The measure

is done falling edge to falling edge, ensuring a better accuracy in the case of slow signal slopes. In this case, the receiving pattern should be any character starting with 10xx bits.

#### 17.3.14. ModBus communication

The USART offers basic support for the implementation of ModBus/RTU and ModBus/ASCII protocols by implementing an end of block detection.

In the ModBus/RTU mode, the end of one block is recognized by an idle line for more than 2 characters time. This function is implemented through the programmable timeout function.

To detect the idle line, the RTEN bit in the USART\_CTL1 register and the RTIE in the USART\_CTL0 register must be set. The USART\_RT register must be set to the value corresponding to a timeout of 2 characters time. After the last stop bit is received, when the receive line is idle for this duration, an interrupt will be generated, informing the software that the current block reception is completed.

In the ModBus/ASCII mode, the end of a block is recognized by a specific (CR/LF) character sequence. The USART manages this mechanism using the character match function by programming the LF ASCII code in the ADDR field and activating the address match interrupt (AMIE=1). When a LF has been received or can check the CR/LF in the DMA buffer, the software will be informed.

#### 17.3.15. Wakeup from Deep-sleep mode

The USART is able to wake up the MCU from Deep-sleep mode by the standard RBNE interrupt or the WUM interrupt.

The UESM bit must be set and the USART clock must be set to IRC8M or LXTAL (refer to the reset and clock unit RCU section).

When using the standard RBNE interrupt, the RBNEIE bit must be set before entering Deep-sleep mode.

When using the WUIE interrupt, the source of WUIE interrupt may be selected through the WUM bit fields.

DMA must be disabled before entering Deep-sleep mode. Before entering Deep-sleep mode, software must check that the USART is not performing a transfer, by checking the BSY flag in the USART\_STAT register. The REA bit must be checked to ensure the USART is actually enabled.

When the wakeup event is detected, the WUF flag is set by hardware and a wakeup interrupt is generated if the WUIE bit is set, independently of whether the MCU is in stop or active mode.

### 17.3.16. USART interrupts

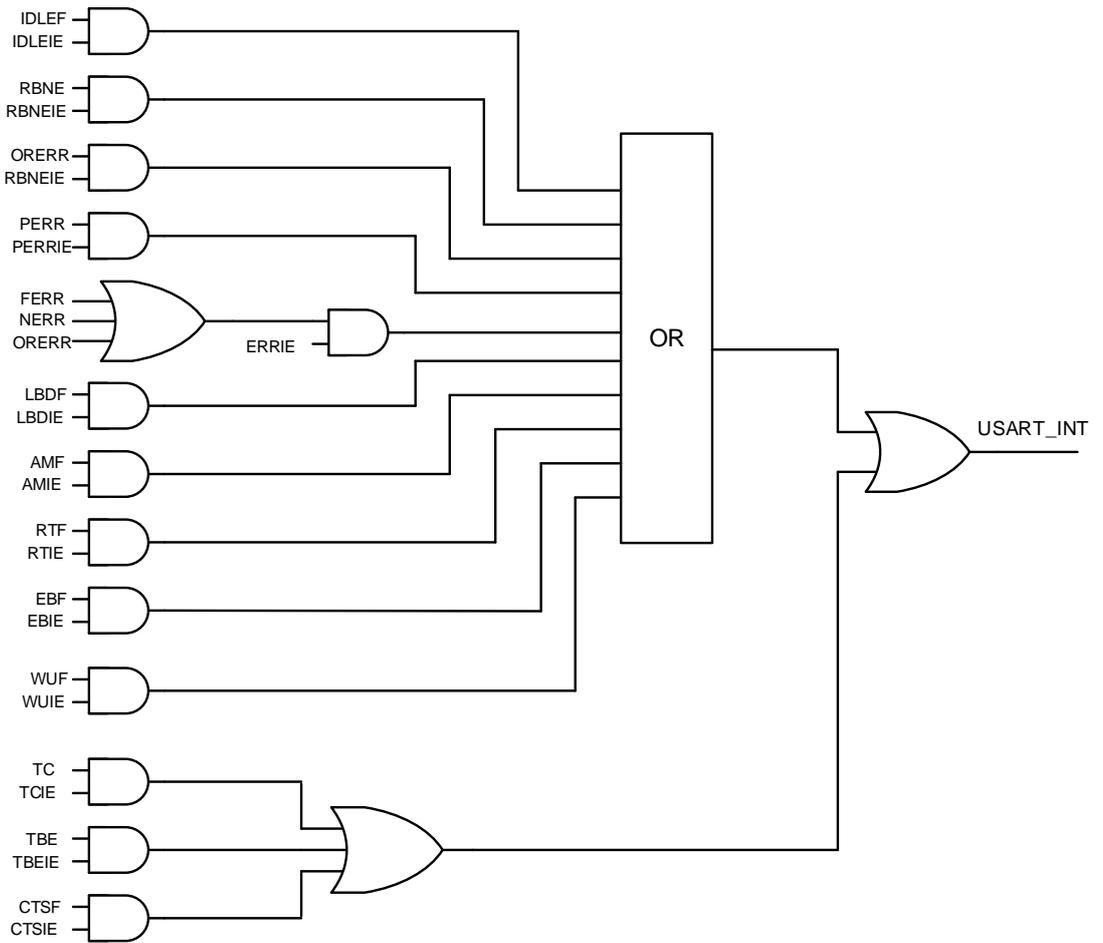
The USART interrupt events and flags are listed in the table below.

**Table 17-3. USART interrupt requests**

Interrupt event	Event flag	Enable Control bit
Transmit data register empty	TBE	TBEIE
CTS flag	CTSF	CTSIE
Transmission complete	TC	TCIE
Received data ready to be read	RBNE	RBNEIE
Overrun error detected	ORERR	
Idle line detected	IDLEF	IDLEIE
Parity error flag	PERR	PERRIE
Break detected flag in LIN mode	LBDP	LBDIE
Reception Errors (Noise flag, overrun error, framing error) in DMA reception	NERR or ORERR or FERR	ERRIE
Character match	AMF	AMIE
Receiver timeout error	RTF	RTIE
End of Block	EBF	EBIE
Wakeup from Deep-sleep mode	WUF	WUIE

All of the interrupt events are ORed together before being sent to the interrupt controller, so the USART can only generate a single interrupt request to the controller at any given time. Software can service multiple interrupt events in a single interrupt service routine

**Figure 17-16. USART interrupt mapping diagram**



## 17.4. Register definition

### 17.4.1. Control register 0 (USART\_CTL0)

Address offset: 0x00

Reset value: 0x0000\_0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				EBIE	RTIE	DEA[4:0]				DED[4:0]					
				rw	rw	rw				rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OVSMOD	AMIE	MEN	WL	WM	PCEN	PM	PERRIE	TBEIE	TCIE	RBNEIE	IDLEIE	TEN	REN	UESM	UEN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:28	Reserved	Must be kept at reset value
27	EBIE	End of Block interrupt enable 0: End of Block interrupt is disabled 1: End of Block interrupt is enabled This bit is reserved in USART1.
26	RTIE	Receiver timeout interrupt enable 0: Receiver timeout interrupt is disabled 1: Receiver timeout interrupt is enabled This bit is reserved in USART1.
25:21	DEA[4:0]	Driver Enable assertion time These bits are used to define the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in sample time units (1/8 or 1/16 bit time), which are configured by the OVSMOD bit. This bit field cannot be written when the USART is enabled (UEN=1).
20:16	DED[4:0]	Driver Enable de-assertion time These bits are used to define the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal. It is expressed in sample time units (1/8 or 1/16 bit time), which are configured by the OVSMOD bit. This bit field cannot be written when the USART is enabled (UEN=1).
15	OVSMOD	Oversample mode 0: Oversampling by 16 1: Oversampling by 8 This bit must be kept cleared in LIN, IrDA and smartcard modes.

		This bit field cannot be written when the USART is enabled (UEN=1).
14	AMIE	ADDR match interrupt enable 0: ADDR match interrupt is disabled 1: ADDR match interrupt is enabled
13	MEN	Mute mode enable 0: Mute mode disabled 1: Mute mode enabled
12	WL	Word length 0: 8 Data bits 1: 9 Data bits This bit field cannot be written when the USART is enabled (UEN=1).
11	WM	Wakeup method in mute mode 0: Idle Line 1: Address Mark This bit field cannot be written when the USART is enabled (UEN=1).
10	PCEN	Parity control enable 0: Parity control disabled 1: Parity control enabled This bit field cannot be written when the USART is enabled (UEN=1).
9	PM	Parity mode 0: Even parity 1: Odd parity This bit field cannot be written when the USART is enabled (UEN=1).
8	PERRIE	Parity error interrupt enable 0: Parity error interrupt is disabled 1: An interrupt will occur whenever the PERR bit is set in USART_STAT.
7	TBEIE	Transmitter register empty interrupt enable 0: Interrupt is inhibited 1: An interrupt will occur whenever the TBE bit is set in USART_STAT
6	TCIE	Transmission complete interrupt enable 0: Transmission complete interrupt is disabled 1: An interrupt will occur whenever the TC bit is set in USART_STAT.
5	RBNEIE	Read data buffer not empty interrupt and overrun error interrupt enable 0: Read data register not empty interrupt and overrun error interrupt disabled 1: An interrupt will occur whenever the ORERR bit is set or the RBNE bit is set in USART_STAT.
4	IDLEIE	IDLE line detected interrupt enable 0: IDLE line detected interrupt disabled

1: An interrupt will occur whenever the IDLEF bit is set in USART\_STAT.

3	TEN	<p>Transmitter enable</p> <p>0: Transmitter is disabled</p> <p>1: Transmitter is enabled</p>
2	REN	<p>Receiver enable</p> <p>0: Receiver is disabled</p> <p>1: Receiver is enabled and begins searching for a start bit</p>
1	UESM	<p>USART enable in Deep-sleep mode</p> <p>0: USART not able to wake up the MCU from Deep-sleep mode.</p> <p>1: USART able to wake up the MCU from Deep-sleep mode. Providing that the clock source for the USART must be IRC8M or LXTAL.</p> <p>This bit is reserved in USART1.</p>
0	UEN	<p>USART enable</p> <p>0: USART prescaler and outputs disabled</p> <p>1: USART prescaler and outputs enabled</p>

## 17.4.2. Control register 1 (USART\_CTL1)

Address offset: 0x04

Reset value: 0x0000\_0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDR[7:0]								RTEN	ABDM[1:0]	ABDEN	MSBF	DINV	TINV	RINV	
rw								rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STRP	LMEN	STB[1:0]	CKEN	CPL	CPH	CLEN	Reserved	LBDIE	LBLEN	ADDM	Reserved				
rw	rw	rw	rw	rw	rw	rw		rw	rw	rw					

Bits	Fields	Descriptions
31:24	ADDR[7:0]	<p>Address of the USART terminal</p> <p>These bits give the address of the USART terminal.</p> <p>In multiprocessor communication during mute mode or Deep-sleep mode, this is used for wakeup with address mark detection. The received frame, the MSB of which is equal to 1, will be compared to these bits. When the ADDM bit is reset, only the ADDR[3:0] bits are used to compare.</p> <p>In normal reception, these bits are also used for character detection. The whole received character (8-bit) is compared to the ADDR[7:0] value and AMF flag is set on matching.</p> <p>This bit field cannot be written when both reception (REN=1) and USART (UEN=1) are enabled.</p>

23	RTEN	Receiver timeout enable 0: Receiver timeout function disabled 1: Receiver timeout function enabled This bit is reserved in USART1.
22:21	ABDM[1:0]	Auto baud rate mode 00: Falling edge to rising edge measurement (measurement of the start bit) 01: Falling edge to falling edge measurement (the received frame must be in a Start 10xxxxxx frame format) 10: Reserved. 11: Reserved This bit field cannot be written when the USART is enabled (UEN=1). This bit is reserved in USART1.
20	ABDEN	Auto baud rate enable 0: Auto baud rate detection is disabled 1: Auto baud rate detection is enabled This bit is reserved in USART1.
19	MSBF	Most significant bit first 0: Data is transmitted/received with the LSB first 1: Data is transmitted/received with the MSB first This bit field cannot be written when the USART is enabled (UEN=1).
18	DINV	Data bit level inversion 0: Data bit signal values are not inverted 1: Data bit signal values are inverted This bit field cannot be written when the USART is enabled (UEN=1).
17	TINV	TX pin level inversion 0: TX pin signal values are not inverted 1: TX pin signal values are inverted This bit field cannot be written when the USART is enabled (UEN=1).
16	RINV	RX pin level inversion 0: RX pin signal values are not inverted 1: RX pin signal values are inverted This bit field cannot be written when the USART is enabled (UEN=1).
15	STRP	Swap TX/RX pins 0: The TX and RX pins functions are not swapped 1: The TX and RX pins functions are swapped This bit field cannot be written when the USART is enabled (UEN=1).
14	LMEN	LIN mode enable 0: LIN mode disabled 1: LIN mode enabled This bit field cannot be written when the USART is enabled (UEN=1).

		This bit is reserved in USART1.
13:12	STB[1:0]	<p>STOP bits length</p> <p>00: 1 Stop bit</p> <p>01: 0.5 Stop bit</p> <p>10: 2 Stop bits</p> <p>11: 1.5 Stop bit</p> <p>This bit field cannot be written when the USART is enabled (UEN=1).</p>
11	CKEN	<p>CK pin enable</p> <p>0: CK pin disabled</p> <p>1: CK pin enabled</p> <p>This bit field cannot be written when the USART is enabled (UEN=1).</p> <p>This bit is reserved in USART1.</p>
10	CPL	<p>Clock polarity</p> <p>0: Steady low value on CK pin outside transmission window in synchronous mode</p> <p>1: Steady high value on CK pin outside transmission window in synchronous mode</p> <p>This bit field cannot be written when the USART is enabled (UEN=1).</p>
9	CPH	<p>Clock phase</p> <p>0: The first clock transition is the first data capture edge in synchronous mode</p> <p>1: The second clock transition is the first data capture edge in synchronous mode</p> <p>This bit field cannot be written when the USART is enabled (UEN=1).</p>
8	CLEN	<p>CK length</p> <p>0: The clock pulse of the last data bit (MSB) is not output to the CK pin in synchronous mode</p> <p>1: The clock pulse of the last data bit (MSB) is output to the CK pin in synchronous mode</p> <p>This bit field cannot be written when the USART is enabled (UEN=1)</p>
7	Reserved	Must be kept at reset value
6	LBDIE	<p>LIN break detection interrupt enable</p> <p>0: LIN break detection interrupt is disabled</p> <p>1: An interrupt will occur whenever the LBDF bit is set in USART_STAT</p> <p>This bit is reserved in USART1.</p>
5	LBLEN	<p>LIN break frame length</p> <p>0: 10 bit break detection</p> <p>1: 11 bit break detection</p> <p>This bit field cannot be written when the USART is enabled (UEN=1).</p> <p>This bit is reserved in USART1.</p>
4	ADDM	<p>Address detection mode</p> <p>This bit is used to select between 4-bit address detection and full-bit address detection.</p> <p>0: 4-bit address detection</p> <p>1: full-bit address detection. In 7-bit, 8-bit and 9-bit data modes, the address detection is</p>

done on 6-bit, 7-bit and 8-bit address (ADDR[5:0], ADDR[6:0] and ADDR[7:0]) respectively

This bit field cannot be written when the USART is enabled (UEN=1).

3:0          Reserved          Must be kept at reset value

### 17.4.3. Control register 2 (USART\_CTL2)

Address offset: 0x08

Reset value: 0x0000\_0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									WUIE	WUM[1:0]		SCRNUM[2:0]			Reserved
									rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEP	DEM	DDRE	OVRD	OSB	CTSIE	CTSEN	RTSEN	DENT	DENR	SCEN	NKEN	HDEN	IRLP	IREN	ERRIE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:23	Reserved	Must be kept at reset value
22	WUIE	Wakeup from Deep-sleep mode interrupt enable 0: Wakeup from Deep-sleep mode interrupt is disabled 1: Wakeup from Deep-sleep mode interrupt is enabled This bit is reserved in USART1.
21:20	WUM[1:0]	Wakeup mode from Deep-sleep mode These bits are used to specify the event which activates the WUF (Wakeup from Deep-sleep mode flag) in the USART_STAT register. 00: WUF active on address match, which is defined by ADDR and ADDM 01: Reserved 10: WUF active on Start bit 11: WUF active on RBNE This bit field cannot be written when the USART is enabled (UEN=1). This bit is reserved in USART1.
19:17	SCRNUM[2:0]	Smartcard auto-retry number In smartcard mode, these bits specify the number of retries in transmission and reception. In transmission mode, a transmission error (FERR bit set) will occur after this number of automatic retransmission retries. In reception mode, reception error (RBNE and PERR bits set) will occur after this number or erroneous reception trials. When these bits are configured as 0x0, there will be no automatic retransmission in

		transmit mode. This bit field is only can be cleared to 0 when the USART is enabled (UEN=1), to stop retransmission. This bit is reserved in USART1.
16	Reserved	Must be kept at reset value
15	DEP	Driver enable polarity mode 0: DE signal is active high 1: DE signal is active low This bit field cannot be written when the USART is enabled (UEN=1).
14	DEM	Driver enable mode This bit is used to activate the external transceiver control, through the DE signal, which is output on the RTS pin. 0: DE function is disabled 1: DE function is enabled This bit field cannot be written when the USART is enabled (UEN=1).
13	DDRE	Disable DMA on reception error 0: DMA is not disabled in case of reception error. The DMA request is not asserted to make sure the erroneous data is not transferred, but the next correct received data will be transferred. The RBNE is kept 0 to prevent overrun, but the corresponding error flag is set. This mode can be used in Smartcard mode 1: DMA is disabled following a reception error. The DMA request is not asserted until the error flag is cleared. The RBNE flag and corresponding error flag will be set. The software must first disable the DMA request (DMAR = 0) or clear RBNE before clearing the error flag This bit field cannot be written when the USART is enabled (UEN=1).
12	OVRD	Overrun disable 0: Overrun functionality is enabled. The ORERR error flag will be set when received data is not read before receiving new data, and the new data will be lost 1: Overrun functionality is disabled. The ORERR error flag will not be set when received data is not read before receiving new data, and the new received data overwrites the previous content of the USART_RDATA register This bit field cannot be written when the USART is enabled (UEN=1).
11	OSB	One sample bit method 0: Three sample bit method 1: One sample bit method This bit field cannot be written when the USART is enabled (UEN=1).
10	CTSIE	CTS interrupt enable 0: CTS interrupt is disabled 1: An interrupt will occur whenever the CTS bit is set in USART_STAT
9	CTSEN	CTS enable

		0: CTS hardware flow control disabled 1: CTS hardware flow control enabled This bit field cannot be written when the USART is enabled (UEN=1).
8	RTSEN	RTS enable 0: RTS hardware flow control disabled 1: RTS hardware flow control enabled, data can be requested only when there is space in the receive buffer This bit field cannot be written when the USART is enabled (UEN=1).
7	DENT	DMA enable for transmission 0: DMA mode is disabled for transmission 1: DMA mode is enabled for transmission
6	DENR	DMA enable for reception 0: DMA mode is disabled for reception 1: DMA mode is enabled for reception
5	SCEN	Smartcard mode enable 0: Smartcard Mode disabled 1: Smartcard Mode enabled This bit field cannot be written when the USART is enabled (UEN=1). This bit is reserved in USART1.
4	NKEN	NACK enable in Smartcard mode 0: Disable NACK transmission when parity error 1: Enable NACK transmission when parity error This bit field cannot be written when the USART is enabled (UEN=1). This bit is reserved in USART1.
3	HDEN	Half-duplex enable 0: Half duplex mode is disabled 1: Half duplex mode is enabled This bit field cannot be written when the USART is enabled (UEN=1).
2	IRLP	IrDA low-power 0: Normal mode 1: Low-power mode This bit field cannot be written when the USART is enabled (UEN=1).
1	IREN	IrDA mode enable 0: IrDA disabled 1: IrDA enabled This bit field cannot be written when the USART is enabled (UEN=1). This bit is reserved in USART1.
0	ERRIE	Error interrupt enable 0: Error interrupt disabled

1: An interrupt will occur whenever the FERR bit or the ORERR bit or the NERR bit is set in USART\_STAT in multibuffer communication

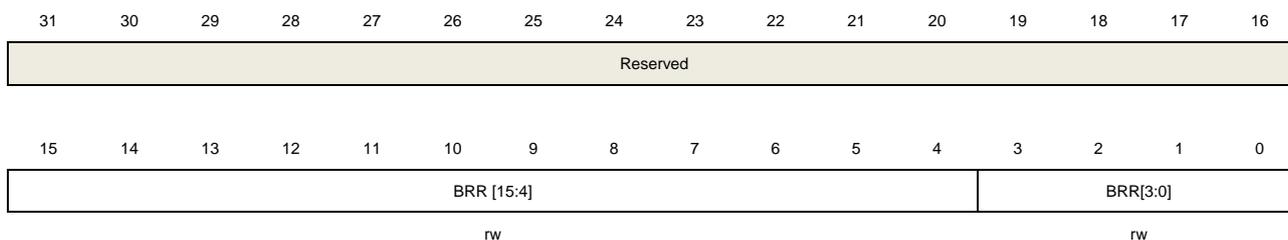
## 17.4.4. Baud rate generator register (USART\_BAUD)

Address offset: 0x0C

Reset value: 0x0000\_0000

This register has to be accessed by word (32-bit)

This register cannot be written when the USART is enabled (UEN=1)



Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:4	BRR[15:4]	Integer of baud-rate divider DIV_INT[11:0] = BRR[15:4]
3:0	BRR [3:0]	Fraction of baud-rate divider If OVSMOD = 0, USARTDIV [3:0] = BRR [3:0]; If OVSMOD = 1, USARTDIV [3:1] = BRR [2:0], BRR [3] must be reset.

## 17.4.5. Prescaler and guard time configuration register (USART\_GP)

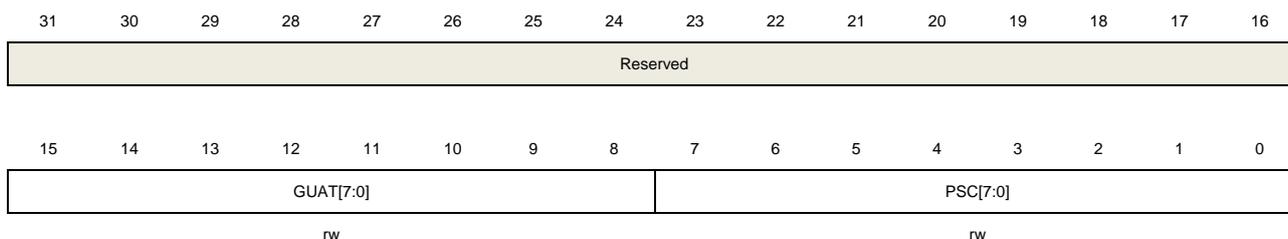
Address offset: 0x10

Reset value: 0x0000\_0000

This register has to be accessed by word (32-bit)

This register cannot be written when the USART is enabled (UEN=1)

This register is reserved in USART1



Bits	Fields	Descriptions
------	--------	--------------

31:16	Reserved	Must be kept at reset value
15:8	GUAT[7:0]	Guard time value in smartcard mode This bit field cannot be written when the USART is enabled (UEN=1).
7:0	PSC[7:0]	<p>Prescaler value for dividing the system clock</p> <p><b>In IrDA Low-power mode</b>, the division factor is the prescaler value.</p> <p>00000000: Reserved - do not program this value</p> <p>00000001: divides the source clock by 1</p> <p>00000010: divides the source clock by 2</p> <p>...</p> <p><b>In IrDA normal mode</b>,</p> <p>00000001: can be set this value only</p> <p><b>In smartcard mode</b>, the prescaler value for dividing the system clock is stored in PSC[4:0] bits. And the bits of PSC[7:5] must be kept at reset value. The division factor is twice as the prescaler value.</p> <p>00000: Reserved - do not program this value</p> <p>00001: divides the source clock by 2</p> <p>00010: divides the source clock by 4</p> <p>00011: divides the source clock by 6</p> <p>...</p> <p>This bit field cannot be written when the USART is enabled (UEN=1).</p>

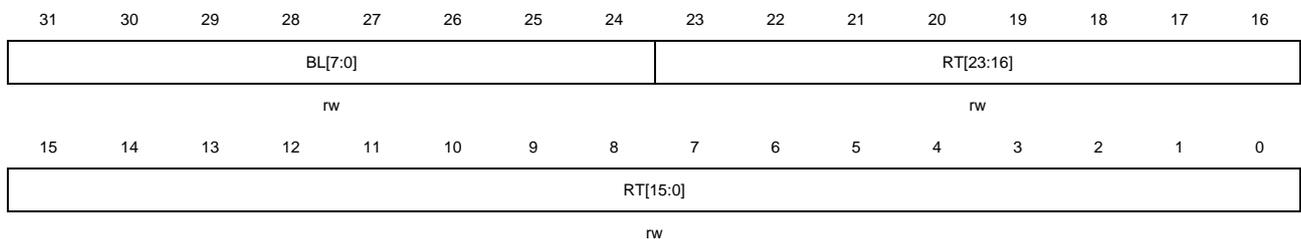
## 17.4.6. Receiver timeout register (USART\_RT)

Address offset: 0x14

Reset value: 0x0000\_0000

This register has to be accessed by word (32-bit)

This bit is reserved in USART1



Bits	Fields	Descriptions
31:24	BL[7:0]	<p>Block Length</p> <p>These bits specify the block length in smartcard T=1 Reception. Its value equals the number of information characters + the length of the Epilogue Field (1-LEC/2-CRC) - 1. This value, which must be programmed only once per received block, can be</p>

programmed after the start of the block reception (using the data from the LEN character in the Prologue Field). The block length counter is reset when TBE=0 in smartcard mode.

In other modes, when REN=0 (receiver disabled) and/or when the EBC bit is written to 1, the Block length counter is reset.

**23:0 RT[23:0]** Receiver timeout threshold

These bits are used to specify receiver timeout value in terms of number of baud clocks. In standard mode, the RTF flag is set if no new start bit is detected for more than the RT value after the last received character.

In smartcard mode, the CWT and BWT are implemented by this value. In this case, the timeout measurement is started from the start bit of the last received character.

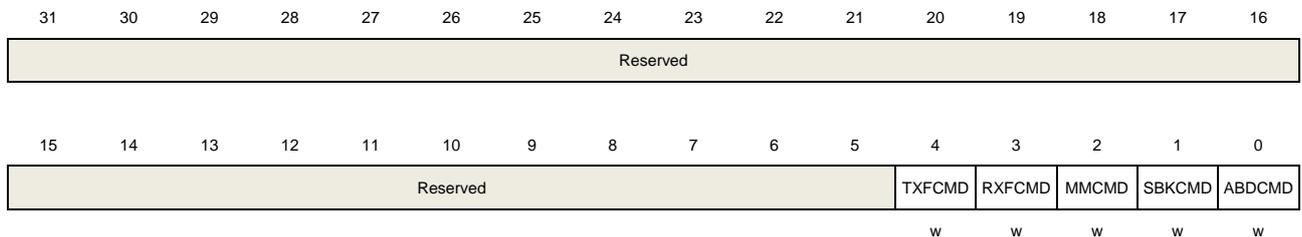
These bits can be written on the fly. The RTF flag will be set if the new value is lower than or equal to the counter. These bits must only be programmed once per received character.

### 17.4.7. Command register (USART\_CMD)

Address offset: 0x18

Reset value: 0x0000\_0000

This register has to be accessed by word (32-bit)



Bits	Fields	Descriptions
31:5	Reserved	Must be kept at reset value
4	TXFCMD	Transmit data flush request Writing 1 to this bit sets the TBE flag, to discard the transmit data. This bit is reserved in USART1.
3	RXFCMD	Receive data flush command Writing 1 to this bit clears the RBNE flag to discard the received data without reading it.
2	MMCMD	Mute mode command Writing 1 to this bit makes the USART into mute mode and sets the RWU flag.
1	SBKCMD	Send break command Writing 1 to this bit sets the SBKF flag and makes the USART send a BREAK frame, as soon as the transmit machine is idle.

0	ABDCMD	Auto baudrate detection command Writing 1 to this bit issues an automatic baud rate measurement command on the next received data frame and resets the ABDF flag in the USART_STAT. This bit is reserved in USART1
---	--------	--

### 17.4.8. Status register (USART\_STAT)

Address offset: 0x1C

Reset value: 0x0000\_00C0

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Reserved										REA	TEA	WUF	RWU	SBF	AMF	BSY
										r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
ABDF	ABDE	Reserved	EBF	RTF	CTS	CTSF	LBDF	TBE	TC	RBNE	IDLEF	ORERR	NERR	FERR	PERR	
r	r		r	r	r	r	r	r	r	r	r	r	r	r	r	

Bits	Fields	Descriptions
31:23	Reserved	Must be kept at reset value
22	REA	Receive enable acknowledge flag This bit, which is set/reset by hardware, reflects the receive enable state of the USART core logic. 0: The USART core receiving logic has not been enabled 1: The USART core receiving logic has been enabled
21	TEA	Transmit enable acknowledge flag This bit, which is set/reset by hardware, reflects the transmit enable state of the USART core logic. 0: The USART core transmitting logic has not been enabled 1: The USART core transmitting logic has been enabled
20	WUF	Wakeup from Deep-sleep mode flag 0: No wakeup from Deep-sleep mode 1: Wakeup from Deep-sleep mode. An interrupt is generated if WUFIE=1 in the USART_CTL2 register and the MCU is in Deep-sleep mode. This bit is set by hardware when a wakeup event, which is defined by the WUM bit field, is detected. Cleared by writing a 1 to the WUC in the USART_INTC register. This bit can also be cleared when UESM is cleared. This bit is reserved in USART1.
19	RWU	Receiver wakeup from mute mode This bit is used to indicate if the USART is in mute mode.

		0: Receiver in active mode 1: Receiver in mute mode It is cleared/set by hardware when a wakeup/mute sequence (address or IDLEIE) is recognized, which is selected by the WAKE bit in the USART_CTL0 register. This bit can only be set by writing 1 to the MMCMD bit in the USART_CMD register when wakeup on IDLEIE mode is selected.
18	SBF	Send break flag 0: No break character is transmitted 1: Break character will be transmitted This bit indicates that a send break character was requested. Set by software, by writing 1 to the SBKCMD bit in the USART_CMD register. Cleared by hardware during the stop bit of break transmission.
17	AMF	ADDR match flag 0: ADDR does not match the received character 1: ADDR matches the received character, An interrupt is generated if AMIE=1 in the USART_CTL0 register. Set by hardware, when the character defined by ADDR [7:0] is received. Cleared by writing 1 to the AMC in the USART_INTC register.
16	BSY	Busy flag 0: USART reception path is idle 1: USART reception path is working
15	ABDF	Auto baudrate detection flag 0: No auto baudrate detection complete 1: Auto baudrate detection complete Set by hardware when the automatic baud rate has been completed. Cleared by writing 1 to the ABDCMD in the USART_CMD register, to request a new auto baudrate detection. This bit is reserved in USART1.
14	ABDE	Auto baudrate detection error 0: No auto baudrate detection error occurred 1: Auto baudrate detection error occurred Set by hardware if the baud rate out of range or character comparison failed Cleared by software, by writing 1 to the ABDRQ bit in the USART_CTL2 register. This bit is reserved in USART1.
13	Reserved	Must be kept at reset value
12	EBF	End of block flag 0: End of Block not reached 1: End of Block (number of characters) reached. An interrupt is generated if the EBIE=1 in the USART_CTL1 register Set by hardware when the number of received bytes (from the start of the block,

		including the prologue) is equal or greater than BLEN + 4. Cleared by writing 1 to EBC bit in USART_INTC register. This bit is reserved in USART1.
11	RTF	Receiver timeout flag 0: Timeout value not reached 1: Timeout value reached without any data reception. An interrupt is generated if RTIE bit in the USART_CTL1 register is set. Set by hardware when the RT value, programmed in the USART_RT register has lapsed without any communication. Cleared by writing 1 to RTC bit in USART_INTC register. The timeout corresponds to the CWT or BWT timings in smartcard mode. This bit is reserved in USART1
10	CTS	CTS level This bit equals to the inverted level of the nCTS input pin. 0: nCTS input pin is in high level 1: nCTS input pin is in low level
9	CTSF	CTS change flag 0: No change occurred on the nCTS status line 1: A change occurred on the nCTS status line. An interrupt will occur if the CTSIE bit is set in USART_CTL2 Set by hardware when the nCTS input toggles. Cleared by writing 1 to CTSC bit in USART_INTC register.
8	LBDF	LIN break detected flag 0: LIN Break is not detected 1: LIN Break is detected. An interrupt will occur if the LBDIE bit is set in USART_CTL1 Set by hardware when the LIN break is detected. Cleared by writing 1 to LBDC bit in USART_INTC register. This bit is reserved in USART1.
7	TBE	Transmit data register empty 0: Data is not transferred to the shift register 1: Data is transferred to the shift register. An interrupt will occur if the TBEIE bit is set in USART_CTL0 Set by hardware when the content of the USART_TDATA register has been transferred into the transmit shift register or writing 1 to TXFCMD bit of the USART_CMD register. Cleared by a write to the USART_TDATA.
6	TC	Transmission completed 0: Transmission is not completed 1: Transmission is complete. An interrupt will occur if the TCIE bit is set in USART_CTL0. Set by hardware if the transmission of a frame containing data is completed and if the TBE bit is set.

		Cleared by writing 1 to TCC bit in USART_INTC register.
5	RBNE	<p>Read data buffer not empty</p> <p>0: Data is not received</p> <p>1: Data is received and ready to be read. An interrupt will occur if the RBNEIE bit is set in USART_CTL0.</p> <p>Set by hardware when the content of the receive shift register has been transferred to the USART_RDATA.</p> <p>Cleared by reading the USART_RDATA or writing 1 to RXFCMD bit of the USART_CMD register.</p>
4	IDLEF	<p>IDLE line detected flag</p> <p>0: No Idle Line is detected</p> <p>1: Idle Line is detected. An interrupt will occur if the IDLEIE bit is set in USART_CTL0</p> <p>Set by hardware when an Idle Line is detected. It will not be set again until the RBNE bit has been set itself.</p> <p>Cleared by writing 1 to IDLEC bit in USART_INTC register.</p>
3	ORERR	<p>Overrun error</p> <p>0: No Overrun error is detected</p> <p>1: Overrun error is detected. An interrupt will occur if the RBNEIE bit is set in USART_CTL0. In multibuffer communication, an interrupt will occur if the ERRIE bit is set in USART_CTL2.</p> <p>Set by hardware when the word in the receive shift register is ready to be transferred into the USART_RDATA register while the RBNE bit is set.</p> <p>Cleared by writing 1 to OREC bit in USART_INTC register.</p>
2	NERR	<p>Noise error flag</p> <p>0: No noise error is detected</p> <p>1: Noise error is detected. In multibuffer communication, an interrupt will occur if the ERRIE bit is set in USART_CTL2.</p> <p>Set by hardware when noise error is detected on a received frame.</p> <p>Cleared by writing 1 to NEC bit in USART_INTC register.</p>
1	FERR	<p>Frame error flag</p> <p>0: No framing error is detected</p> <p>1: Frame error flag or break character is detected. In multibuffer communication, an interrupt will occur if the ERRIE bit is set in USART_CTL2.</p> <p>Set by hardware when a de-synchronization, excessive noise or a break character is detected. This bit will be set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame), when USART transmits in smartcard mode.</p> <p>Cleared by writing 1 to FEC bit in USART_INTC register.</p>
0	PERR	<p>Parity error flag</p> <p>0: No parity error is detected</p> <p>1: Parity error flag is detected. An interrupt will occur if the PERRIE bit is set in</p>

USART\_CTL0.

Set by hardware when a parity error occurs in receiver mode.

Cleared by writing 1 to PEC bit in USART\_INTC register.

## 17.4.9. Interrupt status clear register (USART\_INTC)

Address offset: 0x20

Reset value: 0x0000\_0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Reserved											WUC	Reserved		AMC	Reserved	
											w			w		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved			EBC	RTC	Reserved	CTSC	LBDC	Reserved	TCC	Reserved	IDLEC	OREC	NEC	FEC	PEC	
			w	w			w	w			w	w	w	w	w	

Bits	Fields	Descriptions
31:21	Reserved	Must be kept at reset value
20	WUC	Wakeup from Deep-sleep mode clear Writing 1 to this bit clears the WUF bit in the USART_STAT register. This bit is reserved in USART1.
19:18	Reserved	Must be kept at reset value
17	AMC	ADDR match clear Writing 1 to this bit clears the AMF bit in the USART_STAT register.
16:13	Reserved	Must be kept at reset value
12	EBC	End of block clear Writing 1 to this bit clears the EBF bit in the USART_STAT register. This bit is reserved in USART1.
11	RTC	Receiver timeout clear Writing 1 to this bit clears the RTF flag in the USART_STAT register. This bit is reserved in USART1.
10	Reserved	Must be kept at reset value
9	CTSC	CTS change clear Writing 1 to this bit clears the CTSF bit in the USART_STAT register.
8	LBDC	LIN break detected clear Writing 1 to this bit clears the LBDF flag in the USART_STAT register. This bit is reserved in USART1.

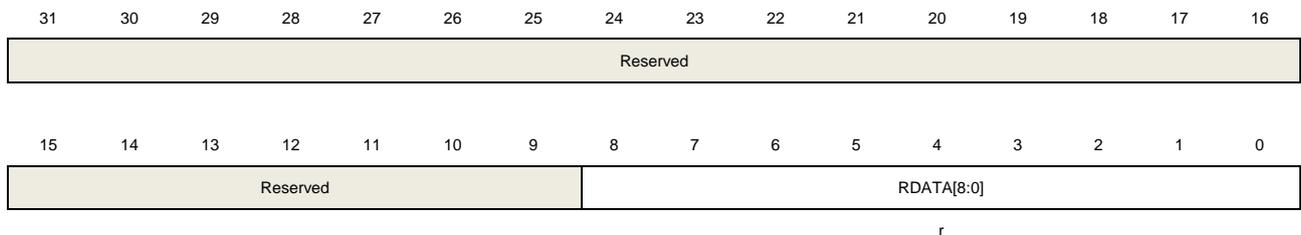
7	Reserved	Must be kept at reset value
6	TCC	Transmission complete clear Writing 1 to this bit clears the TC bit in the USART_STAT register.
5	Reserved	Must be kept at reset value
4	IDLEC	Idle line detected clear Writing 1 to this bit clears the IDLEF bit in the USART_STAT register.
3	OREC	Overrun error clear Writing 1 to this bit clears the ORERR bit in the USART_STAT register.
2	NEC	Noise detected clear Writing 1 to this bit clears the NERR bit in the USART_STAT register.
1	FEC	Frame error flag clear Writing 1 to this bit clears the FERR bit in the USART_STAT register
0	PEC	Parity error clear Writing 1 to this bit clears the PERR bit in the USART_STAT register.

#### 17.4.10. Receive data register (USART\_RDATA)

Offset: 0x24

Reset value: Undefined

This register has to be accessed by word (32-bit)



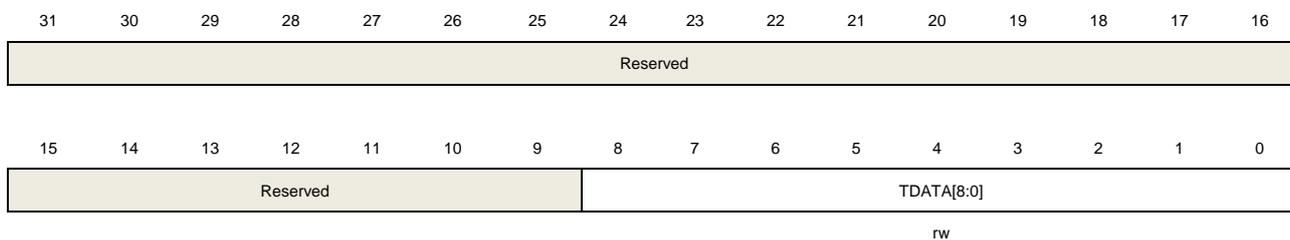
Bits	Fields	Descriptions
31:9	Reserved	Must be kept at reset value
8:0	RDATA[8:0]	Receive Data value The received data character is contained in these bits. The value read in the MSB (bit 7 or bit 8 depending on the data length) will be the received parity bit, if receiving with the parity is enabled (PCEN bit set to 1 in the USART_CTL0 register).

#### 17.4.11. Transmit data register (USART\_TDATA)

Offset: 0x28

Reset value: Undefined

This register has to be accessed by word (32-bit)



Bits	Fields	Descriptions
31:9	Reserved	Must be kept at reset value
8:0	TDATA[8:0]	Transmit Data value The transmit data character is contained in these bits. The value written in the MSB (bit 7 or bit 8 depending on the data length) will be replaced by the parity, when transmitting with the parity is enabled (PCEN bit set to 1 in the USART_CTL0 register). This register must be written only when TBE bit in USART_STAT register is set.

## 18. Inter-integrated circuit interface(I2C)

### 18.1. Overview

The I2C (inter-integrated circuit) module provides an I2C interface which is an industry standard two-line serial interface for MCU to communicate with external I2C interface. I2C bus uses two serial lines: a serial data line, SDA, and a serial clock line, SCL.

The I2C interface implements standard I2C protocol with standard-mode and fast-mode as well as CRC calculation and checking, SMBus (system management bus) and PMBus (power management bus). It also supports multi-master I2C bus. The I2C interface provides DMA mode for users to reduce CPU overload.

### 18.2. Characteristics

- Parallel-bus to I2C-bus protocol converter and interface
- Both master and slave functions with the same interface
- Bi-directional data transfer between master and slave
- Supports 7-bit and 10-bit addressing and general call addressing
- Multi-master capability
- Supports standard-mode (up to 100 kHz) and fast-mode (up to 400 kHz)
- Configurable SCL stretching in slave mode
- Supports DMA mode
- SMBus 2.0 and PMBus compatible
- 2 Interrupts: one for successful byte transmission and the other for error event
- Optional PEC (packet error checking) generation and check

**For GD32F170xx and GD32F190xx devices, additional features are shown below.**

- Support SAM\_V mode

### 18.3. Function overview

[Figure 18-1. I2C module block diagram](#) below provides details on the internal configuration of the I2C interface.

Figure 18-1. I2C module block diagram

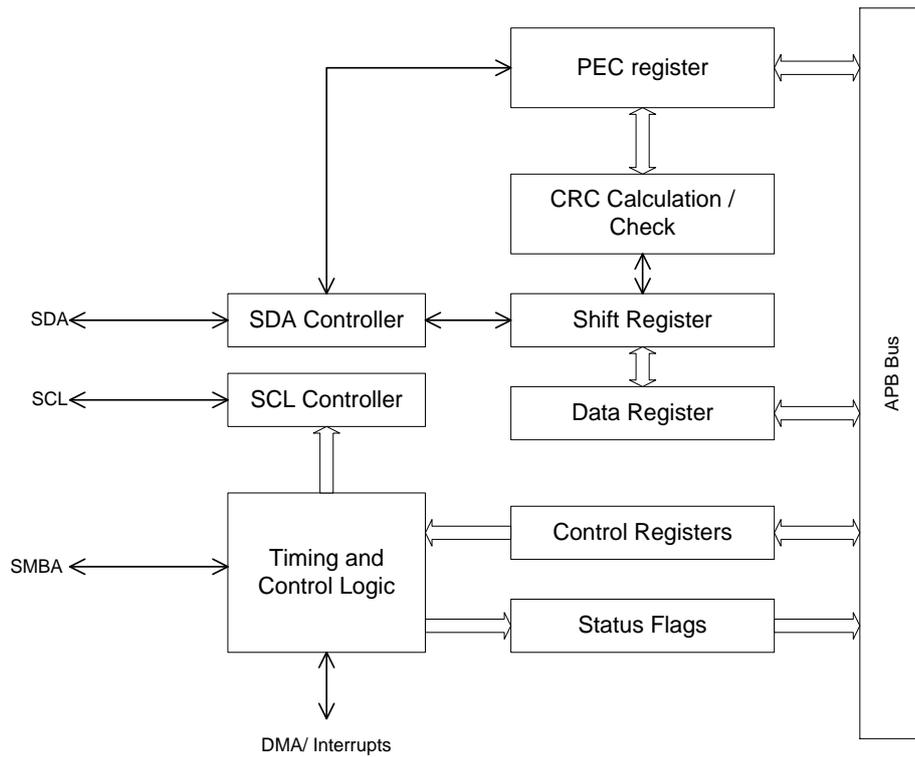


Table 18-1. Definition of I2C-bus terminology (refer to the I2C specification of Philips semiconductors)

Term	Description
Transmitter	the device which sends data to the bus
Receiver	the device which receives data from the bus
Master	the device which initiates a transfer, generates clock signals and terminates a transfer
Slave	the device addressed by a master
Multi-master	more than one master can attempt to control the bus at the same time without corrupting the message
Synchronization	procedure to synchronize the clock signals of two or more devices
Arbitration	procedure to ensure that, if more than one master tries to control the bus simultaneously, only one is allowed to do so and the winning master's message is not corrupted

### 18.3.1. SDA and SCL lines

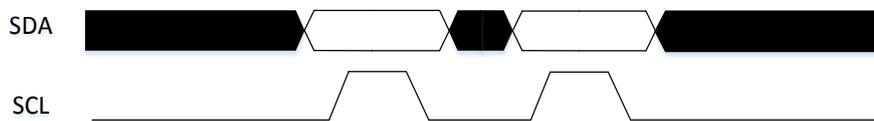
The I2C module has two external lines, the serial data SDA and serial clock SCL lines. The two wires carry information between the devices connected to the bus. Both SDA and SCL are bidirectional lines, connected to a positive supply voltage via current-source or pull-up resistor. When the bus is free, both lines are HIGH. The output stages of devices connected to the bus must have an open-drain or open-collect to perform the wired-AND function. Data

on the I2C-bus can be transferred at rates of up to 100 kbit/s in the standard-mode and up to 400 kbit/s in the fast-mode. Due to the variety of different technology devices (CMOS, NMOS, bipolar) that can be connected to the I2C-bus, the voltage levels of the logical '0' (LOW) and '1' (HIGH) are not fixed and depend on the associated level of  $V_{DD}$ .

### 18.3.2. Data validation

The data on the SDA line must be stable during the HIGH period of the clock. The HIGH or LOW state of the data line can only change when the clock signal on the SCL line is LOW (see [Figure 18-2. Data validation](#)). One clock pulse is generated for each data bit transferred.

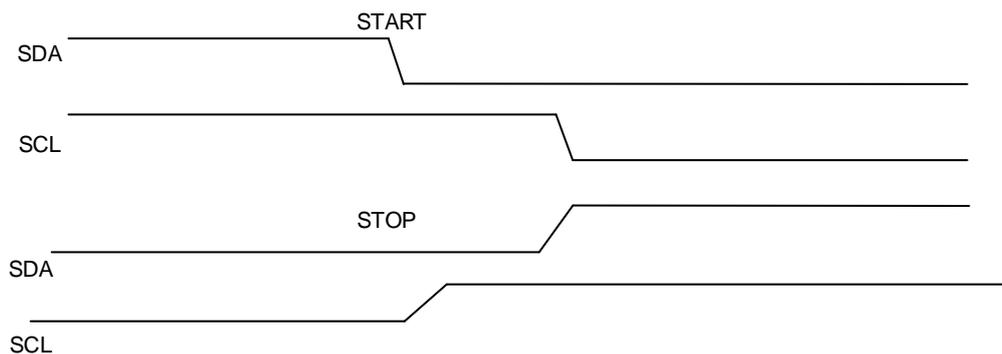
**Figure 18-2. Data validation**



### 18.3.3. START and STOP condition

All transactions begin with a START (S) and are terminated by a STOP (P) (see Figure 18-3). A HIGH to LOW transition on the SDA line while SCL is HIGH defines a START condition. A LOW to HIGH transition on the SDA line while SCL is HIGH defines a STOP condition.

**Figure 18-3. START and STOP condition**

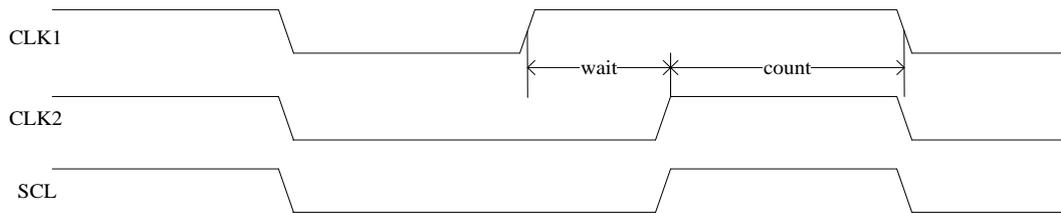


### 18.3.4. Clock synchronization

Two masters can begin transmitting on a free bus at the same time and there must be a method for deciding which master takes control of the bus and complete its transmission. This is done by clock synchronization and bus arbitration. In a single master system, clock synchronization and bus arbitration are unnecessary.

Clock synchronization is performed using the wired-AND connection of I2C interfaces to the SCL line. This means that a HIGH to LOW transition on the SCL line causes the masters concerned to start counting off their LOW period and, once a master clock has gone LOW, it holds the SCL line in that state until the clock HIGH state is reached (see [Figure 18-4. Clock synchronization](#)). However, if another clock is still within its LOW period, the LOW to HIGH transition of this clock may not change the state of the SCL line. The SCL line is therefore held LOW by the master with the longest LOW period. Masters with shorter LOW periods enter a HIGH wait-state during this time.

**Figure 18-4. Clock synchronization**



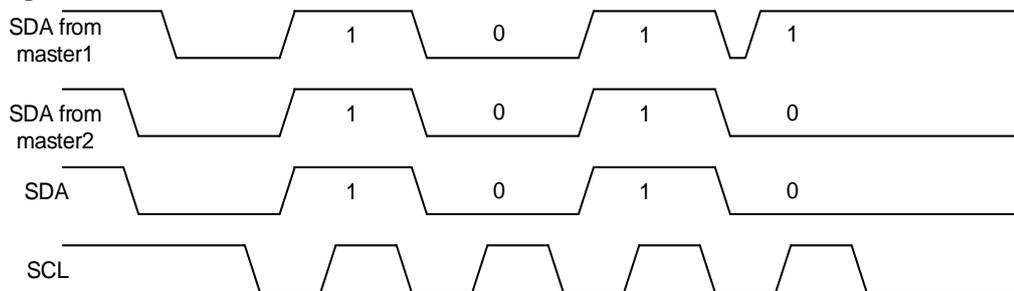
### 18.3.5. Arbitration

Arbitration, like synchronization, is part of the protocol where more than one master is used in the system. Slaves are not involved in the arbitration procedure.

A master may start a transfer only if the bus is free. Two masters may generate a START condition within the minimum hold time of the START condition which results in a valid START condition on the bus. Arbitration is then required to determine which master will complete its transmission.

Arbitration proceeds bit by bit. During every bit, while SCL is HIGH, each master checks to see whether the SDA level matches what it has sent. This process may take many bits. Two masters can even complete an entire transaction without error, as long as the transmissions are identical. The first time a master tries to send a HIGH, but detects that the SDA level is LOW, then the master knows that it has lost the arbitration and turns off its SDA output driver. The other master goes on to complete its transaction.

**Figure 18-5. SDA Line arbitration**



### 18.3.6. I2C communication flow

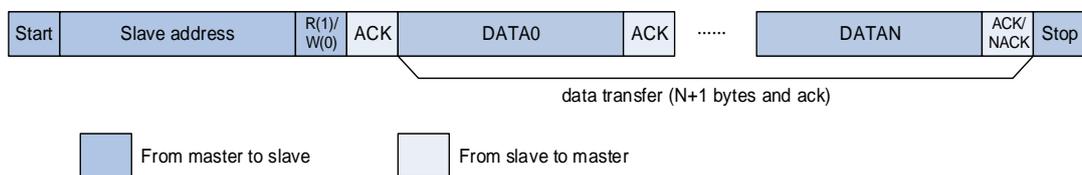
Each I2C device is recognized by a unique address (whether it is a microcontroller, LCD driver,

memory or keyboard interface) and can operate as either a transmitter or receiver, depending on the function of the device.

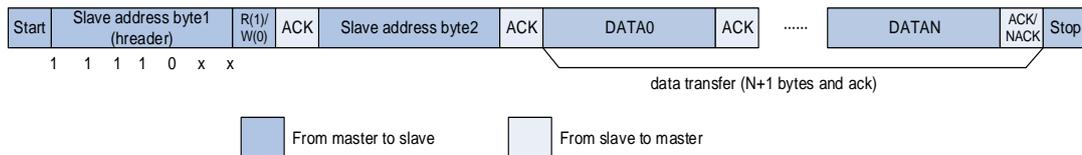
An I2C slave will continue to detect addresses after a START condition on I2C bus and compare the detected address with its slave address which is programmable by software. Once the two addresses match, the I2C slave will send an ACK to the I2C bus and responses to the following command on I2C bus: transmitting or receiving the desired data. Additionally, if General Call is enabled by software, the I2C slave always responses to a General Call Address (0x00). The I2C block support both 7-bit and 10-bit address modes.

An I2C master always initiates or end a transfer using START or STOP condition and it's also responsible for SCL clock generation.

**Figure 18-6. I2C communication flow with 7-bit address.**



**Figure 18-7. I2C communication flow with 10-bit address.**



### 18.3.7. Programming model

An I2C device such as LCD driver may only be a receiver, whereas a memory can both receive and transmit data. In addition to transmitters and receivers, devices can also be considered as masters or slaves when performing data transfers. A master is the device which initiates a data transfer on the bus and generates the clock signals to permit that transfer. At that time, any device addressed is considered a slave.

An I2C device is able to transmit or receive data whether it's a master or a slave, thus, there're 4 operation modes for an I2C device:

- Master Transmitter
- Master Receiver
- Slave Transmitter
- Slave Receiver

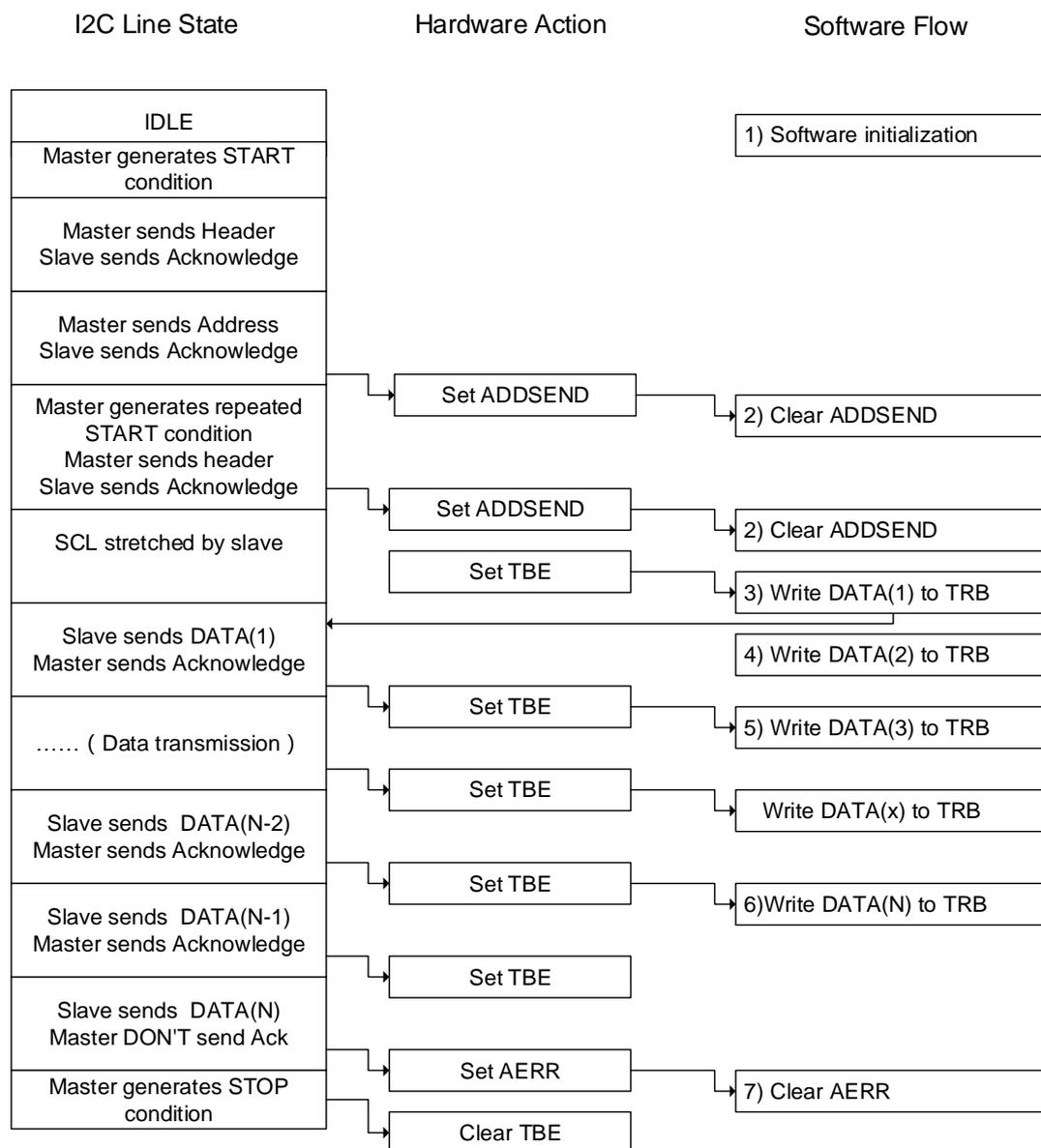
I2C block supports all of the four I2C modes. After system reset, it works in slave mode. If it's programmed by software and finished sending a START condition on I2C bus, it changes into master mode. The I2C changes back to slave mode after it's programmed by software and finished sending a STOP condition on I2C bus.

### Programming model in slave transmitting mode

As is shown in the figure below, the following software procedure should be followed if users wish to make transaction in slave transmitter mode:

1. First of all, software should enable I2C peripheral clock as well as configure clock related registers in I2C\_CTL1 to make sure correct I2C timing. After enabled and configured, I2C operates in its default slave state and waits for START condition followed by address on I2C bus.
2. After receiving a START condition followed by a matched address, either in 7-bit format or in 10-bit format, the I2C hardware sets the ADDSEND bit in I2C\_STAT0 register, which should be monitored by software either by polling or interrupt. After that software should read I2C\_STAT0 and then I2C\_STAT1 to clear ADDSEND bit. If 10-bit addressing format is selected, the I2C master should then send a repeated START(Sr) condition followed by a header to the I2C bus. The slave sets ADDSEND bit again after it detects the repeated START(Sr) condition and the following header. Software needs to clear the ADDSEND bit again by reading I2C\_STAT0 and then I2C\_STAT1.
3. Now I2C enters data transmission stage and hardware sets TBE bit because both the shift register and data register I2C\_DATA are empty. Once TBE is set, Software should write the first byte of data to I2C\_DATA register, TBE is not cleared in this case because the write byte in I2C\_DATA is moved to the internal shift register immediately. I2C begins to transmit data to I2C bus as soon as the shift register is not empty.
4. During the first byte's transmission, software can write the second byte to I2C\_DATA, and this time TBE is cleared because neither I2C\_DATA nor shift register is empty.
5. Any time TBE is set, software can write a byte to I2C\_DATA as long as there are still data to be transmitted.
6. During the second last byte's transmission, software write the last data to I2C\_DATA to clear the TBE flag and doesn't care TBE anymore. So TBE will be set after the byte's transmission and not cleared until a STOP condition.
7. I2C master doesn't acknowledge to the last byte according to the I2C protocol, so after sending the last byte, I2C slave will wait for the STOP condition on I2C bus and sets AERR (Acknowledge Error) bit to notify software that transmission completes. Software clears AERR bit by writing 0 to it.

**Figure 18-8. Programming model for slave transmitting**



**Programming model in slave receiving mode**

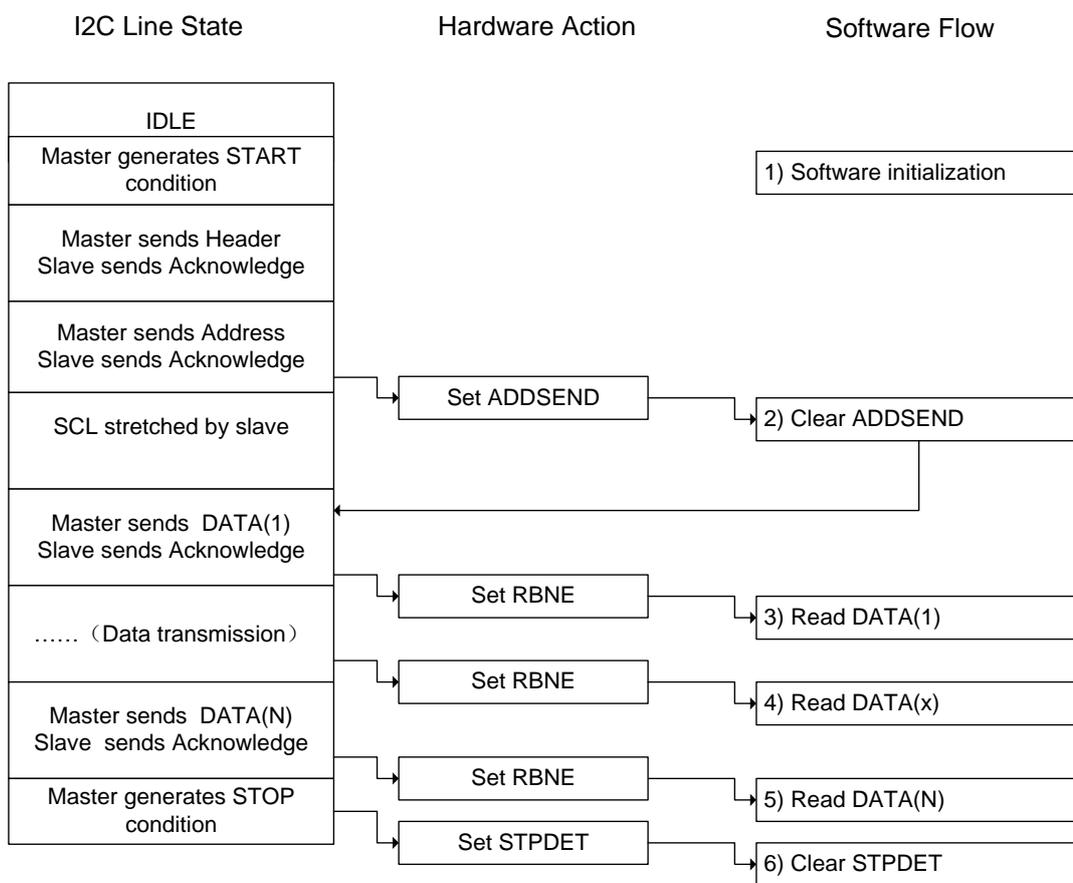
As is shown in the figure below, the following software procedure should be followed if users wish to make reception in slave receiver mode:

1. First of all, software should enable I2C peripheral clock as well as configure clock related registers in I2C\_CTL1 to make sure correct I2C timing. After enabled and configured, I2C operates in its default slave state and waits for START condition followed by address on I2C bus.
2. After receiving a START condition followed by a matched 7-bit or 10-bit address, the I2C hardware sets the ADDSEND bit in I2C status register, which should be monitored by software either by polling or interrupt. After that software should read I2C\_STAT0 and then I2C\_STAT1 to clear ADDSEND bit. The I2C begins to receive data to I2C bus as

soon as ADDSEND bit is cleared.

3. As soon as the first byte is received, RBNE is set by hardware. Software can now read the first byte from I2C\_DATA and RBNE is cleared as well.
4. Any time RBNE is set, software can read a byte from I2C\_DATA.
5. After last byte is received, RBNE is set. Software reads the last byte.
6. STPDET bit is set when I2C detects a STOP condition on I2C bus and software reads I2C\_STAT0 and then write I2C\_CTL0 to clear the STPDET bit.

**Figure 18-9. Programming model for slave receiving**



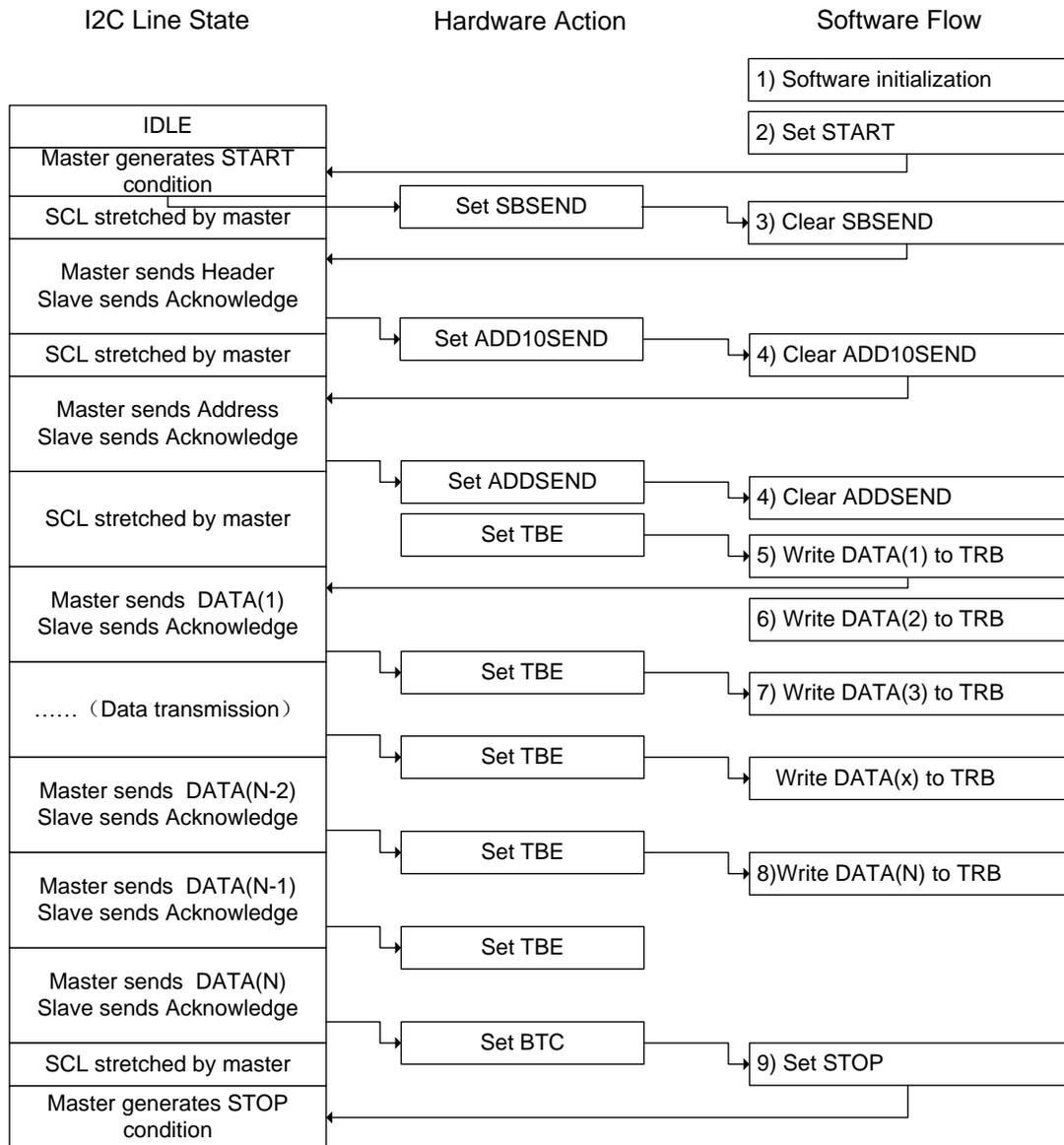
**Programming model in master transmitting mode**

As it shows in figure below, the following software procedure should be followed if users wish to make transaction in master transmitter mode:

1. First of all, software should enable I2C peripheral clock as well as configure clock related registers in I2C\_CTL1 to make sure correct I2C timing. After enabled and configured, I2C operates in its default slave state and waits for START condition followed by address on I2C bus.
2. Software set START bit requesting I2C to generate a START condition to I2C bus.

3. After sending a START condition, the I2C hardware sets the SBSEND bit in I2C status register and enters master mode. Now software should clear the SBSEND bit by reading I2C\_STAT0 and then writing a 7-bit address or header of a 10-bit address to I2C\_DATA. I2C begins to send address or header to I2C bus as soon as SBSEND bit is cleared. If the address sent is a header of 10-bit address, the hardware sets ADD10SEND bit after sending header and software should clear the ADD10SEND bit by reading I2C\_STAT0 and writing 10-bit lower address to I2C\_DATA.
4. After the 7-bit or 10-bit address is sent, the I2C hardware sets the ADDSEND bit and software should clear the ADDSEND bit by reading I2C\_STAT0 and then I2C\_STAT1.
5. Now I2C enters data transmission stage and hardware sets TBE bit because both the shift register and data register I2C\_DATA are empty. Software now write the first byte data to I2C\_DATA register, but the TBE is not cleared because the write byte in I2C\_DATA is moved to internal shift register immediately. The I2C begins to transmit data to I2C bus as soon as shift register is not empty.
6. During the first byte's transmission, software can write the second byte to I2C\_DATA, and this time TBE is cleared because neither I2C\_DATA nor shift register is empty.
7. Any time TBE is set, software can write a byte to I2C\_DATA as long as there are still data to be transmitted.
8. During the second last byte's transmission, software write the last data to I2C\_DATA to clear the TBE flag and doesn't care TBE anymore. So TBE will be asserted after the byte's transmission and not cleared until a STOP condition.
9. After sending the last byte, I2C master sets BTC bit because both shift register and I2C\_DATA are empty. Software should program a STOP request now, and the I2C clears both TBE and BTC flags after sending a STOP condition.

**Figure 18-10. Programming model for master transmitting**



**Programming model in master receiving mode**

In master receiving mode, a master is responsible for generating NACK for the last byte reception and then sending STOP condition on I2C bus. So, special attention should be paid to ensure the correct ending of data reception. Two solutions for master receiving are provided here for your application: Solution A and B. Solution A requires the software’s quick response to I2C events, while Solution B doesn’t.

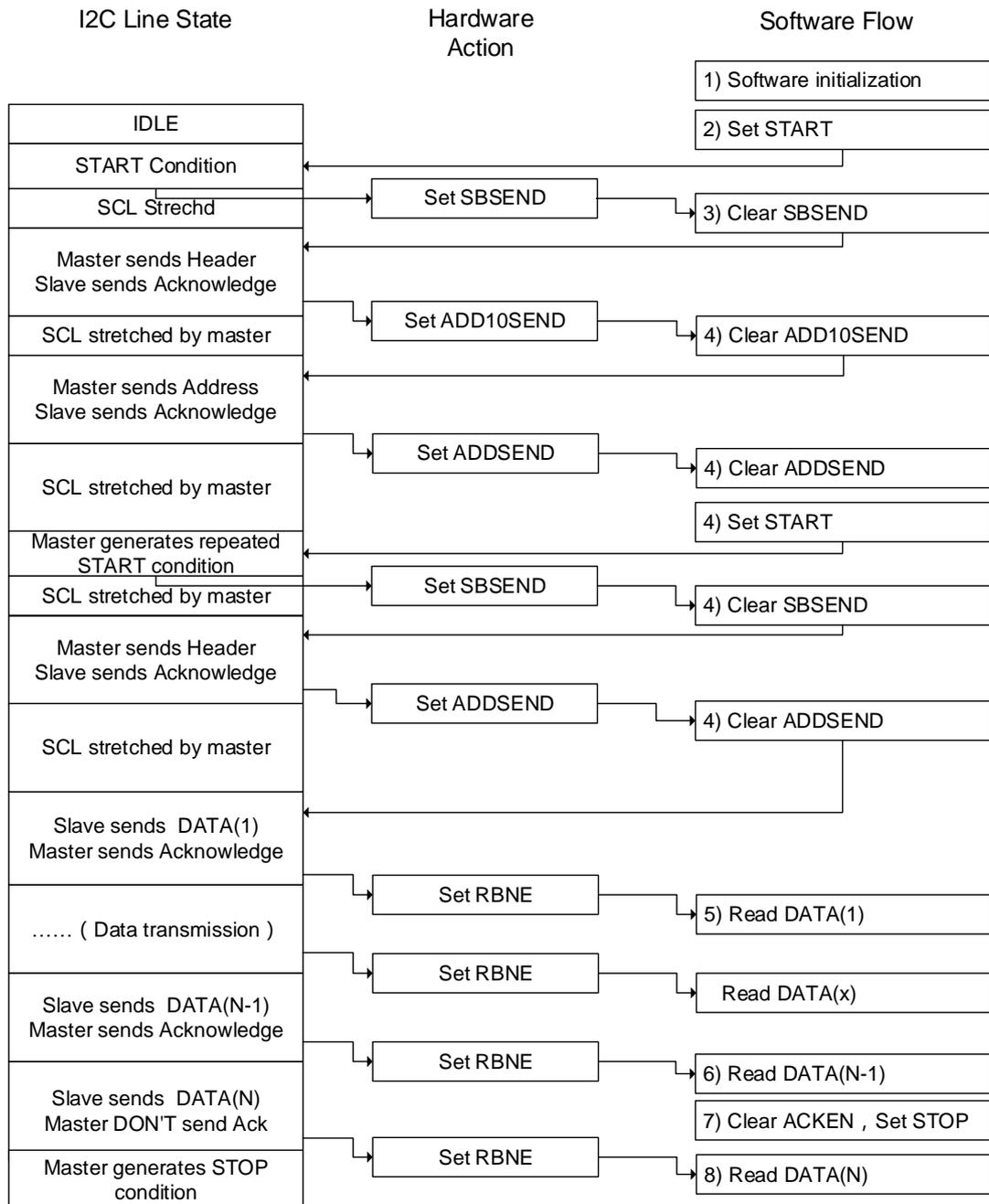
**Solution A**

1. First of all, software should enable I2C peripheral clock as well as configure clock related registers in I2C\_CTL1 to make sure correct I2C timing. After enabled and configured, I2C operates in its default slave state and waits for START condition followed by address on I2C bus.

2. Software set START bit requesting I2C to generate a START condition to I2C bus.
3. After sending a START condition, the I2C hardware sets the SBSEND bit in I2C status register and enters master mode. Now software should clear the SBSEND bit by reading I2C\_STAT0 and then writing a 7-bit address or header of a 10-bit address to I2C\_DATA. I2C begins to send address or header to I2C bus as soon as SBSEND bit is cleared. If the address sent is a header of 10-bit address, the hardware sets ADD10SEND bit after sending header and software should clear the ADD10SEND bit by reading I2C\_STAT0 and writing 10-bit lower address to I2C\_DATA.
4. After the 7-bit or 10-bit address is sent, the I2C hardware sets the ADDSEND bit and software should clear the ADDSEND bit by reading I2C\_STAT0 and then I2C\_STAT1. If the address is in 10-bit format, software should then set START bit again to generate a repeated START condition on I2C bus and SBSEND is set after the repeated START is sent out. Software should clear the SBSEND bit by reading I2C\_STAT0 and writing header to I2C\_DATA. Then the header is sent out to I2C bus, and ADDSEND is set again. Software should again clear ADDSEND by reading I2C\_STAT0 and then I2C\_STAT1.
5. As soon as the first byte is received, RBNE is set by hardware. Software now can read the first byte from I2C\_DATA and RBNE is cleared as well.
6. Any time RBNE is set, software can read a byte from I2C\_DATA.
7. After the second last byte is received, the software should clear ACKEN bit and set STOP bit. These actions should complete before the end of the last byte's receiving to ensure that NACK is sent for the last byte.
8. After last byte is received, RBNE is set. Software reads the last byte. I2C doesn't send ACK to the last byte and generate a STOP condition after the transmission of the last byte.

Above steps require byte number  $N > 1$ . If  $N = 1$ , Step 7 should be performed after Step 4 and completed before the end of the single byte's receiving.

**Figure 18-11. Programming model for master receiving using Solution A**



**Solution B**

1. First of all, software should enable I2C peripheral clock as well as configure clock related registers in I2C\_CTL1 to make sure correct I2C timing. After enabled and configured, I2C operates in its default slave state and waits for START condition followed by address on I2C bus.
2. Software set START bit requesting I2C to generate a START condition to I2C bus.
3. After sending a START condition, the I2C hardware sets the SBSEND bit in I2C status

register and enters master mode. Now software should clear the SBSEND bit by reading I2C\_STAT0 and then writing a 7-bit address or header of a 10-bit address to I2C\_DATA. I2C begins to send address or header to I2C bus as soon as SBSEND bit is cleared. If the address sent is a header of 10-bit address, the hardware sets ADD10SEND bit after sending header and software should clear the ADD10SEND bit by reading I2C\_STAT0 and writing 10-bit lower address to I2C\_DATA.

4. After the 7-bit or 10-bit address is sent, the I2C hardware sets the ADDSEND bit and software should clear the ADDSEND bit by reading I2C\_STAT0 and then I2C\_STAT1. If the address is in 10-bit format, software should then set START bit again to generate a repeated START condition on I2C bus and SBSEND is set after the repeated START is sent out. Software should clear the SBSEND bit by reading I2C\_STAT0 and writing header to I2C\_DATA. Then the header is sent out to I2C bus, and ADDSEND is set again. Software should again clear ADDSEND by reading I2C\_STAT0 and then I2C\_STAT1.
5. As soon as the first byte is received, RBNE is set by hardware. Software now can read the first byte from I2C\_DATA and RBNE is cleared as well.
6. Any time RBNE is set, software can read a byte from I2C\_DATA until the master receives N-3 bytes.

As shown in [Figure 18-12. Programming model for master receiving using Solution B](#), the N-2 byte is not read out by software, so after the N-1 byte is received, both BTC and RBNE are asserted. The bus is stretched by master to prevent the reception of the last byte. Then software should clear ACKEN bit.

7. Software reads out N-2 byte, clearing BTC. After this the N-1 byte is moved from shift register to I2C\_DATA and bus is released and begins to receive the last byte.
8. After last byte is received, both BTC and RBNE is set again. Software sets STOP bit and master sends out a STOP condition on bus.
9. Software reads the N-1 byte, clearing BTC. After this the last byte is moved from shift register to I2C\_DATA.
10. Software reads the last byte, clearing RBNE.

Above steps require that byte number  $N > 2$ .  $N=1$  or  $N=2$  are similar:

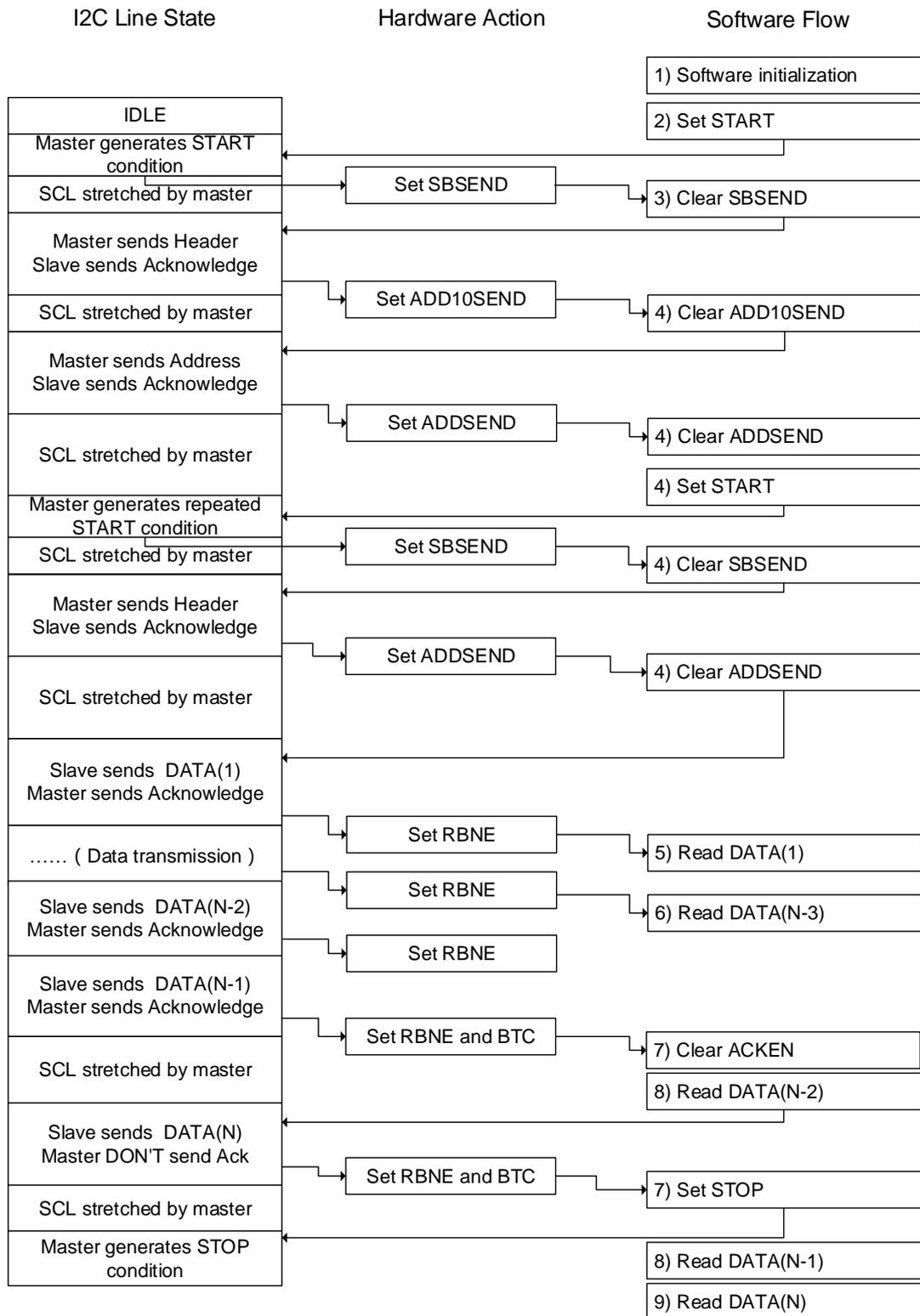
### **N=1**

In Step4, software should reset ACK bit before clearing ADDSEND bit and set STOP bit after clearing ADDSEND bit. Step 5 is the last step when  $N=1$ .

### **N=2**

In Step 2, software should set POAP bit before set START bit. In Step 4, software should reset ACKEN bit before clearing ADDSEND bit. In Step 5, software should wait until BTC is set and then set STOP bit and reads I2C\_DATA twice.

**Figure 18-12. Programming model for master receiving using Solution B**



### 18.3.8. SCL line stretching

The SCL line stretching function is designed to avoid overflow error in reception and underflow error in transmission. As is shown in Programming Model, when the TBE and BTC bit of a

transmitter is set, the transmitter stretches the SCL line low until the transfer buffer register is filled with the next transmit data. When the RBNE and BTC bit of a receiver is set, the receiver stretches the SCL line low until the data in the transfer buffer is read out.

When works in slave mode, the SCL line stretching function can be disabled by setting the DISSTRC bit in the I2C\_CTL0 register. If this bit is set, the software is required to be quick enough to serve the TBE, RBNE and BTC status, otherwise, overflow or underflow situation might occur.

### 18.3.9. Use DMA for data transfer

As is shown in Programming Model, each time TBE or RBNE is asserted, software should write or read a byte, this may cause CPU's high overload. The DMA controller can be used to process TBE and RBNE flag: each time TBE or RBNE is asserted, DMA controller does a read or write operation automatically.

The DMA request is enabled by the DMAON bit in the I2C\_CTL1 register. This bit should be set before clearing the ADDSEND status. If the SCL line stretching function is disabled for a slave device, the DMAON bit should be set before the ADDSEND event.

Refer to the specification of the DMA controller for the configuration method of a DMA stream. The DMA controller must be configured and enabled before I2C transfer. When the configured number of byte has been transferred, the DMA controller generates End of Transfer (EOT) interrupt.

When a master receives two or more bytes, the DMALST bit in the I2C\_CTL1 register should be set. The I2C master will not send nack after the last byte. The software can set the STOP bit to generate a stop condition in the ISR of the DMA EOT interrupt.

When a master receives only one byte, the ACKEN bit must be cleared before clearing the ADDSEND status. Software can set the STOP bit to generate a stop condition after clearing the ADDSEND status, or in the ISR of the DMA EOT interrupt.

### 18.3.10. Packet error checking

There is a CRC-8 calculator in I2C block to perform Packet Error Checking for I2C data. The polynomial of the CRC is  $x^8 + x^2 + x + 1$  which is compatible with the SMBus protocol. If enabled by setting PECEN bit, the PEC will calculate all the data transmitted through I2C including address. I2C is able to send out the PEC value after the last data byte or check the received PEC value with its calculated PEC using the PECTRANS bit. In DMA mode, the I2C will send or check PEC value automatically if PECEN bit is set.

### 18.3.11. SMBus support

The System Management Bus (abbreviated to SMBus or SMB) is a single-ended simple two-wire bus for the purpose of lightweight communication. Most commonly it is found in computer

motherboards for communication with power source for ON/OFF instructions. It is derived from I2C for communication with low-bandwidth devices on a motherboard, especially power related chips such as a laptop's rechargeable battery subsystem (see Smart Battery Data).

### **SMBus protocol**

Each message transaction on SMBus follows the format of one of the defined SMBus protocols. The SMBus protocols are a subset of the data transfer formats defined in the I2C specifications. I2C devices that can be accessed through one of the SMBus protocols are compatible with the SMBus specifications. I2C devices that do not adhere to these protocols cannot be accessed by standard methods as defined in the SMBus and Advanced Configuration and Power Management Interface (abbreviated to ACPI) specifications.

### **Address resolution protocol**

The SMBus uses I2C hardware and I2C hardware addressing, but adds second-level software for building special systems. Additionally, its specifications include an Address Resolution Protocol that can make dynamic address allocations. Dynamic reconfiguration of the hardware and software allow bus devices to be 'hot-plugged' and used immediately, without restarting the system. The devices are recognized automatically and assigned unique addresses. This advantage results in a plug-and-play user interface. In both those protocols there is a very useful distinction made between a System Host and all the other devices in the system that can have the names and functions of masters or slaves.

### **Time-out feature**

SMBus has a time-out feature which resets devices if a communication takes too long. This explains the minimum clock frequency of 10 kHz to prevent locking up the bus. I2C can be a 'DC' bus, meaning that a slave device stretches the master clock when performing some routine while the master is accessing it. This will notify to the master that the slave is busy but does not want to lose the communication. The slave device will allow continuation after its task is completed. There is no limit in the I2C bus protocol as to how long this delay can be, whereas for a SMBus system, it would be limited to 35ms. SMBus protocol just assumes that if something takes too long, then it means that there is a problem on the bus and that all devices must reset in order to clear this mode. Slave devices are not allowed to hold the clock low too long.

### **Packet error checking**

SMBus 2.0 and 1.1 allow Packet Error Checking (PEC). In that mode, a PEC (packet error code) byte is appended at the end of each transaction. The byte is calculated as CRC-8 checksum, calculated over the entire message including the address and read/write bit. The polynomial used is  $x^8+x^2+x+1$  (the CRC-8-ATM HEC algorithm, initialized to zero).

## SMBus alert

The SMBus has an extra optional shared interrupt signal called SMBALERT# which can be used by slaves to tell the host to ask its slaves about events of interest. SMBus also defines a less common "Host Notify Protocol", providing similar notifications but passing more data and building on the I2C multi-master mode.

## SMBus programming flow

The programming flow for SMBus is similar to normal I2C. In order to use SMBus mode, the application should configure several SMBus specific registers, response to some SMBus specific flags and implement the upper protocols described in SMBus specification.

1. Before communication, SMBEN bit in I2C\_CTL0 should be set and SMBSEL and ARPEN bits should be configured to desired value.
2. In order to support address resolution protocol (ARP) (ARPEN=1), the software should response to HSTSMB flag in SMBus Host Mode (SMBTYPE =1) or DEFSMB flag in SMBus Device Mode, and implement the function of ARP protocol.
3. In order to support SMBus Alert Mode, the software should response to SMBALT flag and implement the related function.

### 18.3.12. Status, errors and interrupts

There are several status and error flags in I2C, and interrupt may be asserted from these flags by setting some register bits (refer to I2C register for detail).

**Table 18-2. Event status flags**

Event Flag Name	Description
SBSEND	START condition sent (master)
ADDSEND	Address sent or received
ADD10SEND	Header of 10-bit address sent
STPDET	STOP condition detected
BTC	Byte transmission completed
TBE	I2C_DATA is empty when transmitting
RBNE	I2C_DATA is not empty when receiving

**Table 18-3. I2C error flags**

I2C Error Name	Description
BERR	Bus error
LOSTARB	Arbitration lost
OUERR	Over-run or under-run when SCL stretch is disabled.
AERR	No acknowledge received
PECERR	CRC value doesn't match

<b>I2C Error Name</b>	<b>Description</b>
SMBTO	Bus timeout in SMBus mode
SMBALT	SMBus Alert

## 18.4. Register definition

### 18.4.1. Control register 0 (I2C\_CTL0)

Address offset: 0x00

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SRESET	Reserved	SALT	PECTRANS	POAP	ACKEN	STOP	START	DISSTRC	GCEN	PECEN	ARPEN	SMBSEL	Reserved	SMBEN	I2CEN
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw

Bits	Fields	Descriptions
15	SRESET	Software reset I2C, software should wait until the I2C lines are released to reset the I2C 0: I2C is not under reset 1: I2C is under reset
14	Reserved	Must be kept the reset value
13	SALT	SMBus Alert. Issue alert through SMBA pin. Software can set and clear this bit and hardware can clear this bit. 0: Don't issue alert through SMBA pin 1: Issue alert through SMBA pin
12	PECTRANS	PEC Transfer Software set and clear this bit while hardware clears this bit when PEC is transferred or START/STOP condition detected or I2CEN=0 0: Don't transfer PEC value 1: Transfer PEC
11	POAP	Position of ACK and PEC when receiving This bit is set and cleared by software and cleared by hardware when I2CEN=0 0: ACKEN bit specifies whether to send ACK or NACK for the current byte that is being received. PECTRANS bit indicates that the current receiving byte is a PEC byte 1: ACKEN bit specifies whether to send ACK or NACK for the next byte that is to be received, PECTRANS bit indicates the next byte that is to be received is a PEC byte
10	ACKEN	Whether or not to send an ACK This bit is set and cleared by software and cleared by hardware when I2CEN=0 0: ACK will not be sent 1: ACK will be sent

9	STOP	<p>Generate a STOP condition on I2C bus</p> <p>This bit is set and cleared by software and set by hardware when SMBUs timeout and cleared by hardware when STOP condition detected.</p> <p>0: STOP will not be sent</p> <p>1: STOP will be sent</p>
8	START	<p>Generate a START condition on I2C bus</p> <p>This bit is set and cleared by software and and cleared by hardware when START condition detected or I2CEN=0</p> <p>0: START will not be sent</p> <p>1: START will be sent</p>
7	DISSTRC	<p>Whether to stretch SCL low when data is not ready in slave mode.</p> <p>This bit is set and cleared by software.</p> <p>0: SCL Stretching is enabled</p> <p>1: SCL Stretching is disabled</p>
6	GCEN	<p>Whether or not to response to a General Call (0x00)</p> <p>0: Slave won't response to a General Call</p> <p>1: Slave will response to a General Call</p>
5	PECEN	<p>PEC Calculation Switch</p> <p>0: PEC Calculation off</p> <p>1: PEC Calculation on</p>
4	ARPEN	<p>ARP protocol in SMBus switch</p> <p>0: ARP is disabled</p> <p>1: ARP is enabled</p>
3	SMBSEL	<p>SMBus Type Selection</p> <p>0: Device</p> <p>1: Host</p>
2	Reserved	<p>Must keep the reset value</p>
1	SMBEN	<p>SMBus/I2C mode switch</p> <p>0: I2C mode</p> <p>1: SMBus mode</p>
0	I2CEN	<p>I2C peripheral enable</p> <p>0: I2C is disabled</p> <p>1: I2C is enabled</p>

#### 18.4.2. Control register 1 (I2C\_CTL1)

Address offset: 0x04

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			DMALST	DMAON	BUFIE	EVIE	ERRIE	Reserved			I2CCLK[5:0]				
			rw	rw	rw	rw	rw				rw				

Bits	Fields	Descriptions
15:13	Reserved	Must be kept the reset value
12	DMALST	Flag indicating DMA last transfer 0: Next DMA EOT is not the last transfer 1: Next DMA EOT is the last transfer
11	DMAON	DMA mode switch 0: DMA mode disabled 1: DMA mode enabled
10	BUFIE	Buffer interrupt enable 0: No interrupt asserted when TBE = 1 or RBNE = 1 1: Interrupt asserted when TBE = 1 or RBNE = 1 if EVIE=1
9	EVIE	Event interrupt enable 0: Event interrupt disabled 1: Event interrupt enabled, means that interrupt will be generated when SBSSEND, ADDSEND, ADD10SEND, STPDET or BTC flag asserted or TBE=1 or RBNE=1 if BUFIE=1.
8	ERRIE	Error interrupt enable 0: Error interrupt disabled 1: Error interrupt enabled, means that interrupt will be generated when BERR, LOSTARB, AERR, OUERR, PECERR, SMBTO or SMBALT flag asserted.
7:6	Reserved	Must be kept the reset value
5:0	I2CCLK[5:0]	I2C Peripheral clock frequency I2CCLK[5:0] should be the frequency of input APB clock in MHz which is at least 2. 0h - 1h: Not allowed 2h - 36h: 2 MHz~36MHz 37h - 63h: Not allowed due to the limitation of APB clock

### 18.4.3. Slave address register 0 (I2C\_SADDR0)

Address offset: 0x08

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)

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

ADDFOR MAT	Reserved	ADDRESS[9:8]	ADDRESS[7:1]	ADDRESS S0
rw		rw	rw	rw

Bits	Fields	Descriptions
15	ADDFORMAT	Address mode for the I2C slave 0: 7-bit Address 1: 10-bit Address
14:10	Reserved	Must be kept the reset value
9:8	ADDRESS[9:8]	Highest two bits of a 10-bit address
7:1	ADDRESS[7:1]	7-bit address or bits 7:1 of a 10-bit address
0	ADDRESS0	Bit 0 of a 10-bit address

#### 18.4.4. Slave address register 1 (I2C\_SADDR1)

Address offset: 0x0C

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								ADDRESS2[7:1]							DUADEN
								rw							rw

Bits	Fields	Descriptions
15:8	Reserved	Must be kept the reset value
7:1	ADDRESS2[7:1]	Second I2C address for the slave in Dual-Address mode
0	DUADEN	Dual-Address mode switch 0: Dual-Address mode disabled 1: Dual-Address mode enabled

#### 18.4.5. Transfer buffer register (I2C\_DATA)

Address offset: 0x10

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)

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

Bits	Fields	Descriptions
15:8	Reserved	Must be kept the reset value
7:0	TRB[7:0]	Transmission or reception data buffer

#### 18.4.6. Transfer status register 0 (I2C\_STAT0)

Address offset: 0x14

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMBALT	SMBTO	Reserved	PECERR	OUERR	AERR	LOSTAR B	BERR	TBE	RBNE	Reserved	STPDET	ADD10S END	BTC	ADDSEN D	SBSEND
rc_w0	rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	r	r		r	r	r	r	r

Bits	Fields	Descriptions
15	SMBALT	SMBus Alert status This bit is set by hardware and cleared by writing 0. 0: SMBA pin not pulled down (device mode) or no Alert detected (host mode) 1: SMBA pin pulled down (device mode) or Alert detected (host mode)
14	SMBTO	Timeout signal in SMBus mode This bit is set by hardware and cleared by writing 0. 0: No timeout error 1: Timeout event occurs (SCL is low for 25 ms)
13	Reserved	Must keep the reset value
12	PECERR	PEC error when receiving data This bit is set by hardware and cleared by writing 0. 0: Received PEC and calculated PEC match 1: Received PEC and calculated PEC don't match, I2C will send NACK careless of ACKEN bit.
11	OUERR	Over-run or under-run situation occurs in slave mode, when SCL stretching is disabled. In slave receiving mode, if the last byte in I2C_DATA is not read out while the following byte is already received, over-run occurs. In slave transmitting mode, if the current byte is already sent out, while the I2C_DATA is still empty, under-run occurs. This bit is set by hardware and cleared by writing 0. 0: No over-run or under-run occurs 1: Over-run or under-run occurs
10	AERR	Acknowledge Error This bit is set by hardware and cleared by writing 0.

		0: No Acknowledge Error 1: Acknowledge Error
9	LOSTARB	Arbitration Lost in master mode This bit is set by hardware and cleared by writing 0. 0: No Arbitration Lost 1: Arbitration Lost occurs and the I2C block changes back to slave mode.
8	BERR	A bus error occurs indication an unexpected START or STOP condition on I2C bus This bit is set by hardware and cleared by writing 0. 0: No bus error 1: A bus error detected
7	TBE	I2C_DATA is Empty during transmitting This bit is set by hardware after it moves a byte from I2C_DATA to shift register and cleared by writing a byte to I2C_DATA. If both the shift register and I2C_DATA are empty, writing I2C_DATA won't clear TBE (refer to Programming Model for detail). 0: I2C_DATA is not empty 1: I2C_DATA is empty, software can write
6	RBNE	I2C_DATA is not Empty during receiving This bit is set by hardware after it moves a byte from shift register to I2C_DATA and cleared by reading it. If both BTC and RBNE are asserted, reading I2C_DATA won't clear RBNE because the shift register's byte is moved to I2C_DATA immediately. 0: I2C_DATA is empty 1: I2C_DATA is not empty, software can read
5	Reserved	Must be kept the reset value
4	STPDET	STOP condition detected in slave mode This bit is set by hardware and cleared by reading I2C_STAT0 and then writing I2C_CTL0 0: STOP condition not detected in slave mode 1: STOP condition detected in slave mode
3	ADD10SEND	Header of 10-bit address is sent in master mode This bit is set by hardware and cleared by reading I2C_STAT0 and writing I2C_DATA. 0: No header of 10-bit address sent in master mode 1: Header of 10-bit address is sent in master mode
2	BTC	Byte transmission completed. If a byte is already received in shift register but I2C_DATA is still full in receiving mode or a byte is already sent out from shift register but I2C_DATA is still empty in transmitting mode, the BTC flag is asserted if SCL stretching enabled.

		This bit is set by hardware.
		This bit can be cleared by 3 ways as follow:
		1. Reading I2C_STAT0 followed by reading or writing
		2. Hardware clearing: sending the STOP condition or START condition
		3. Bit 0 (I2CEN bit) of the I2C_CTL0 is reset.
		0: BTC not asserted
		1: BTC asserted
1	ADDSEND	Address is sent in master mode or received and matches in slave mode. This bit is set by hardware and cleared by reading I2C_STAT0 and reading I2C_STAT1. 0: No address sent or received 1: Address sent out in master mode or a matched address is received in slave mode
0	SBSEND	START condition sent out in master mode This bit is set by hardware and cleared by reading I2C_STAT0 and writing I2C_DATA 0: No START condition sent 1: START condition sent

## 18.4.7. Transfer status register 1 (I2C\_STAT1)

Address offset: 0x18

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ECV[7:0]								DUMODF	HSTSMB	DEFSMB	RXGC	Reserved	TRS	I2CBSY	MASTER
r								r	r	r	r		r	r	r

Bits	Fields	Descriptions
15:8	ECV[7:0]	Packet Error Checking Value that calculated by hardware when PEC is enabled.
7	DUMODF	Dual Flag in slave mode indicating which address is matched in Dual-Address mode This bit is cleared by hardware after a STOP or a START condition or I2CEN=0 0: SADDR0 address matches 1: SADDR1 address matches
6	HSTSMB	SMBus Host Header detected in slave mode This bit is cleared by hardware after a STOP or a START condition or I2CEN=0 0: No SMBus Host Header detected 1: SMBus Host Header detected

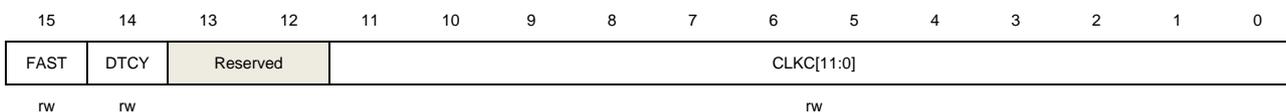
5	DEFSMB	Default address of SMBus Device This bit is cleared by hardware after a STOP or a START condition or I2CEN=0. 0: The default address has not been received 1: The default address has been received for SMBus Device
4	RXGC	General call address (00h) received. This bit is cleared by hardware after a STOP or a START condition or I2CEN=0. 0: No general call address (00h) received 1: General call address (00h) received
3	Reserved	Must be kept the reset value
2	TRS	Whether the I2C is a transmitter or a receiver This bit is cleared by hardware after a STOP or a START condition or I2CEN=0 or LOSTARB=1. 0: Receiver 1: Transmitter
1	I2CBSY	Busy flag This bit is cleared by hardware after a STOP condition 0: No I2C communication. 1: I2C communication active.
0	MASTER	A flag indicating whether I2C block is in master or slave mode. This bit is cleared by hardware after a STOP or a START condition or I2CEN=0 or LOSTARB=1. 0: Slave mode 1: Master mode

### 18.4.8. Clock configure register (I2C\_CKCFG)

Address offset: 0x1C

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



Bits	Fields	Descriptions
15	FAST	I2C speed selection in master mode 0: Standard speed 1: Fast speed
14	DTCY	Duty cycle in fast mode 0: $T_{low}/T_{high} = 2$

1:  $T_{low}/T_{high} = 16/9$

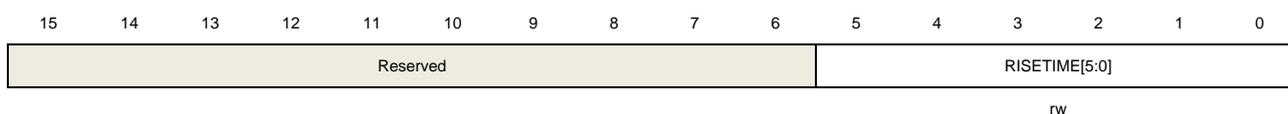
13:12	Reserved	Must be kept the reset value
11:0	CLKC[11:0]	<p>I2C Clock control in master mode</p> <p>In standard speed mode: <math>T_{high} = T_{low} = CLKC * T_{PCLK1}</math></p> <p>In fast speed mode if DTCY=0:</p> $T_{high} = CLKC * T_{PCLK1}$ , $T_{low} = 2 * CLKC * T_{PCLK1}$ <p>In fast speed mode if DTCY=1:</p> $T_{high} = 9 * CLKC * T_{PCLK1}$ , $T_{low} = 16 * CLKC * T_{PCLK1}$

### 18.4.9. Rise time register (I2C\_RT)

Address offset: 0x20

Reset value: 0x0002

This register can be accessed by half-word (16-bit) or word (32-bit)



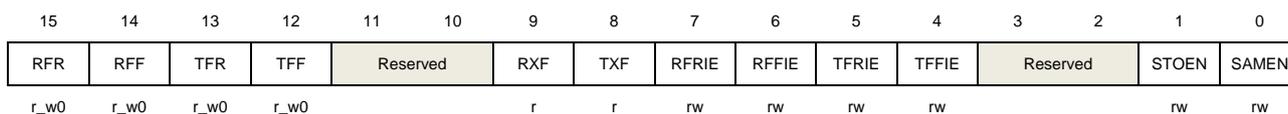
Bits	Fields	Descriptions
15:6	Reserved	Must be kept the reset value
5:0	RISETIME[5:0]	<p>Maximum rise time in master mode</p> <p>The RISETIME value should be the maximum SCL rise time incremented by 1.</p>

### 18.4.10. SAM control and status register (I2C\_SAMCS) of GD32F170xx and GD32F190xx devices

Address offset: 0x80

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



Bits	Fields	Descriptions
15	RFR	Rxframe rise flag, cleared by software write 0
14	RFF	Rxframe fall flag, cleared by software write 0

13	TFR	Txframe rise flag, cleared by software write 0
12	TFF	Txframe fall flag, cleared by software write 0
11:10	Reserved	Must be kept the reset value
9	RXF	Level of Rxframe signal
8	TXF	Level of Txframe signal
7	RFRIE	Rxframe rise interrupt enable 0: Disable 1: Enable
6	RFFIE	Rxframe fall interrupt enable 0: Disable 1: Enable
5	TFRIE	Txframe rise interrupt enable 0: Disable 1: Enable
4	TFFIE	Txframe fall interrupt enable 0: Disable 1: Enable
3:2	Reserved	Must be kept the reset value
1	STOEN	SAM_V interface timeout detect enable 0: Disable 1: Enable
0	SAMEN	SAM_V interface enable 0: Disable 1: Enable

## 19. Serial peripheral interface/Inter-IC sound (SPI/I2S)

### 19.1. Overview

The SPI/I2S module can communicate with external devices using the SPI protocol or the I2S audio protocol.

The Serial Peripheral Interface (SPI) provides a SPI protocol of data transmission and reception function in master or slave mode. Both full-duplex and simplex communication modes are supported, with hardware CRC calculation and checking.

For GD32F130xx and GD32F150xx devices, the SPI Interface supports single wire configuration in both master and slave mode. But for GD32F170xx and GD32F190xx devices, the SPI Interface also supports quad wire configuration in master mode except supporting single wire configuration.

The inter-IC sound (I2S) supports four audio standards: I2S Phillips standard, MSB justified standard, LSB justified standard, and PCM standard. I2S works at either master or slave mode for transmission and reception.

### 19.2. Characteristics

#### 19.2.1. SPI characteristics

- Master or slave operation with full-duplex or simplex mode
- Separate transmit and receive buffer, 16 bits wide
- Data frame size can be 8 or 16 bits
- Bit order can be LSB first or MSB first
- Software and hardware NSS management
- Hardware CRC calculation, transmission and checking
- Transmission and reception using DMA

**For GD32F170xx and GD32F190xx devices, additional features are shown below.**

- Quad-SPI configuration available in master mode (only in SPI1)

#### 19.2.2. I2S characteristics

- Master or slave operation with transmission or reception mode
- Four I2S standards supported: Phillips, MSB justified, LSB justified and PCM standard

- Data length can be 16 bits, 24 bits or 32 bits
- Channel length can be 16 bits or 32 bits
- Transmission and reception using a 16 bits wide buffer
- Audio sample frequency can be 8 kHz to 192 kHz using I2S clock divider
- Programmable idle state clock polarity
- Master clock (MCK) can be output
- Transmission and reception using DMA

### 19.3. SPI block diagram

Figure 19-1. Block diagram of SPI for GD32F130xx and GD32F150xx devices

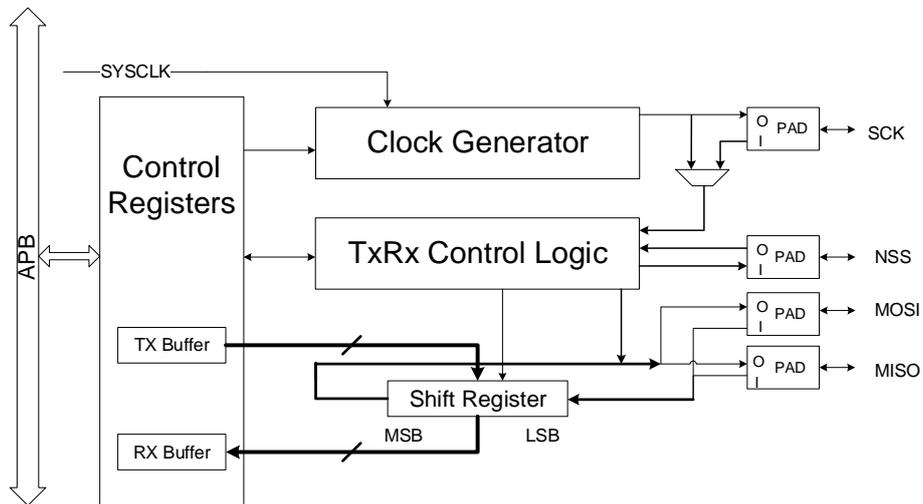
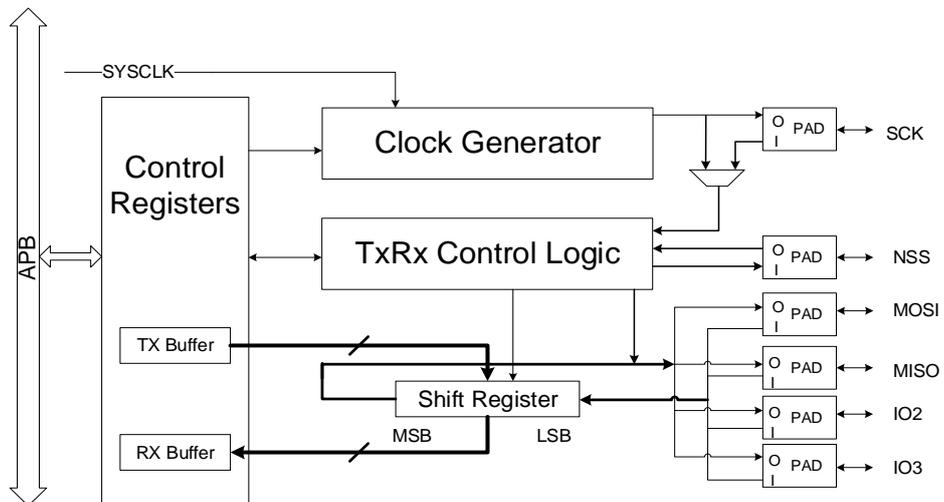


Figure 19-2. Block diagram of SPI for GD32F170xx and GD32F190xx devices



## 19.4. SPI signal description

### 19.4.1. Normal configuration

Table 19-1. SPI signal description

Pin Name	Direction	Description
SCK	I / O	Master: SPI Clock Output Slave: SPI Clock Input
MISO	I / O	Master: Data reception line Slave: Data transmission line Master with Bidirectional mode: Not used Slave with Bidirectional mode: Data transmission and reception Line.
MOSI	I / O	Master: Data transmission line Slave: Data reception line Master with Bidirectional mode: Data transmission and reception Line. Slave with Bidirectional mode: Not used
NSS	I / O	Software NSS Mode: Not Used Master in Hardware NSS Mode: NSS output (NSSDRV=1) for single master or (NSSDRV=0) for multi-master application. Slave in Hardware NSS Mode: NSS input, as a chip select signal for slave.

### 19.4.2. Quad-SPI configuration for GD32F170xx and GD32F190xx devices

SPI is in single wire mode by default and enters into Quad-SPI mode after QMOD bit in SPI\_QCTL register is set (only available in SPI1). Quad-SPI mode can only work at master mode.

Software is able to drive IO2 and IO3 pins high in normal Non-Quad-SPI mode by using IO23\_DRV bit in SPI\_QCTL register.

The SPI is connected to external devices through 6 pins in Quad-SPI mode:

Table 19-2. Quad-SPI signal description

Pin Name	Direction	Description
SCK	O	SPI Clock Output
MOSI	I / O	Transmission or Reception Data 0 line
MISO	I / O	Transmission or Reception Data 1 line
IO2	I / O	Transmission or Reception Data 2 line
IO3	I / O	Transmission or Reception Data 3 line

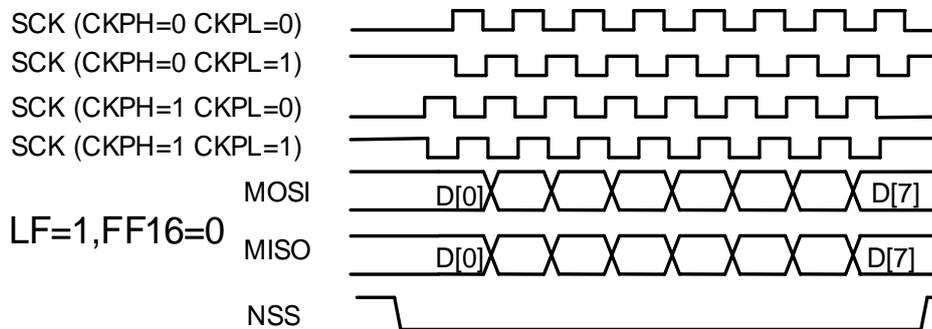
NSS	O	NSS output
-----	---	------------

## 19.5. SPI function overview

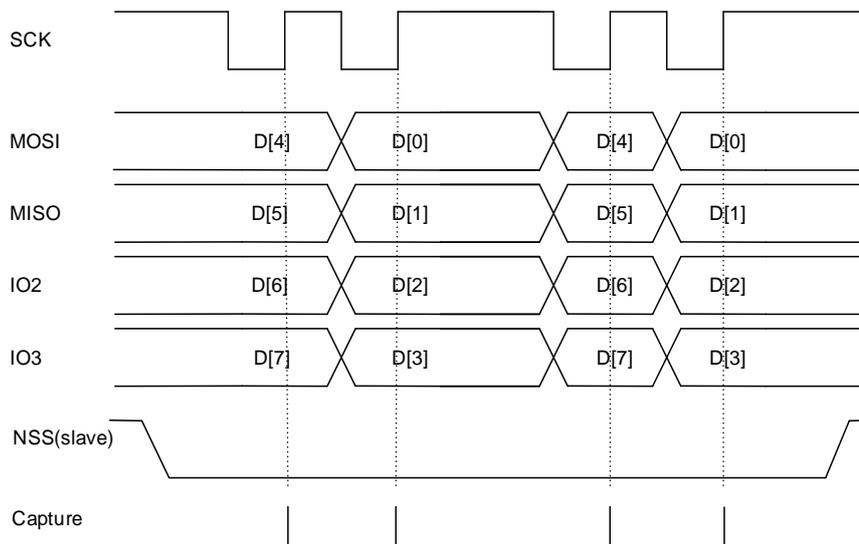
### 19.5.1. SPI clock timing and data format

CKPL and CKPH bits in SPI\_CTL0 register decide the timing of SPI clock and data signal. The CKPL bit decides the SCK level when idle and CKPH bit decides either first or second clock edge is a valid sampling edge.

**Figure 19-3. SPI timing diagram in normal mode**



**Figure 19-4. SPI timing diagram in Quad-SPI mode (CKPL=1, CKPH=1, LF=0) for GD32F170xx and GD32F190xx devices**



In normal mode, the length of data is configured by the FF16 bit in the SPI\_CTL0 register. Data length is 16 bits if FF16=1, otherwise is 8 bits. The data frame length is fixed to 8 bits in Quad-SPI mode.

Data order is configured by LF bit in SPI\_CTL0 register, and SPI will first send the LSB if

LF=1, or the MSB if LF=0.

## 19.5.2. NSS function

### Slave Mode

When slave mode is configured (MSTMOD=0), SPI gets NSS level from NSS pin in hardware NSS mode (SWNSSSEN = 0) or from SWNSS bit in software NSS mode (SWNSSSEN = 1) and transmits/receives data only when NSS level is low. In software NSS mode, NSS pin is not used.

### Master mode

In master mode (MSTMOD=1) if the application uses multi-master connection, NSS can be configured to hardware input mode (SWNSSSEN=0, NSSDRV=0) or software mode (SWNSSSEN=1). Then, once the NSS pin (in hardware NSS mode) or the SWNSS bit (in software NSS mode) goes low, the SPI automatically enters to slave mode and triggers a master fault flag CONFERR.

If the application wants to use NSS line to control the SPI slave, NSS should be configured to hardware output mode (SWNSSSEN=0, NSSDRV=1). NSS stays high after SPI is enabled and goes low when transmission or reception process begins.

The application may also use a general purpose IO as NSS pin to realize more flexible NSS.

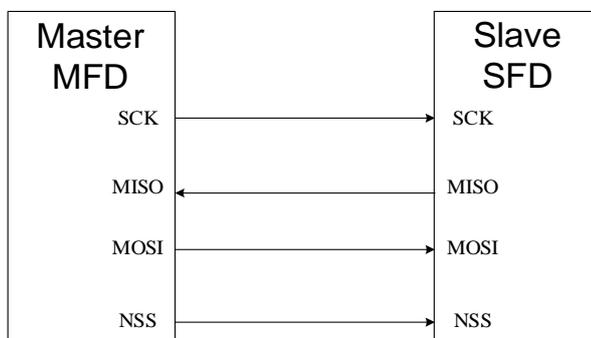
## 19.5.3. SPI operation modes

**Table 19-3. SPI operation modes**

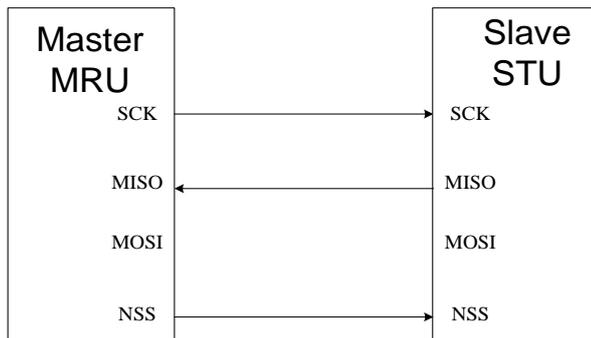
Mode	Description	Register Configuration	Data Pin Usage
MFD	Master Full-Duplex	MSTMOD = 1 RO = 0 BDEN = 0 BDOEN: Don't care	MOSI: Transmission MISO: Reception
MTU	Master Transmission with unidirectional connection	MSTMOD = 1 RO = 0 BDEN = 0 BDOEN: Don't care	MOSI: Transmission MISO: Not used
MRU	Master Reception with unidirectional connection	MSTMOD = 1 RO = 1 BDEN = 0 BDOEN: Don't care	MOSI: Not used MISO: Reception
MTB	Master Transmission with bidirectional connection	MSTMOD = 1 RO = 0	MOSI: Transmission MISO: Not used

Mode	Description	Register Configuration	Data Pin Usage
		BDEN = 1 BDOEN = 1	
MRB	Master Reception with bidirectional connection	MSTMOD = 1 RO = 0 BDEN = 1 BDOEN = 0	MOSI: Reception MISO: Not used
SFD	Slave Full-Duplex	MSTMOD = 0 RO = 0 BDEN = 0 BDOEN: Don't care	MOSI: Reception MISO: Transmission
STU	Slave Transmission with unidirectional connection	MSTMOD = 0 RO = 0 BDEN = 0 BDOEN: Don't care	MOSI: Not used MISO: Transmission
SRU	Slave Reception with unidirectional connection	MSTMOD = 0 RO = 1 BDEN = 0 BDOEN: Don't care	MOSI: Reception MISO: Not used
STB	Slave Transmission with bidirectional connection	MSTMOD = 0 RO = 0 BDEN = 1 BDOEN = 1	MOSI: Not used MISO: Transmission
SRB	Slave Reception with bidirectional connection	MSTMOD = 0 RO = 0 BDEN = 1 BDOEN = 0	MOSI: Not used MISO: Reception

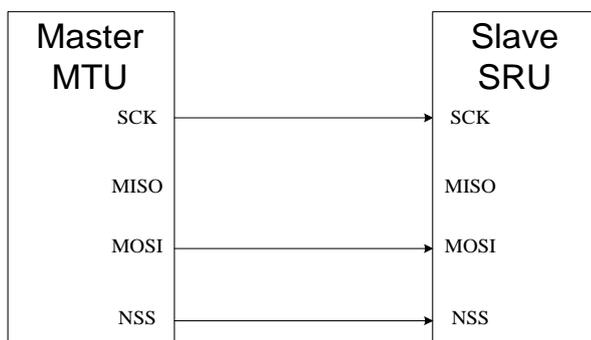
**Figure 19-5. A typical Full-duplex connection**



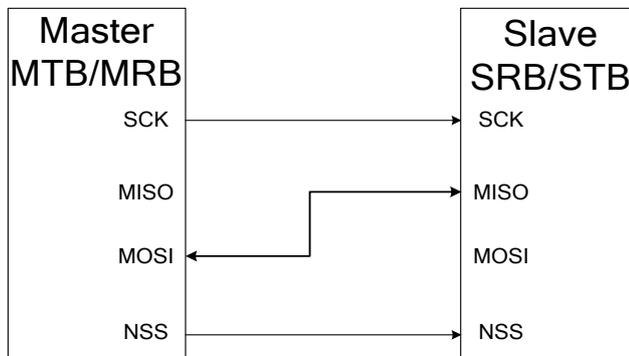
**Figure 19-6. A typical simplex connection (Master: Receive, Slave: Transmit)**



**Figure 19-7. A typical simplex connection (Master: Transmit only, Slave: Receive)**



**Figure 19-8. A typical bidirectional connection**



### SPI initialization sequence

Before transmitting or receiving data, application should follow the SPI initialization sequence described below:

1. If master mode is used, program the PSC [2:0] bits in SPI\_CTL0 register to generate SCK with desired baud rate, otherwise, ignore this step.
2. Program data format (FF16 bit in the SPI\_CTL0 register).
3. Program the clock timing register (CKPL and CKPH bits in the SPI\_CTL0 register).
4. Program the frame format (LF bit in the SPI\_CTL0 register).
5. Program the NSS mode (SWNSSSEN and NSSDRV bits in the SPI\_CTL0 register)

according to the application's demand as described above in [NSS function](#) section.

6. Configure MSTMOD, RO, BDEN and BDOEN depending on the operation modes described above.
7. For GD32F170xx and GD32F190xx devices, If Quad-SPI mode is used, set the QMOD bit in SPI\_QCTL register. Ignore this step if Quad-SPI mode is not used.
8. Enable the SPI (set the SPIEN bit).

### SPI basic transmission and reception sequence

#### Transmission sequence

After the initialization sequence, the SPI is enabled and stays at idle state. In master mode, the transmission starts when the application writes a data into the transmit buffer. In slave mode the transmission starts when SCK clock signal begins to toggle at SCK pin and NSS level is low, so application should ensure that data is already written into transmit buffer before the transmission starts in slave mode.

When SPI begins to send a data frame, it first loads this data frame from the data buffer to the shift register and then begins to transmit the loaded data frame, TBE (transmit buffer empty) flag is set after the first bit of this frame is transmitted. After TBE flag is set, which means the transmit buffer is empty, the application should write SPI\_DATA register again if it has more data to transmit.

In master mode, software should write the next data into SPI\_DATA register before the transmission of current data frame is completed if it desires to generate continuous transmission.

#### Reception sequence

The incoming data will be moved from shift register to the receive buffer after the last valid sample clock and also, RBNE (receive buffer not empty) will be set. The application should read SPI\_DATA register to get the received data and this will clear the RBNE flag automatically. In MRU and MRB modes, hardware continuously sends clock signal to receive the next data frame, while in full-duplex master mode (MFD), hardware only receives the next data frame when the transmit buffer is not empty.

#### SPI operation sequence in different modes (Not Quad-SPI mode)

In full-duplex mode, either MFD or SFD, application should monitor the RBNE and TBE flags and follow the sequences described above.

The transmission mode (MTU, MTB, STU or STB) is similar to full-duplex mode, except that application should ignore the RBNE and OVRE flags and only perform transmission sequence described above.

In master reception mode (MRU or MRB), the behavior is different from full-duplex mode or

transmission mode. In MRU or MRB mode, the SPI continuously generates SCK just after SPI is enabled, until the SPI is disabled. So the application should ignore the TBE flag and read out reception buffer in time after the RBNE flag is set, otherwise a data overrun fault will occur.

The slave reception mode (SRU or SRB) is similar to full-duplex mode, except that application should ignore the TBE flag and only perform reception sequence described above.

### **Quad-SPI mode operation sequence for GD32F170xx and GD32F190xx devices**

The Quad-SPI mode is designed to control quad SPI flash.

In order to enter Quad-SPI mode, the software should first verify that the TBE bit is set and TRANS bit is cleared, then set QMOD bit in SPI\_QCTL register. In Quad-SPI mode, BDEN, BDOEN, CRCEN, CRCNT, FF16, RO and LF in SPI\_CTL0 register should be kept cleared and MSTMOD should be set to ensure that SPI is in master mode. SPIEN, PSC, CKPL and CKPH should be configured as desired.

There are 2 operation modes in Quad-SPI mode: quad write and quad read, decided by QRD bit in SPI\_QCTL register.

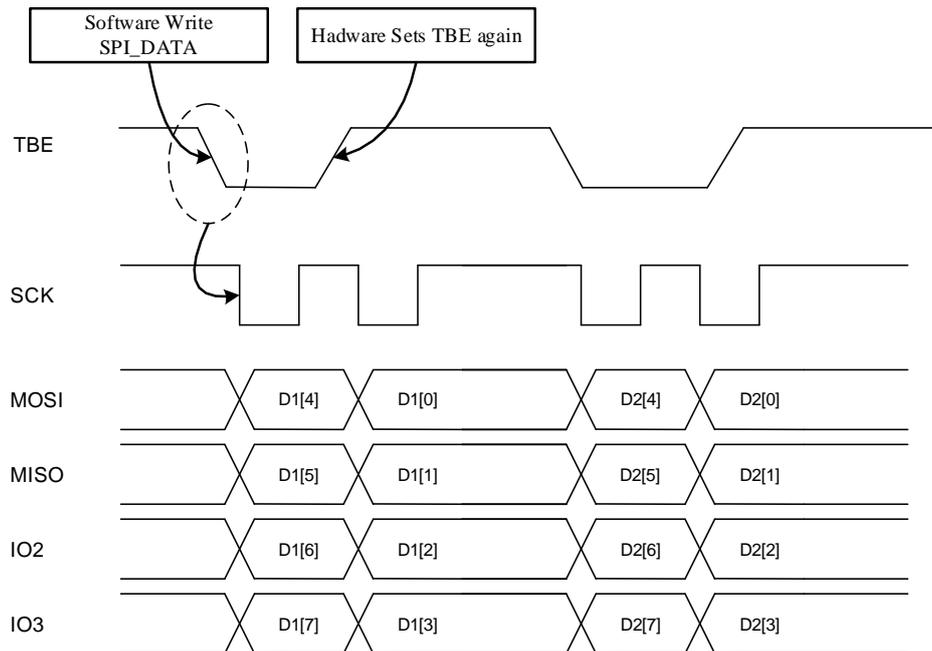
#### **Quad write operation**

SPI works in quad write mode when QMOD is set and QRD is cleared in SPI\_QCTL register. In this mode, MOSI, MISO, IO2 and IO3 are all used as output pins. SPI begins to generate clock on SCK line and transmit data on MOSI, MISO, IO2 and IO3 as soon as data is written into SPI\_DATA (TBE is cleared) and SPIEN is set. Once SPI starts transmission, it always checks TBE status at the end of a frame and stops when condition is not met.

The operation flow for transmitting in quad mode:

1. Configure clock prescaler, clock polarity, phase, etc. in SPI\_CTL0 and SPI\_CTL1 based on your application requirements.
2. Set QMOD bit in SPI\_QCTL register and then enable SPI by setting SPIEN in SPI\_CTL0.
3. Write the byte to SPI\_DATA register and the TBE will be cleared.
4. Wait until TBE is set by hardware again before writing the next byte.

**Figure 19-9. Timing diagram of quad write operation in Quad-SPI mode**



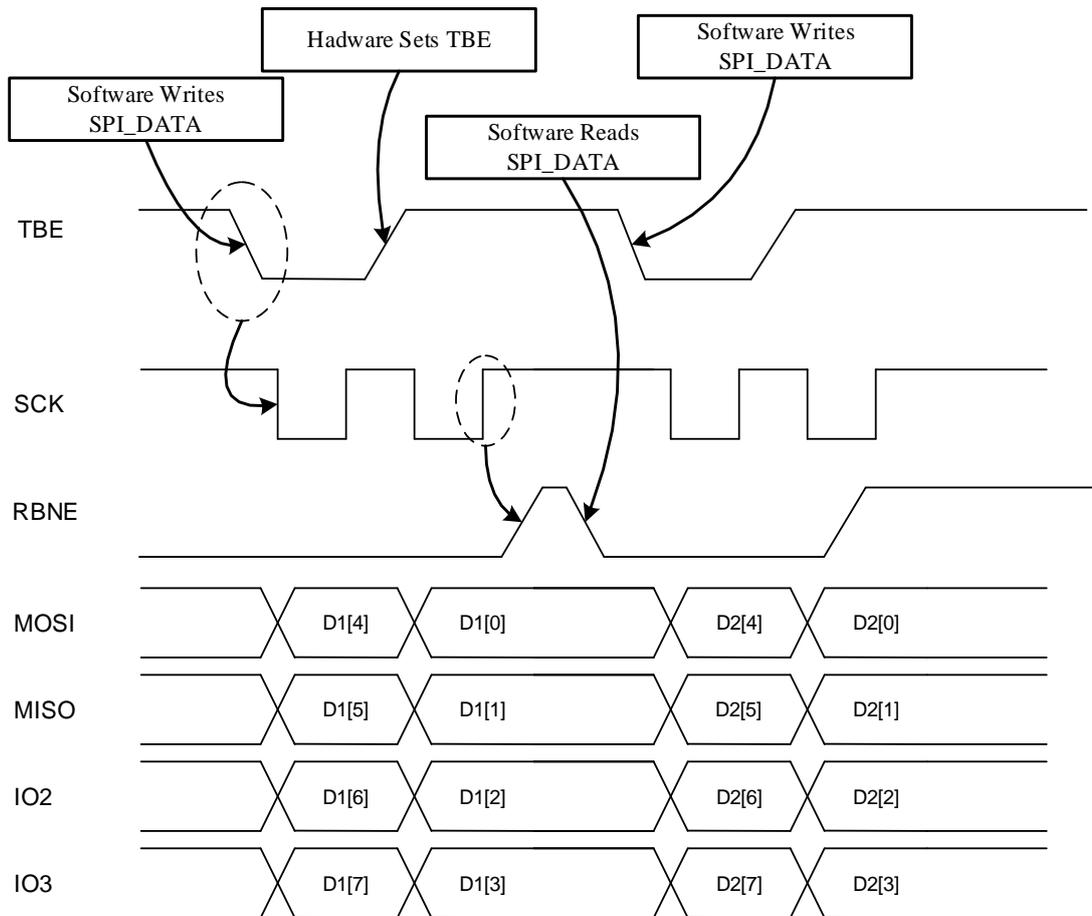
**Quad read operation**

SPI works in quad read mode when QMOD and QRD are both set in SPI\_QCTL register. In this mode, MOSI, MISO, IO2 and IO3 are all used as input pins. SPI begins to generate clock on SCK line as soon as a data is written into SPI\_DATA (TBE is cleared) and SPIEN is set. Writing data into SPI\_DATA is only to generate SCK clocks, so the written data can be any value. Once SPI starts transmission, it always checks SPIEN and TBE status at the end of a frame and stops when condition is not met. So, software should always write dummy data into SPI\_DATA to make SPI generate SCK.

The operation flow for receiving in quad mode:

1. Configure clock prescaler, clock polarity, phase, etc. in SPI\_CTL0 and SPI\_CTL1 register based on your application requirements.
2. Set QMOD and QRD bits in SPI\_QCTL register and then enable SPI by setting SPIEN in SPI\_CTL0 register.
3. Write an arbitrary byte (for example, 0xFF) to SPI\_DATA register.
4. Wait until the RBNE flag is set and read SPI\_DATA to get the received byte.
5. Write an arbitrary byte (for example, 0xFF) to SPI\_DATA to receive the next byte.

Figure 19-10. Timing diagram of quad read operation in Quad-SPI mode



### SPI disabling sequence

Different sequences are used to disable the SPI in different operation modes:

#### MFD SFD

Wait for the last RBNE flag and then receive the last data. Confirm that TBE=1 and TRANS=0. At last, disable the SPI by clearing SPIEN bit.

#### MTU MTB STU STB

Write the last data into SPI\_DATA and wait until the TBE flag is set and then wait until the TRANS flag is cleared. Disable the SPI by clearing SPIEN bit.

#### MRU MRB

After getting the second last RBNE flag, read out this data and delay for a SCK clock time and then, disable the SPI by clearing SPIEN bit. Wait until the last RBNE flag is set and read out the last data.

## SRU SRB

Application can disable the SPI when it doesn't want to receive data, and then wait until the TRANS=0 to ensure the on-going transfer completes.

### Quad-SPI mode for GD32F170xx and GD32F190xx devices

Before leaving quad wire mode or disabling SPI, software should first check that, TBE bit is set and TRANS bit is cleared, then the QMOD bit in SPI\_QCTL register and SPIEN bit in SPI\_CTL0 register are cleared.

#### 19.5.4. DMA function

The DMA function frees the application from data writing and reading process during transfer, to improve the system efficiency.

DMA function in SPI is enabled by setting DMATEN and DMAREN bits in SPI\_CTL1 register. To use DMA function, application should first correctly configure DMA modules, then configure SPI module according to the initialization sequence, at last enable SPI.

After being enabled, If DMATEN is set, SPI will generate a DMA request each time TBE=1, then DMA will acknowledge to this request and write data into the SPI\_DATA register automatically. If DMAREN is set, SPI will generate a DMA request each time RBNE=1, then DMA will acknowledge to this request and read data from the SPI\_DATA register automatically.

#### 19.5.5. CRC function

There are two CRC calculators in SPI: one for transmission and the other for reception. The CRC calculation uses the polynomial in SPI\_CRCPOLY register.

Application can switch on the CRC function by setting CRCEN bit in SPI\_CTL0 register. The CRC calculators continuously calculate CRC for each bit transmitted and received on lines, and the calculated CRC values can be read from SPI\_TCRC and SPI\_RCRC register.

To transmit the calculated CRC value, application should set the CRCNT bit in SPI\_CTL0 register after the last data is written to the transmit buffer. In full-duplex mode (MFD or SFD) the SPI treats the incoming data as a CRC value when it transmits a CRC and will check the received CRC value. In reception mode (MRB, MRU, SRU and SRB), the application should set the CRCNT bit after the second-last data frame is received. When CRC checking fails, the CRCERR flag will be set.

If DMA function is enabled, application doesn't need to operate CRCNT bit and hardware will automatically process the CRC transmitting and checking.

## 19.6. SPI interrupts

### 19.6.1. Status flags

- Transmit buffer empty flag (TBE)

This bit is set when the transmit buffer is empty, the software can write the next data to the transmit buffer by writing the SPI\_DATA register.

- Receive buffer not empty flag (RBNE)

This bit is set when receive buffer is not empty, which means that one data is received and stored in the receive buffer, and software can read the data by reading the SPI\_DATA register.

- SPI Transmitting On-Going flag (TRANS)

TRANS is a status flag to indicate whether the transfer is on-going or not. It is set and cleared by internal hardware and not controlled by software. This flag doesn't generate any interrupt.

**Note:** TRANS is set after the first bit is transmitted. So TBE or RBNE must be judged as the communication finished, instead of TRANS.

### 19.6.2. Error conditions

- Configuration Fault Error (CONFERR)

CONFERR is an error flag in master mode. In NSS hardware mode and the NSSDRV is not enabled, the CONFERR is set when the NSS pin is pulled low. In NSS software mode, the CONFERR is set when the SWNSS bit is 0. When the CONFERR is set, the SPIEN bit and the MSTMOD bit are cleared by hardware, the SPI is disabled and the device is forced into slave mode.

The SPIEN and MSTMOD bit are write protection until the CONFERR is cleared. The CONFERR bit of the slave cannot be set. In a multi-master configuration, the device can be in slave mode with CONFERR bit set, which means there might have been a multi-master conflict for system control.

- Rx Overrun Error (RXORERR)

The RXORERR bit is set if a data is received when the RBNE is set. That means, the last data has not been read out and the newly incoming data is received. The receive buffer contents won't be covered with the newly incoming data, so the newly incoming data is lost.

- CRC Error (CRCERR)

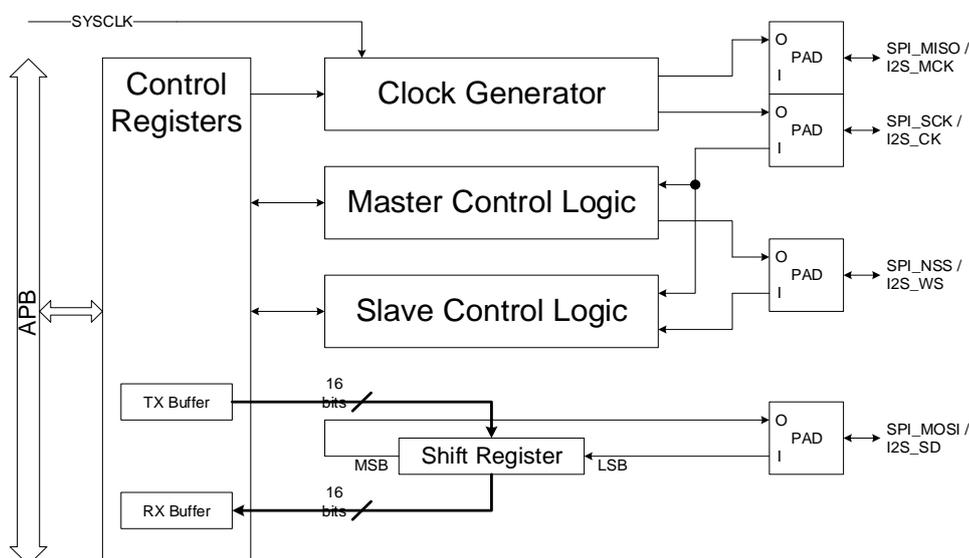
When the CRCEN bit is set, the CRC calculation result of the received data in the SPI\_RCRC register is compared with the received CRC value after the last data, the CRCERR is set when they are different.

**Table 19-4. SPI interrupt requests**

Flag	Description	Clear Method	Interrupt Enable bit
TBE	Transmit buffer empty	Write SPI_DATA register.	TBEIE
RBNE	Receive buffer not empty	Read SPI_DATA register.	RBNEIE
CONFERR	Configuration Fault Error	Read or write SPI_STAT register, then write SPI_CTL0 register.	ERRIE
RXORERR	Rx Overrun Error	Read SPI_DATA register, then read SPI_STAT register.	
CRCERR	CRC error	Write 0 to CRCERR bit	

## 19.7. I2S block diagram

**Figure 19-11. Block diagram of I2S**



There are five sub modules to support I2S function, including control registers, clock generator, master control logic, slave control logic and shift register. All the user configuration registers are implemented in the control registers module, including the TX buffer and RX buffer. The clock generator is used to produce I2S communication clock in master mode. The master control logic is implemented to generate the I2S\_WS signal and control the communication in master mode. The slave control logic is implemented to control the communication in slave mode according to the received I2SCK and I2S\_WS. The shift register handles the serial data transmission and reception on I2S\_SD.

## 19.8. I2S signal description

There are four pins on the I2S interface, including I2S\_CK, I2S\_WS, I2S\_SD and I2S\_MCK.

I2S\_CK is the serial clock signal, which shares the same pin with SPI\_SCK. I2S\_WS is the frame control signal, which shares the same pin with SPI\_NSS. I2S\_SD is the serial data signal, which shares the same pin with SPI\_MOSI. I2S\_MCK is the master clock signal, which shares the same pin with SPI\_MISO. It produces a frequency rate equal to  $256 \times F_s$ , and  $F_s$  is the audio sampling frequency.

## 19.9. I2S function overview

### 19.9.1. I2S audio standards

The I2S audio standard is selected by the I2SSTD bits in the SPI\_I2SCTL register. Four audio standards are supported, including I2S Phillips standard, MSB justified standard, LSB justified standard, and PCM standard. All standards except PCM handle audio data time-multiplexed on two channels (the left channel and the right channel). For these standards, the I2S\_WS signal indicates the channel side. For PCM standard, the I2S\_WS signal indicates frame synchronization information.

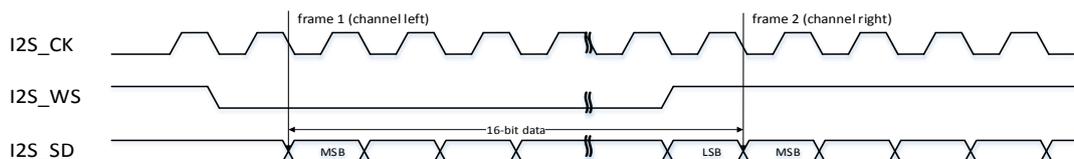
The data length and the channel length are configured by the DTLEN bits and CHLEN bit in the SPI\_I2SCTL register. Since the channel length must be greater than or equal to the data length, four packet types are available. They are 16-bit data packed in 16-bit frame, 16-bit data packed in 32-bit frame, 24-bit data packed in 32-bit frame, and 32-bit data packed in 32-bit frame. The data buffer for transmission and reception is 16-bit wide. In the case that the data length is 24 bits or 32 bits, two write or read operations to or from the SPI\_DATA register are needed to complete a frame. In the case that the data length is 16 bits, only one write or read operation to or from the SPI\_DATA register is needed to complete a frame. When using 16-bit data packed in 32-bit frame, 16-bit 0 is inserted by hardware automatically to extend the data to 32-bit format.

For all standards and packet types, the most significant bit (MSB) is always sent first. For all standards based on two channels time-multiplexed, the channel left is always sent first followed by the channel right.

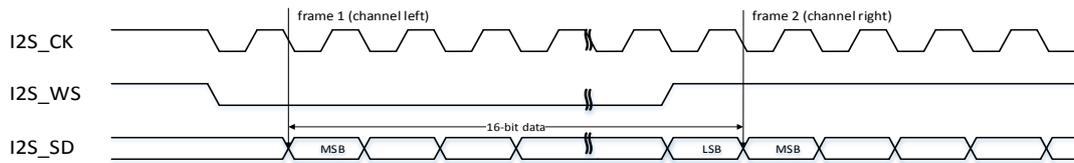
#### I2S Phillips standard

For I2S Phillips standard, I2S\_WS and I2S\_SD are updated on the falling edge of I2S\_CK. The timing diagrams for each configuration are shown below.

**Figure 19-12. I2S Phillips standard timing diagram (DTLEN=00, CHLEN=0, CKPL=0)**

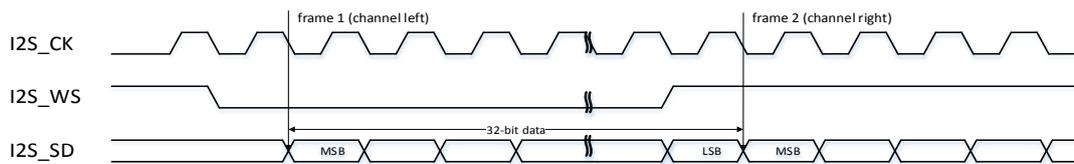


**Figure 19-13. I2S Phillips standard timing diagram (DTLEN=00, CHLEN=0, CKPL=1)**

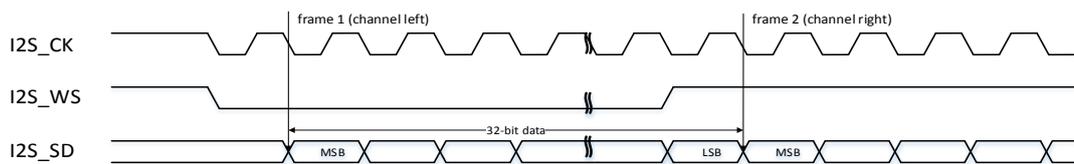


When the packet type is 16-bit data packed in 16-bit frame, only one write or read operation to or from the SPI\_DATA register is needed to complete a frame.

**Figure 19-14. I2S Phillips standard timing diagram (DTLEN=10, CHLEN=1, CKPL=0)**

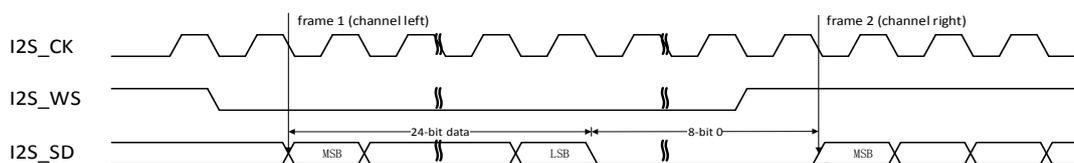


**Figure 19-15. I2S Phillips standard timing diagram (DTLEN=10, CHLEN=1, CKPL=1)**

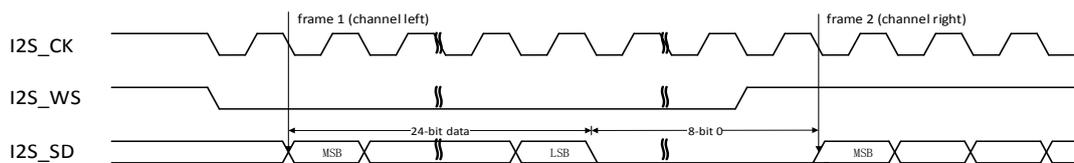


When the packet type is 32-bit data packed in 32-bit frame, two write or read operations to or from the SPI\_DATA register are needed to complete a frame. In transmission mode, if a 32-bit data is going to be sent, the first data written to the SPI\_DATA register should be the higher 16 bits, and the second one should be the lower 16 bits. In reception mode, if a 32-bit data is received, the first data read from the SPI\_DATA register should be higher 16 bits, and the second one should be the lower 16 bits.

**Figure 19-16. I2S Phillips standard timing diagram (DTLEN=01, CHLEN=1, CKPL=0)**



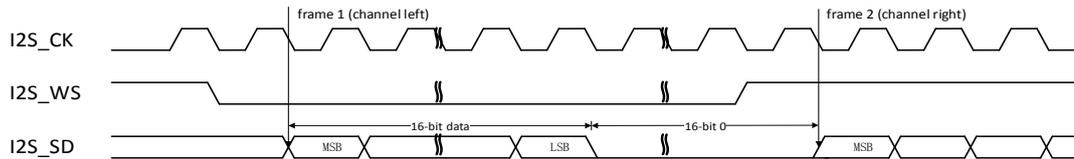
**Figure 19-17. I2S Phillips standard timing diagram (DTLEN=01, CHLEN=1, CKPL=1)**



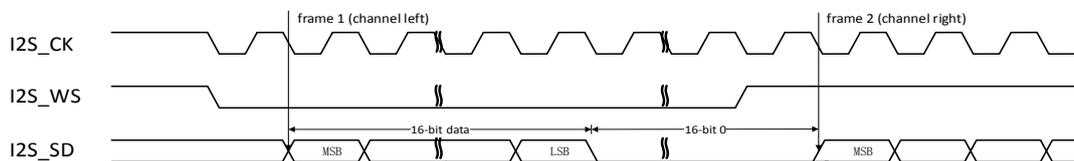
When the packet type is 24-bit data packed in 32-bit frame, two write or read operations to or from the SPI\_DATA register are needed to complete a frame. In transmission mode, if a 24-

bit data D[23:0] is going to be sent, the first data written to the SPI\_DATA register should be the higher 16 bits: D[23:8], and the second one should be a 16-bit data. The higher 8 bits of this 16-bit data should be D[7:0] and the lower 8 bits can be any value. In reception mode, if a 24-bit data D[23:0] is received, the first data read from the SPI\_DATA register is D[23:8], and the second one is a 16-bit data. The higher 8 bits of this 16-bit data are D[7:0] and the lower 8 bits are zeros.

**Figure 19-18. I2S Phillips standard timing diagram (DTLEN=00, CHLEN=1, CKPL=0)**



**Figure 19-19. I2S Phillips standard timing diagram (DTLEN=00, CHLEN=1, CKPL=1)**

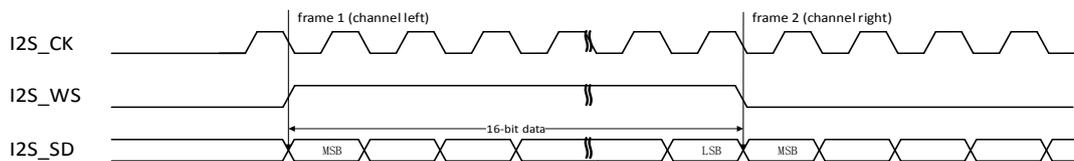


When the packet type is 16-bit data packed in 32-bit frame, only one write or read operation to or from the SPI\_DATA register is needed to complete a frame. The 16 remaining bits are forced by hardware to 0x0000 to extend the data to 32-bit format.

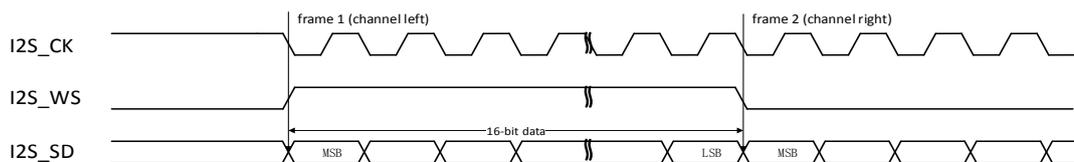
### MSB justified standard

For MSB justified standard, I2S\_WS and I2S\_SD are updated on the falling edge of I2S\_CK. The SPI\_DATA register is handled in the exactly same way as that for I2S Phillips standard. The timing diagrams for each configuration are shown below.

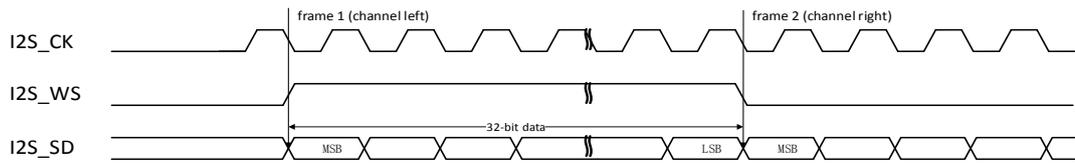
**Figure 19-20. MSB justified standard timing diagram (DTLEN=00, CHLEN=0, CKPL=0)**



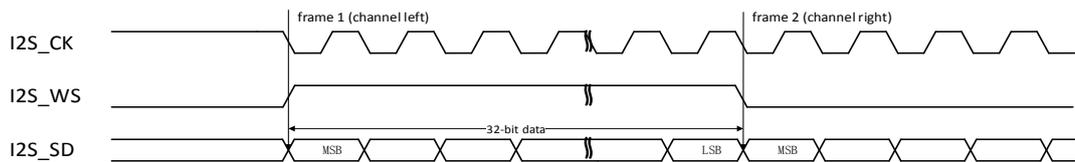
**Figure 19-21. MSB justified standard timing diagram (DTLEN=00, CHLEN=0, CKPL=1)**



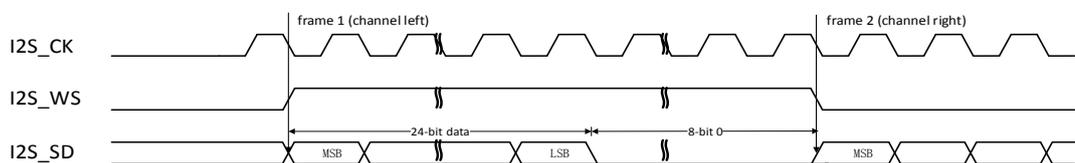
**Figure 19-22. MSB justified standard timing diagram (DTLEN=10, CHLEN=1, CKPL=0)**



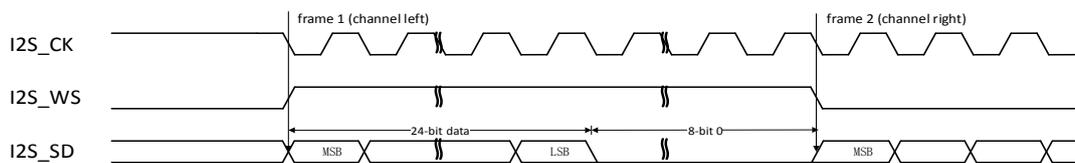
**Figure 19-23. MSB justified standard timing diagram (DTLEN=10, CHLEN=1, CKPL=1)**



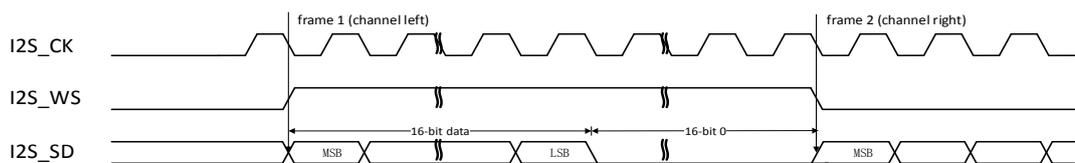
**Figure 19-24. MSB justified standard timing diagram (DTLEN=01, CHLEN=1, CKPL=0)**



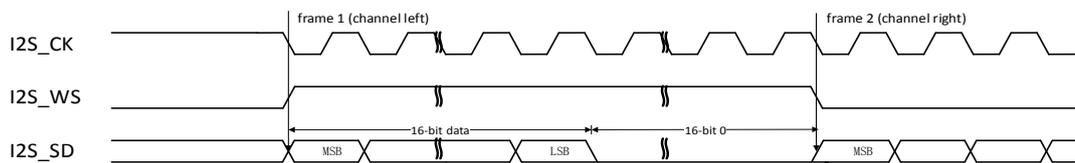
**Figure 19-25. MSB justified standard timing diagram (DTLEN=01, CHLEN=1, CKPL=1)**



**Figure 19-26. MSB justified standard timing diagram (DTLEN=00, CHLEN=1, CKPL=0)**



**Figure 19-27. MSB justified standard timing diagram (DTLEN=00, CHLEN=1, CKPL=1)**

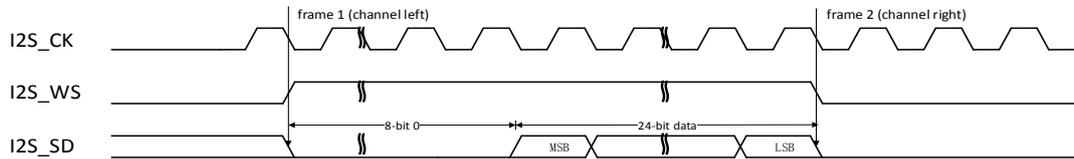


## LSB justified standard

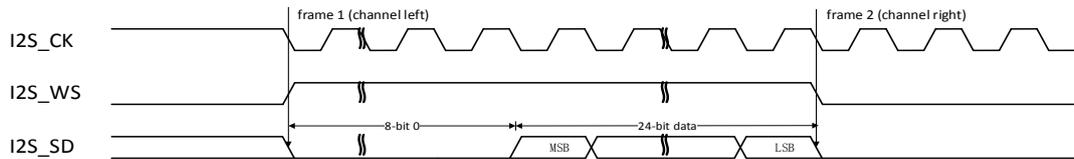
For LSB justified standard, I2S\_WS and I2S\_SD are updated on the falling edge of I2S\_CK. In the case that the channel length is equal to the data length, LSB justified standard and MSB justified standard are exactly the same. In the case that the channel length is greater

than the data length, the valid data is aligned to LSB for LSB justified standard while the valid data is aligned to MSB for MSB justified standard. The timing diagrams for the cases that the channel length is greater than the data length are shown below.

**Figure 19-28. LSB justified standard timing diagram (DTLEN=01, CHLEN=1, CKPL=0)**

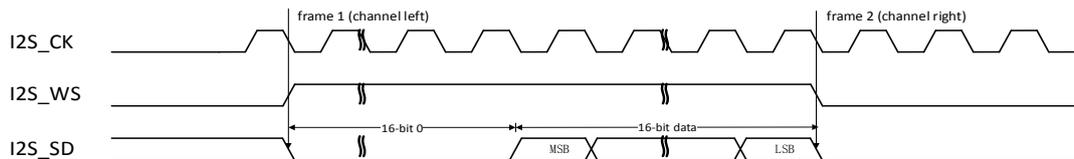


**Figure 19-29. LSB justified standard timing diagram (DTLEN=01, CHLEN=1, CKPL=1)**

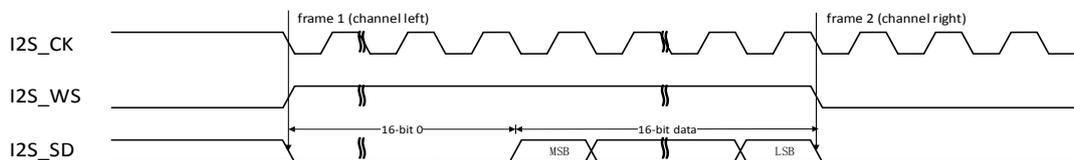


When the packet type is 24-bit data packed in 32-bit frame, two write or read operations to or from the SPI\_DATA register are needed to complete a frame. In transmission mode, if a 24-bit data D [23:0] is going to be sent, the first data written to the SPI\_DATA register should be a 16-bit data. The higher 8 bits of the 16-bit data can be any value and the lower 8 bits should be D [23:16]. The second data written to the SPI\_DATA register should be D [15:0]. In reception mode, if a 24-bit data D [23:0] is received, the first data read from the SPI\_DATA register is a 16-bit data. The high 8 bits of this 16-bit data are zeros and the lower 8 bits are D [23:16]. The second data read from the SPI\_DATA register is D [15:0].

**Figure 19-30. LSB justified standard timing diagram (DTLEN=00, CHLEN=1, CKPL=0)**



**Figure 19-31. LSB justified standard timing diagram (DTLEN=00, CHLEN=1, CKPL=1)**

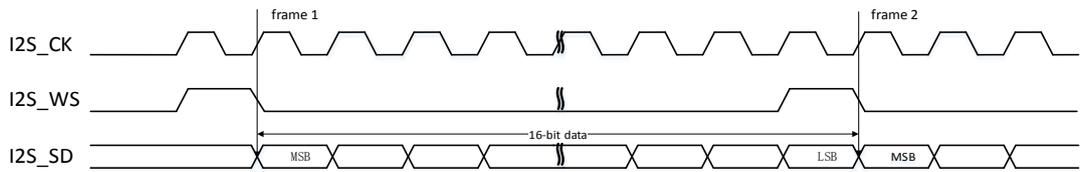


When the packet type is 16-bit data packed in 32-bit frame, only one write or read operation to or from the SPI\_DATA register is needed to complete a frame. The 16 remaining bits are forced by hardware to 0x0000 to extend the data to 32-bit format.

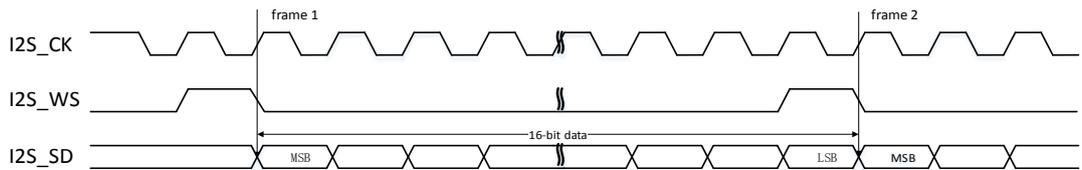
**PCM standard**

For PCM standard, I2S\_WS and I2S\_SD are updated on the rising edge of I2S\_CK, and the I2S\_WS signal indicates frame synchronization information. Both the short frame synchronization mode and the long frame synchronization mode are available and configurable using the PCMSMOD bit in the SPI\_I2SCTL register. The SPI\_DATA register is handled in the exactly same way as that for I2S Phillips standard. The timing diagrams for each configuration of the short frame synchronization mode are shown below.

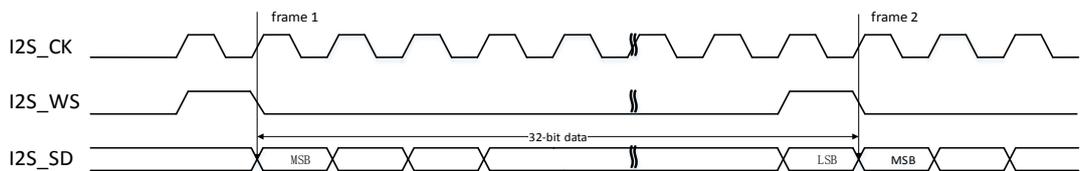
**Figure 19-32. PCM standard short frame synchronization mode timing diagram (DTLEN=00, CHLEN=0, CKPL=0)**



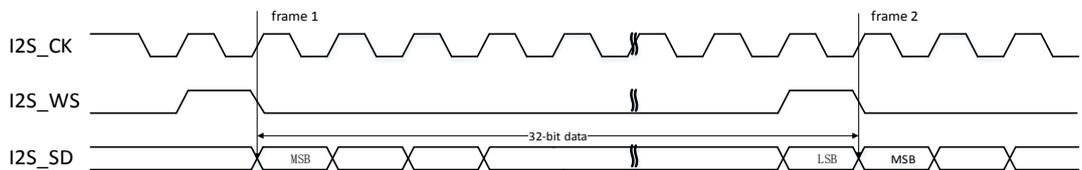
**Figure 19-33. PCM standard short frame synchronization mode timing diagram (DTLEN=00, CHLEN=0, CKPL=1)**



**Figure 19-34. PCM standard short frame synchronization mode timing diagram (DTLEN=10, CHLEN=1, CKPL=0)**

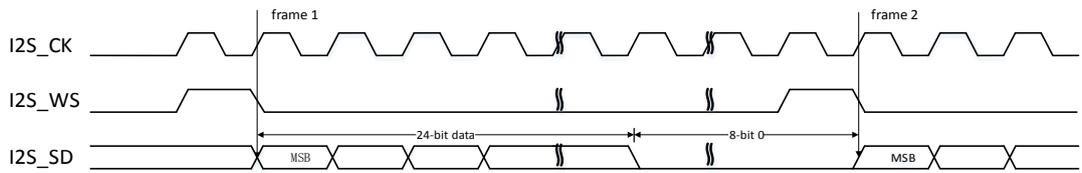


**Figure 19-35. PCM standard short frame synchronization mode timing diagram (DTLEN=10, CHLEN=1, CKPL=1)**

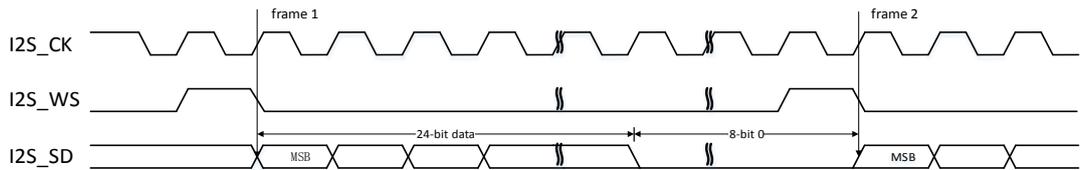


**Figure 19-36. PCM standard short frame synchronization mode timing diagram**

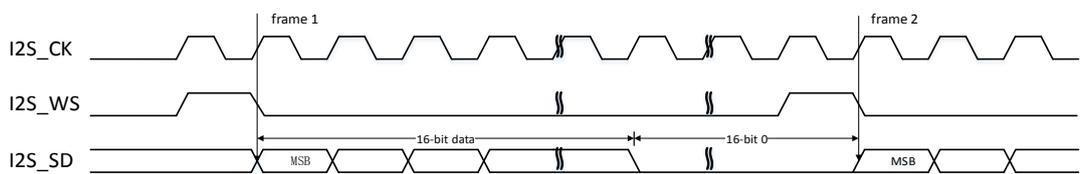
**(DTLEN=01, CHLEN=1, CKPL=0)**



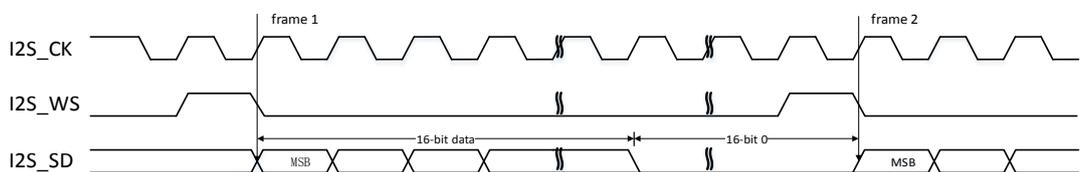
**Figure19-37. PCM standard short frame synchronization mode timing diagram (DTLEN=01, CHLEN=1, CKPL=1)**



**Figure 19-38. PCM standard short frame synchronization mode timing diagram (DTLEN=00, CHLEN=1, CKPL=0)**

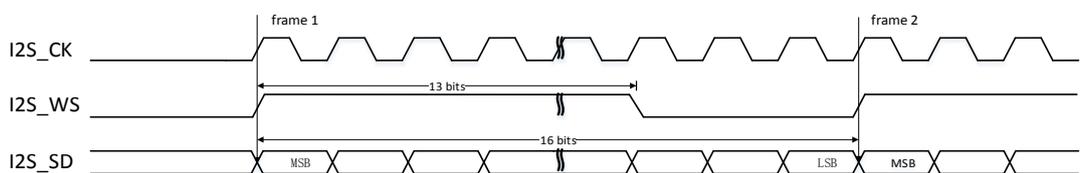


**Figure 19-39. PCM standard short frame synchronization mode timing diagram (DTLEN=00, CHLEN=1, CKPL=1)**



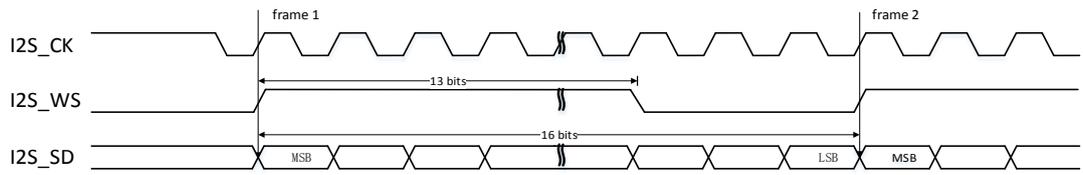
The timing diagrams for each configuration of the long frame synchronization mode are shown below.

**Figure 19-40. PCM standard long frame synchronization mode timing diagram (DTLEN=00, CHLEN=0, CKPL=0)**

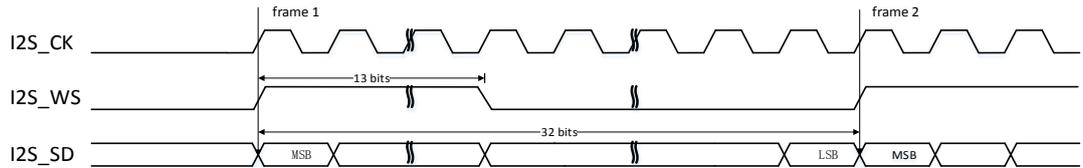


**Figure19-41. PCM standard long frame synchronization mode timing diagram**

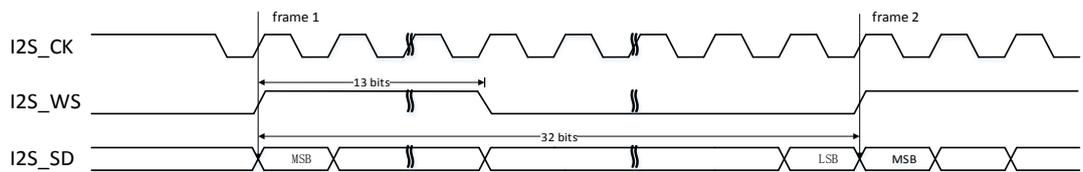
**(DTLEN=00, CHLEN=0, CKPL=1)**



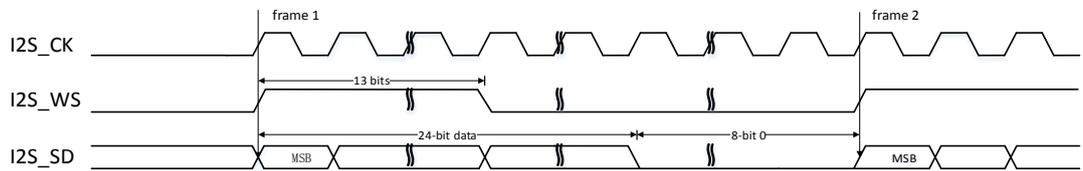
**Figure 19-42. PCM standard long frame synchronization mode timing diagram (DTLEN=10, CHLEN=1, CKPL=0)**



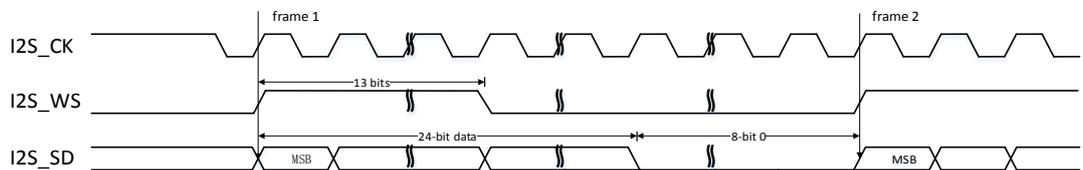
**Figure 19-43. PCM standard long frame synchronization mode timing diagram (DTLEN=10, CHLEN=1, CKPL=1)**



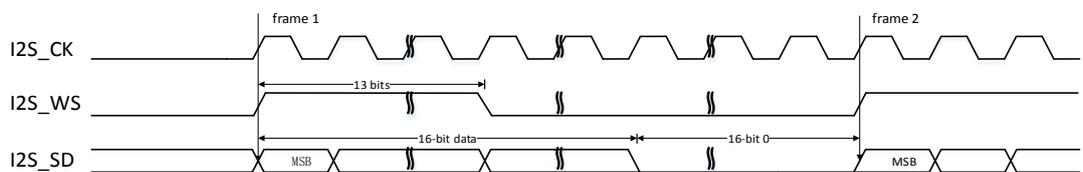
**Figure 19-44. PCM standard long frame synchronization mode timing diagram (DTLEN=01, CHLEN=1, CKPL=0)**



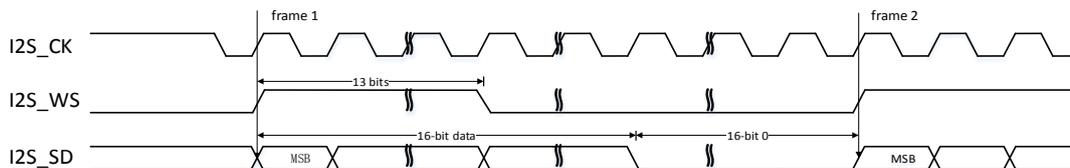
**Figure 19-45. PCM standard long frame synchronization mode timing diagram (DTLEN=01, CHLEN=1, CKPL=1)**



**Figure 19-46. PCM standard long frame synchronization mode timing diagram (DTLEN=00, CHLEN=1, CKPL=0)**

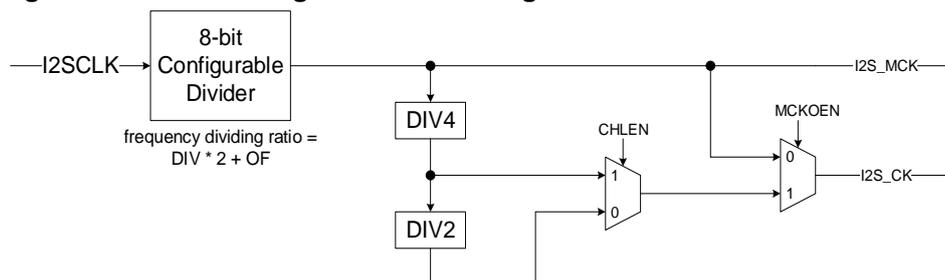


**Figure 19-47. PCM standard long frame synchronization mode timing diagram (DTLEN=00, CHLEN=1, CKPL=1)**



### 19.9.2. I2S clock

**Figure 19-48. Block diagram of I2S clock generator**



The block diagram of I2S clock generator is shown as [Figure 19-48. Block diagram of I2S clock generator](#). The I2S interface clocks are configured by the DIV bits, the OF bit, the MCKOEN bit in the SPI\_I2SPSC register and the CHLEN bit in the SPI\_I2SCTL register. The source clock is the system clock(CK\_SYS). The I2S bitrate can be calculated by the formulas shown in [Table 19-5. I2S bitrate calculation formulas](#).

**Table 19-5. I2S bitrate calculation formulas**

MCKOEN	CHLEN	Formula
0	0	$I2SCLK / (DIV * 2 + OF)$
0	1	$I2SCLK / (DIV * 2 + OF)$
1	0	$I2SCLK / (8 * (DIV * 2 + OF))$
1	1	$I2SCLK / (4 * (DIV * 2 + OF))$

The relationship between audio sampling frequency (Fs) and I2S bitrate is defined by the following formula:

$$Fs = I2S \text{ bitrate} / (\text{number of bits per channel} * \text{number of channels})$$

So, in order to get the desired audio sampling frequency, the clock generator needs to be configured according to the formulas listed in [Table 19-6. Audio sampling frequency calculation formulas](#).

**Table 19-6. Audio sampling frequency calculation formulas**

MCKOEN	CHLEN	Formula
0	0	$I2SCLK / (32 * (DIV * 2 + OF))$
0	1	$I2SCLK / (64 * (DIV * 2 + OF))$
1	0	$I2SCLK / (256 * (DIV * 2 + OF))$
1	1	$I2SCLK / (256 * (DIV * 2 + OF))$

### 19.9.3. Operation

#### Operation modes

The operation mode is selected by the I2SOPMOD bits in the SPI\_I2SCTL register. There are four available operation modes, including master transmission mode, master reception mode, slave transmission mode, and slave reception mode. The direction of I2S interface signals for each operation mode is shown in the [Table 19-7. Direction of I2S interface signals for each operation mode.](#)

**Table 19-7. Direction of I2S interface signals for each operation mode**

Operation mode	I2S_MCK	I2S_CK	I2S_WS	I2S_SD
Master transmission	output or NU(1)	output	output	output
Master reception	output or NU(1)	output	output	input
Slave transmission	input or NU(1)	input	input	output
Slave reception	input or NU(1)	input	input	input

1. NU means the pin is not used by I2S and can be used by other functions.

#### I2S initialization sequence

I2S initialization sequence contains five steps shown below. In order to initialize I2S working in master mode, all the five steps should be done. In order to initialize I2S working in slave mode, only step 2, step 3, step 4 and step 5 should be done.

- Step 1: Configure the DIV [7:0] bits, the OF bit, and the MCKOEN bit in the SPI\_I2SPSC register, in order to define the I2S bitrate and whether I2S\_MCK needs to be provided or not.
- Step 2: Configure the CKPL in the SPI\_I2SCTL register, in order to define the idle state clock polarity.
- Step 3: Configure the I2SSEL bit, the I2SSTD [1:0] bits, the PCMSMOD bit, the I2SOPMOD [1:0] bits, the DTLEN [1:0] bits, and the CHLEN bit in the SPI\_I2SCTL

register, in order to define the I2S feature.

- Step 4: Configure the TBEIE bit, the RBNEIE bit, the ERRIE bit, the DMATEN bit, and the DMAREN bit in the SPI\_CTL1 register, in order to select the potential interrupt sources and the DMA capabilities. This step is optional.
- Step 5: Set the I2SEN bit in the SPI\_I2SCTL register to enable I2S.

### **I2S master transmission sequence**

The TBE flag is used to control the transmission sequence. As is mentioned before, the TBE flag indicates that the transmit buffer is empty, and an interrupt will be generated if the TBEIE bit in the SPI\_CTL1 register is set. At the beginning, the transmit buffer is empty (TBE is high) and no transmission sequence is processing in the shift register. When a half word is written to the SPI\_DATA register (TBE goes low), the data is transferred from the transmit buffer to the shift register (TBE goes high) immediately. At the moment, the transmission sequence begins.

The data is parallel loaded into the 16-bit shift register, and shifted out serially to the I2S\_SD pin, MSB first. The next data should be written to the SPI\_DATA register, when the TBE flag is high. After a write operation to the SPI\_DATA register, the TBE flag goes low. When the current transmission finishes, the data in the transmit buffer is loaded into the shift register, and the TBE flag goes back high. Software should write the next audio data into SPI\_DATA register before the current data finishes, otherwise, the audio data transmission is not continuous.

For all standards except PCM, the I2SCH flag is used to distinguish the channel side to which the data to transfer belongs. The I2SCH flag is refreshed at the moment when the TBE flag goes high. At the beginning, the I2SCH flag is low, indicating the left channel data should be written to the SPI\_DATA register.

In order to switch off I2S, it is mandatory to clear the I2SEN bit after the TBE flag is high and the TRANS flag is low.

### **I2S master reception sequence**

The RBNE flag is used to control the reception sequence. As is mentioned before, the RBNE flag indicates the receive buffer is not empty, and an interrupt will be generated if the RBNEIE bit in the SPI\_CTL1 register is set. The reception sequence begins immediately when the I2SEN bit in the SPI\_I2SCTL register is set. At the beginning, the receive buffer is empty (RBNE is low). When a reception sequence finishes, the received data in the shift register is loaded into the receive buffer (RBNE goes high). The data should be read from the SPI\_DATA register, when the RBNE flag is high. After a read operation to the SPI\_DATA register, the RBNE flag goes low. It is mandatory to read the SPI\_DATA register before the end of the next reception. Otherwise, reception overrun error occurs. The RXORERR flag is set and an interrupt may be generated if the ERRIE bit in the SPI\_CTL1 register is set. In this case, it is necessary to switch off and then switch on I2S before resuming the communication.

For all standards except PCM, the I2SCH flag is used to distinguish the channel side to which the received data belongs. The I2SCH flag is refreshed at the moment when the RBNE flag goes high.

Different sequences are used to disable the I2S in different standards, data length and channel length. The sequences for each case are described below.

- 16-bit data packed in 32-bit frame in the LSB justified standard (DTLEN = 00, CHLEN = 1, and I2SSTD = 10)

1. Wait for the second last RBNE
2. Then wait 17 I2S CK clock (clock on I2S\_CK pin) cycles
3. Clear the I2SEN bit

- 16-bit data packed in 32-bit frame in the audio standards except the LSB justified standard (DTLEN = 00, CHLEN = 1, and I2SSTD is not equal to 10)

1. Wait for the last RBNE
2. Then wait one I2S clock cycle
3. Clear the I2SEN bit

- For all other cases

1. Wait for the second last RBNE
2. Then wait one I2S clock cycle
3. Clear the I2SEN bit

### **I2S slave transmission sequence**

The transmission sequence in slave mode is similar to that in master mode. The difference between them is described below.

In slave mode, the slave has to be enabled before the external master starts the communication. The transmission sequence begins when the external master sends the clock and when the I2S\_WS signal requests the transfer of data. The data has to be written to the SPI\_DATA register before the master initiates the communication. Software should write the next audio data into SPI\_DATA register before the current data finishes. Otherwise, transmission underrun error occurs. The TXURERR flag is set and an interrupt may be generated if the ERRIE bit in the SPI\_CTL1 register is set. In this case, it is mandatory to switch off and switch on I2S to resume the communication. In slave mode, I2SCH is sensitive to the I2S\_WS signal coming from the external master.

In order to switch off I2S, it is mandatory to clear the I2SEN bit after the TBE flag is high and the TRANS flag is low.

## I2S slave reception sequence

The reception sequence in slave mode is similar to that in master mode. The difference between them is described below.

In slave mode, the slave has to be enabled before the external master starts the communication. The reception sequence begins when the external master sends the clock and when the I2S\_WS signal indicates a start of the data transfer. In slave mode, I2SCH is sensitive to the I2S\_WS signal coming from the external master.

In order to switch off I2S, it is mandatory to clear the I2SEN bit immediately after receiving the last RBNE.

### 19.9.4. DMA function

DMA function is the same as SPI mode. The only difference is that the CRC function is not available in I2S mode.

## 19.10. I2S interrupts

### 19.10.1. Status flags

There are four status flags implemented in the SPI\_STAT register, including TBE, RBNE, TRANS and I2SCH. The user can use them to fully monitor the state of the I2S bus.

- Transmit buffer empty flag (TBE)

This bit is set when the transmit buffer is empty, the software can write the next data to the transmit buffer by writing the SPI\_DATA register.

- Receive buffer not empty flag (RBNE)

This bit is set when receive buffer is not empty, which means that one data is received and stored in the receive buffer, and software can read the data by reading the SPI\_DATA register.

- I2S Transmitting On-Going flag (TRANS)

TRANS is a status flag to indicate whether the transfer is on-going or not. It is set and cleared by internal hardware and not controlled by software. This flag doesn't generate any interrupt.

- I2S channel side flag (I2SCH)

This flag indicates the channel side information of the current transfer and has no meaning in PCM mode. It is updated when TBE rises in transmission mode or RBNE rises in reception mode. This flag doesn't generate any interrupt.

### 19.10.2. Error conditions

There are two error conditions:

- Transmission Underrun Error Flag (TXURERR)

This condition occurs when the transmit buffer is empty when the valid SCK signal starts in slave transmission mode.

- Reception Overrun Error Flag (RXORERR)

This condition occurs when the receive buffer is full and a newly incoming data has been completely received. When overrun occurs, the data in receive buffer is not updated and the newly incoming data is lost.

I2S interrupt events and corresponding enabled bits are summed up in the [Table 19-8. I2S interrupt.](#)

**Table 19-8. I2S interrupt**

Flag Name	Description	Clear Method	Interrupt Enable bit
TBE	Transmit buffer empty	Write SPI_DATA register	TBEIE
RBNE	Receive buffer not empty	Read SPI_DATA register	RBNEIE
TXURERR	Transmission underrun error	Read SPI_STAT register	ERRIE
RXORERR	Reception overrun error	Read SPI_DATA register and then read SPI_STAT register.	

## 19.11. Register definition

### 19.11.1. Control register 0 (SPI\_CTL0)

Address offset: 0x00

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit).

This register has no meaning in I2S 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
BDEN	BDOEN	CRCEN	CRCNT	FF16	RO	SWNSS EN	SWNSS	LF	SPIEN	PSC [2:0]		MSTMOD	CKPL	CKPH	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15	BDEN	Bidirectional enable 0: 2 line unidirectional transmit mode 1: 1 line bidirectional transmit mode. The information transfers between the MOSI pin in master and the MISO pin in slave.
14	BDOEN	Bidirectional Transmit Output Enable When BDEN is set, this bit determines the direction of transfer. 0: Work in receive-only mode 1: Work in transmit-only mode
13	CRCEN	CRC Calculation Enable 0: CRC calculation is disabled 1: CRC calculation is enabled.
12	CRCNT	CRC Next Transfer 0: Next transfer is Data 1: Next transfer is CRC value (TCR) When the transfer is managed by DMA, CRC value is transferred by hardware. This bit should be cleared. In full-duplex or transmit-only mode, set this bit after the last data is written to SPI_DATA register. In receive only mode, set this bit after the second last data is received.
11	FF16	Data frame format 0: 8-bit data frame format 1: 16-bit data frame format

10	RO	<p>Receive only</p> <p>When BDEN is cleared, this bit determines the direction of transfer.</p> <p>0: Full-duplex</p> <p>1: Receive-only</p>
9	SWNSSEN	<p>NSS Software Mode Selection</p> <p>0: NSS hardware mode. The NSS level depends on NSS pin.</p> <p>1: NSS software mode. The NSS level depends on SWNSS bit.</p>
8	SWNSS	<p>NSS Pin Selection In NSS Software Mode</p> <p>0: NSS pin is pulled low</p> <p>1: NSS pin is pulled high</p> <p>This bit has an effect only when the SWNSSEN bit is set.</p>
7	LF	<p>LSB First Mode</p> <p>0: Transmit MSB first</p> <p>1: Transmit LSB first</p>
6	SPIEN	<p>SPI Enable</p> <p>0: SPI peripheral is disabled</p> <p>1: SPI peripheral is enabled</p>
5:3	PSC[2:0]	<p>Master Clock Prescaler Selection</p> <p>000: PCLK/2      100: PCLK/32</p> <p>001: PCLK/4      101: PCLK/64</p> <p>010: PCLK/8      110: PCLK/128</p> <p>011: PCLK/16     111: PCLK/256</p> <p>PCLK means PCLK2 when using SPI0 or PCLK1 when using SPI1.</p>
2	MSTMOD	<p>Master Mode Enable</p> <p>0: Slave mode</p> <p>1: Master mode</p>
1	CKPL	<p>Clock Polarity Selection</p> <p>0: CLK pin is pulled low when SPI is idle</p> <p>1: CLK pin is pulled high when SPI is idle</p>
0	CKPH	<p>Clock Phase Selection</p> <p>0: Capture the first data at the first clock transition.</p> <p>1: Capture the first data at the second clock transition</p>

## 19.11.2. Control register 1 (SPI\_CTL1)

Address offset: 0x04

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit).

Reserved															
----------	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TBEIE	RBNEIE	ERRIE	Reserved	NSSDRV	DMATEN	DMAREN	
								rw	rw	rw			rw	rw	rw

Bits	Fields	Descriptions
31:7	Reserved	Must be kept at reset value.
7	TBEIE	Transmit Buffer Empty Interrupt Enable 0: TBE interrupt is disabled. 1: TBE interrupt is enabled. An interrupt is generated when the TBE bit is set
6	RBNEIE	Receive Buffer Not Empty Interrupt Enable 0: RBNE interrupt is disabled. 1: RBNE interrupt is enabled. An interrupt is generated when the RBNE bit is set
5	ERRIE	Errors Interrupt Enable. 0: Error interrupt is disabled. 1: Error interrupt is enabled. An interrupt is generated when the CRCERR bit or the CONFERR bit or the RXORERR bit or the TXURERR bit is set.
4:3	Reserved	Must be kept at reset value.
2	NSSDRV	Drive NSS Output 0: NSS output is disabled. 1: NSS output is enabled. If the NSS pin is configured as output, the NSS pin is pulled low in master mode when SPI is enabled. If the NSS pin is configured as input, the NSS pin should be pulled high in master mode, and this bit has no effect.
1	DMATEN	Transmit Buffer DMA Enable 0: Transmit buffer DMA is disabled 1: Transmit buffer DMA is enabled, when the TBE bit in SPI_STAT is set, it will be a DMA request at corresponding DMA channel.
0	DMAREN	Receive Buffer DMA Enable 0: Receive buffer DMA is disabled 1: Receive buffer DMA is enabled, when the RBNE bit in SPI_STAT is set, it will be a DMA request at corresponding DMA channel.

### 19.11.3. Status register (SPI\_STAT)

Address offset: 0x08

Reset value: 0x0002

This register can be accessed by half-word (16-bit) or word (32-bit).

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								TRANS	RXORERR	CONFERR	CRCERR	TXURERR	I2SCH	TBE	RBNE
								r	r	r	rc_w0	r	r	r	r

Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value.
7	TRANS	Transmitting On-going Bit 0: SPI or I2S is idle. 1: SPI or I2S is currently transmitting and/or receiving a frame This bit is set and cleared by hardware.
6	RXORERR	Reception Overrun Error Bit 0: No reception overrun error occurs. 1: Reception overrun error occurs. This bit is set by hardware and cleared by a read operation on the SPI_DATA register followed by a read access to the SPI_STAT register.
5	CONFERR	SPI Configuration error 0: No configuration fault occurs 1: Configuration fault occurred. (In master mode, the NSS pin is pulled low in NSS hardware mode or SWNSS bit is low in NSS software mode.) This bit is set by hardware and cleared by a read or write operation on the SPI_STAT register followed by a write access to the SPI_CTL0 register. This bit is not used in I2S mode.
4	CRCERR	SPI CRC Error Bit 0: The SPI_RCRC value is equal to the received CRC data at last. 1: The SPI_RCRC value is not equal to the received CRC data at last. This bit is set by hardware and is able to be cleared by writing 0. This bit is not used in I2S mode.
3	TXURERR	Transmission underrun error bit 0: No transmission underrun error occurs. 1: Transmission underrun error occurs. This bit is set by hardware and cleared by a read operation on the SPI_STAT register. This bit is not used in SPI mode.
2	I2SCH	I2S channel side 0: The next data needs to be transmitted or the data just received is channel left. 1: The next data needs to be transmitted or the data just received is channel right. This bit is set and cleared by hardware. This bit is not used in SPI mode, and has no meaning in the I2S PCM mode.
1	TBE	Transmit Buffer Empty

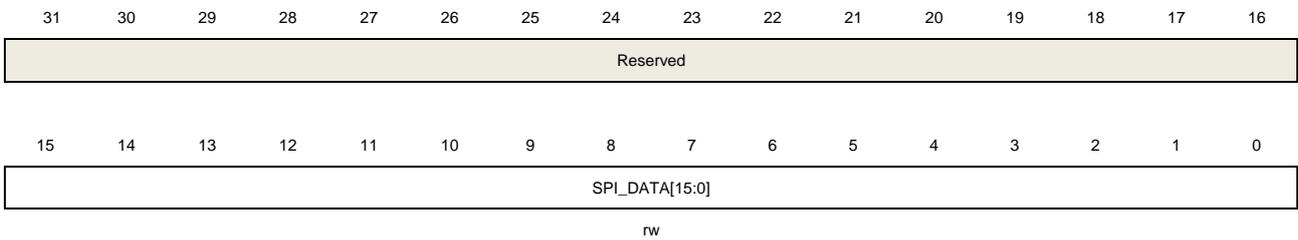
		0: Transmit buffer is not empty 1: Transmit buffer is empty
0	RBNE	Receive Buffer Not Empty 0: Receive buffer is empty 1: Receive buffer is not empty

### 19.11.4. Data register (SPI\_DATA)

Address offset: 0x0C

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit).



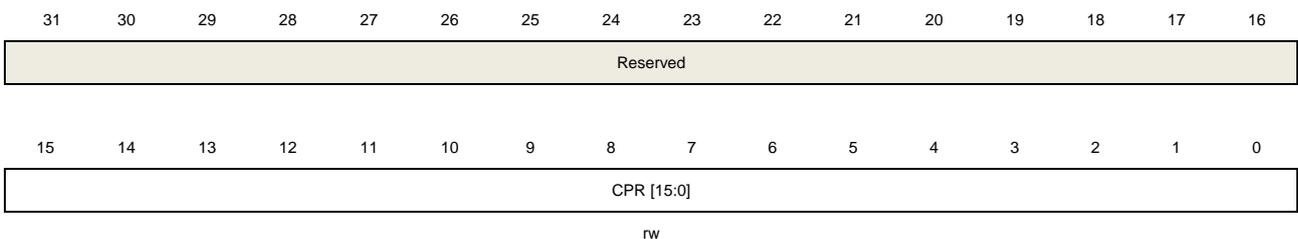
Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	SPI_DATA[15:0]	Data transfer register. The hardware has two buffers, including transmit buffer and receive buffer. Write data to SPI_DATA will save the data to transmit buffer and read data from SPI_DATA will get the data from receive buffer. When the data frame format is set to 8-bit data, the SPI_DATA [15:8] is forced to 0 and the SPI_DATA [7:0] is used for transmission and reception, transmit buffer and receive buffer are 8-bits. If the Data frame format is set to 16-bit data, the SPI_DATA [15:0] is used for transmission and reception, transmit buffer and receive buffer are 16-bit.

### 19.11.5. CRC polynomial register (SPI\_CRCPOLY)

Address offset: 0x10

Reset value: 0x0007

This register can be accessed by half-word (16-bit) or word (32-bit).



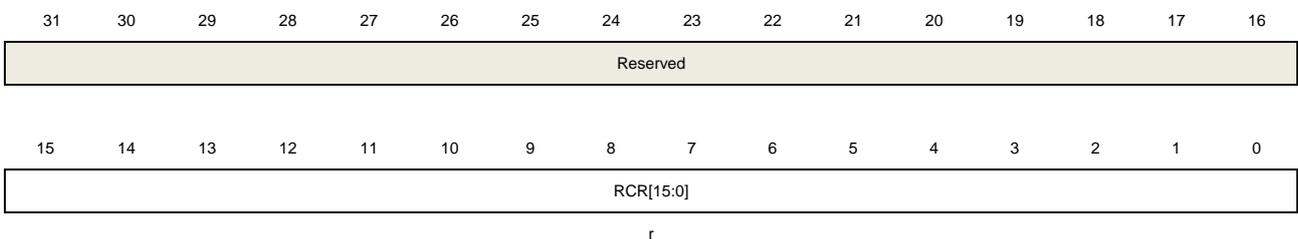
Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	CPR[15:0]	CRC polynomial register This register contains the CRC polynomial and it is used for CRC calculation. The default value is 0007h.

### 19.11.6. RX CRC register (SPI\_RCRC)

Address offset: 0x14

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit).



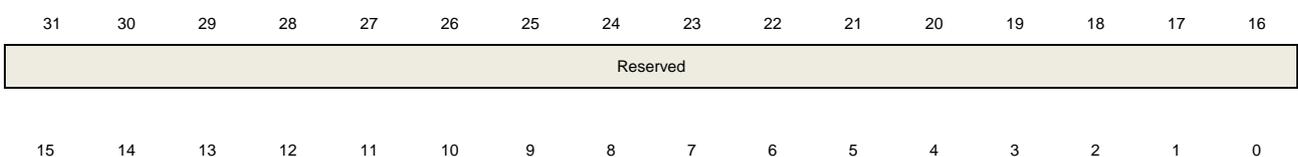
Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	RCR[15:0]	RX CRC register When the CRCERRN bit of SPI_CTL0 is set, the hardware computes the CRC value of the received bytes and saves them in RCR register. If the Data frame format is set to 8-bit data, CRC calculation is based on CRC8 standard, and saves the value in RCR [7:0], when the Data frame format is set to 16-bit data, CRC calculation is based on CRC16 standard, and saves the value in RCR[15:0]. The hardware computes the CRC value after each received bit, when the TRANS is set, a read to this register could return an intermediate value. This register is reset when the CRCEN bit in SPI_CTL0 register or the SPIxRST bit in RCU reset register is set.

### 19.11.7. TX CRC register (SPI\_TCRC)

Address offset: 0x18

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit).



TCR[15:0]

r

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	TCR[15:0]	<p>TX CRC register</p> <p>When the CRCEN bit of SPI_CTL0 is set, the hardware computes the CRC value of the transmitted bytes and saves them in TCR register. If the Data frame format is set to 8-bit data, CRC calculation is based on CRC8 standard, and saves the value in TCR [7:0], when the Data frame format is set to 16-bit data, CRC calculation is based on CRC16 standard, and saves the value in TCR [15:0].</p> <p>The hardware computes the CRC value after each transmitted bit, when the TRANS is set, a read to this register could return an intermediate value. The different frame format (LF bit of the SPI_CTL0) will get different CRC value.</p> <p>This register is reset when the CRCEN bit in SPI_CTL0 register or the SPIxRST bit in RCU reset register is set.</p>

### 19.11.8. I2S control register (SPI\_I2SCTL)

Address offset: 0x1C

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit).

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				I2SSEL	I2SEN	I2SOPMOD[1:0]	PCMSMO D	Reserved	I2SSTD[1:0]	CKPL	DTLEN[1:0]	CHLEN			
				rw	rw	rw	rw		rw	rw	rw	rw			

Bits	Fields	Descriptions
31:12	Reserved	Must be kept at reset value
11	I2SSEL	<p>I2S mode selection</p> <p>0: SPI mode</p> <p>1: I2S mode</p> <p>This bit should be configured when SPI mode or I2S mode is disabled.</p>
10	I2SEN	<p>I2S enable</p> <p>0: I2S is disabled</p> <p>1: I2S is enabled</p> <p>This bit is not used in SPI mode.</p>
9:8	I2SOPMOD[1:0]	I2S operation mode

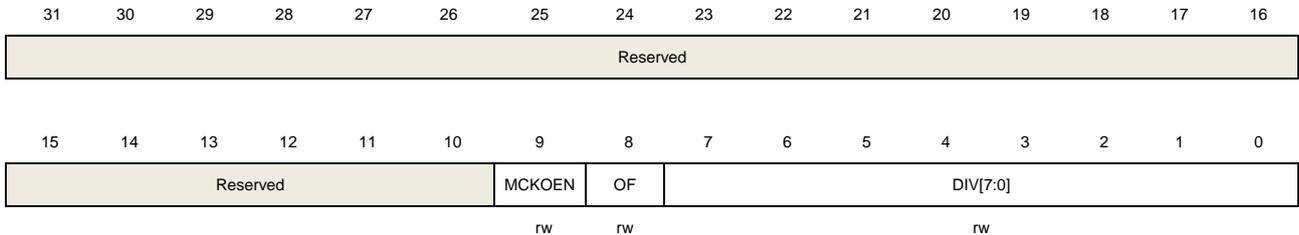
		00: Slave transmission mode
		01: Slave reception mode
		10: Master transmission mode
		11: Master reception mode
		This bit should be configured when I2S mode is disabled.
		This bit is not used in SPI mode.
7	PCMSMOD	PCM frame synchronization mode
		0: Short frame synchronization
		1: long frame synchronization
		This bit has a meaning only when PCM standard is used.
		This bit should be configured when I2S mode is disabled.
		This bit is not used in SPI mode.
6	Reserved	Must be kept at reset value
5:4	I2SSTD[1:0]	I2S standard selection
		00: I2S Phillips standard
		01: MSB justified standard
		10: LSB justified standard
		11: PCM standard
		These bits should be configured when I2S mode is disabled.
		These bits are not used in SPI mode.
3	CKPL	Idle state clock polarity
		0: The idle state of I2S_CK is low level
		1: The idle state of I2S_CK is high level
		This bit should be configured when I2S mode is disabled.
		This bit is not used in SPI mode.
2:1	DTLEN[1:0]	Data length
		00: 16 bits
		01: 24 bits
		10: 32 bits
		11: Reserved
		These bits should be configured when I2S mode is disabled.
		These bits are not used in SPI mode.
0	CHLEN	Channel length
		0: 16 bits
		1: 32 bits
		The channel length must be equal to or greater than the data length.
		This bit should be configured when I2S mode is disabled.
		This bit is not used in SPI mode.

### 19.11.9. I2S clock prescaler register (SPI\_I2SPSC)

Address offset: 0x20

Reset value: 0x0002

This register can be accessed by half-word (16-bit) or word (32-bit).



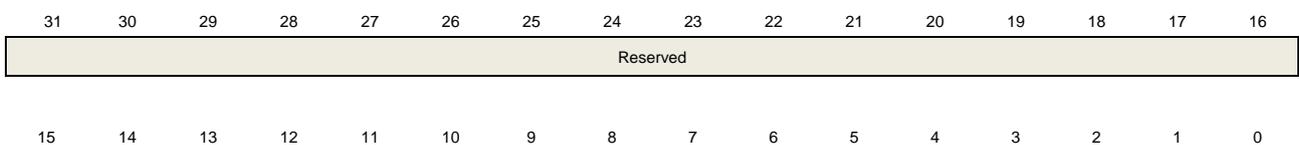
Bits	Fields	Descriptions
31:10	Reserved	Must be kept at reset value
9	MCKOEN	I2S_MCK output enable 0: I2S_MCK output is disabled 1: I2S_MCK output is enabled This bit should be configured when I2S mode is disabled. This bit is not used in SPI mode.
8	OF	Odd factor for the prescaler 0: Real divider value is DIV * 2 1: Real divider value is DIV * 2 + 1 This bit should be configured when I2S mode is disabled. This bit is not used in SPI mode.
7:0	DIV[7:0]	Dividing factor for the prescaler Real divider value is DIV * 2 + OF. DIV must not be 0. These bits should be configured when I2S mode is disabled. These bits are not used in SPI mode.

### 19.11.10. Quad-SPI mode control register (SPI\_QCTL) of GD32F170xx and GD32F190xx devices

Address offset: 0x80

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit).



Reserved	IO23_DR V	QRD	QMOD
	rw	rw	rw

Bits	Fields	Descriptions
31:3	Reserved	Must be kept at reset value
2	IO23_DRV	Drive IO2 and IO3 enable 0: IO2 and IO3 are not driven in single wire mode 1: IO2 and IO3 are driven to high in single wire mode This bit is only available in SPI1.
1	QRD	Quad-SPI mode read select. 0: SPI is in quad wire write mode 1: SPI is in quad wire read mode This bit should be only be configured when SPI is not busy (TRANS bit cleared) This bit is only available in SPI1.
0	QMOD	Quad-SPI mode enable. 0: SPI is in single wire mode 1: SPI is in Quad-SPI mode This bit should only be configured when SPI is not busy (TRANS bit cleared). This bit is only available in SPI1.

## 20. HDMI-CEC controller(HDMI-CEC)

### 20.1. Overview

Consumer Electronics Control (CEC) belongs to a part of HDMI (High-Definition Multimedia Interface) standard. CEC, as a kind of protocol, provides the advanced control functions of all kinds of audio-visual products in a user environment. The CEC protocol gets hardware support from the HDMI-CEC controller.

### 20.2. Characteristics

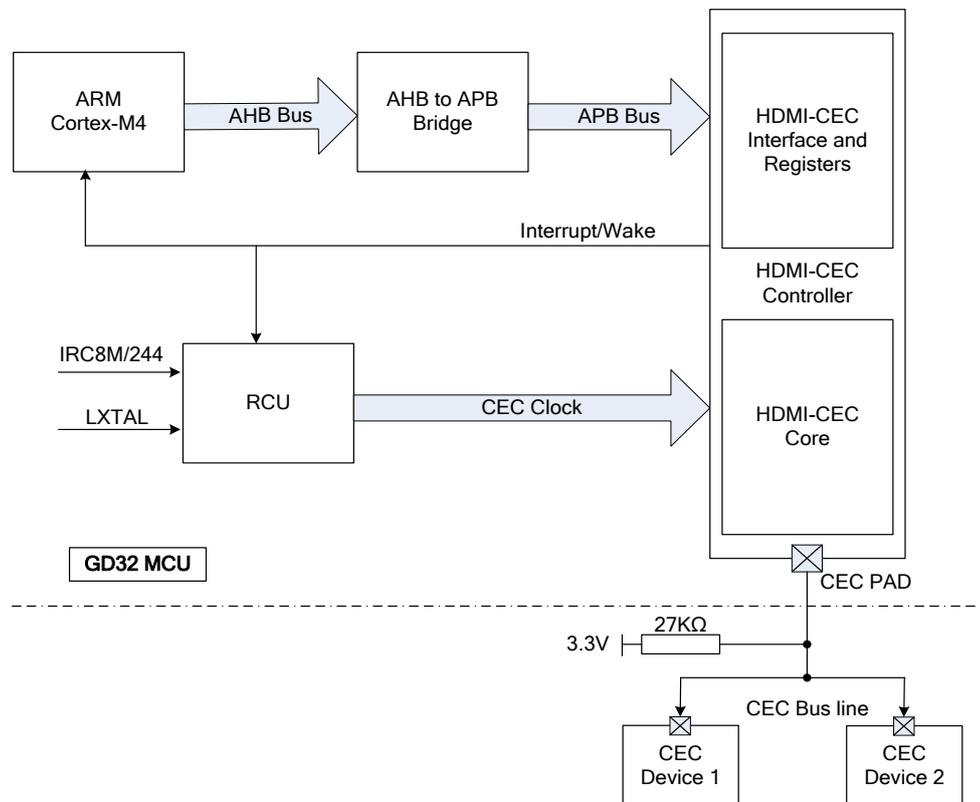
- HDMI-CEC controller complies with HDMI-CEC v1.4 Specification
- Two clock source options for 32.768KHz CEC clock:
  - 1) LXTAL oscillator
  - 2) IRC8M oscillator with settled prescaler (IRC8M/244)
- For ultra low-power applications, HDMI-CEC controller can work in Deep-sleep mode
- Programmable SFT(Signal Free Time) value for arbitration priority:
  - 1) User configure
  - 2) Auto configure by controller as HDMI-CEC protocol specification
- Programmable own address(OAD)
- Listen mode supports user receiving messages on the CEC line but not disturb the CEC line.
- Receive bit-tolerance function support for higher compatibility
- Bit Error Detection
  - 1) Bit period short error(BPSE)
  - 2) Bit period long error(BPLE)
  - 3) Bit rising error(BRE)
- Programmable error-bit generation
  - 1) BPSE detection will always generate error-bit
  - 2) BPLE detection will generate error-bit if BPLEG=1
  - 3) BRE detection will generate error-bit if BREG=1
- Transmission error detection (TERR)
- Transmission underrun detection (TU)
- Reception overrun detection (RO)
- Arbitration fail detection (ARBF), controller will automatic retry transmission if ARBF asserted.

## 20.3. Function overview

### 20.3.1. CEC bus pin

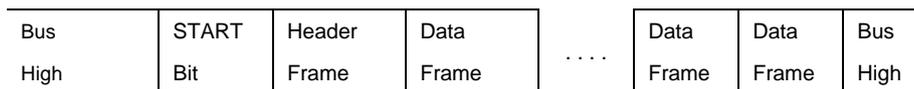
The CEC device communicates with others by only one bidirectional line. When the CEC device is in the output state, in order to allow a wired-and connection, the CEC pin need to be configured in alternate function open drain mode, and an external 27kΩ resistor is needed for pulling-up the CEC pin to a +3.3V supply voltage.

Figure 20-1. HDMI-CEC Controller Block Diagram



### 20.3.2. Message description

A complete message includes one or more frames and the message structure is shown as below:



The frame has two types:

- 1) **Header frame:** The first frame in the message which followed the start-bit consists of the source logical address field and the destination logical address field. The Header frame is always needed.
- 2) **Data frame:** The frames in the message followed the header frame. Data frame is optional.

All frames are ten bits long and have the same basic structure as shown below:

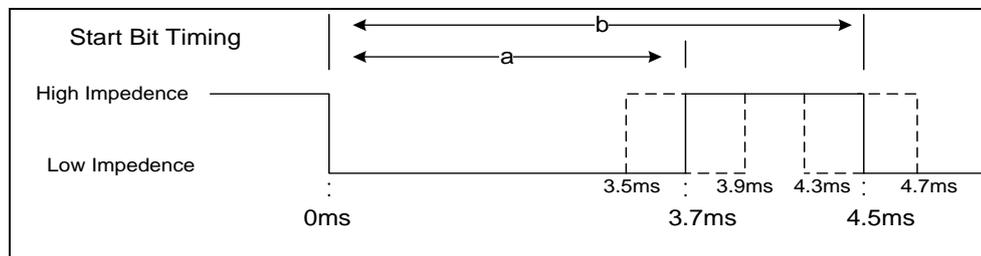
Frame Structure									
7	6	5	4	3	2	1	0		
Information bits								ENDOM	ACK

The information bits are data, opcodes or addresses, dependent on context. The control bits, ENDOM and ACK, are always present and always have the same usage.

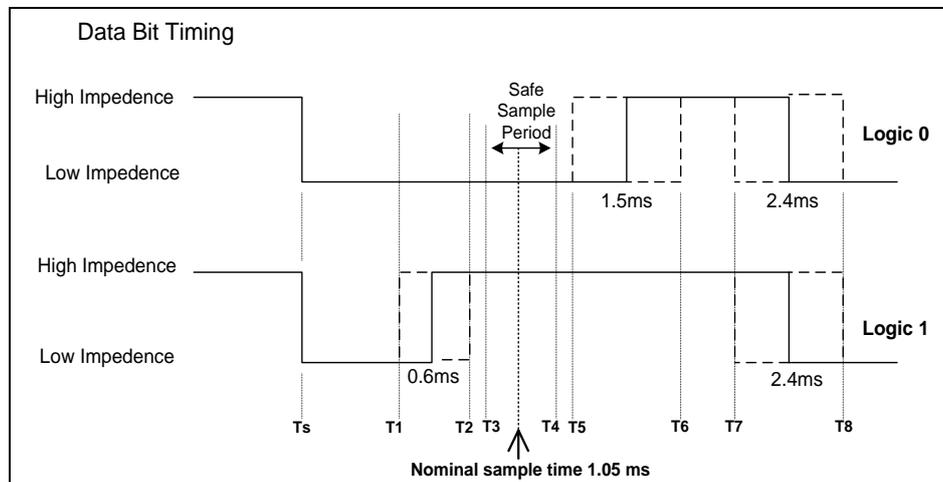
### 20.3.3. Bit timing description

All bits timing in the message are divided into two types: Start bit and Data bit.

1) Start Bit: The start bit has to be validated by its low duration(a) and its total duration(b) showed as below:



2) Data Bit: The valid data bit timing is constrained as below:



**Table 20-1. Data Bit Timing Parameter Table**

Ts	Time (ms)	The bit start event.
T1	0.4ms	When indicating a logical 1, T1 as the earliest time for a low - high transition.
T2	0.8ms	When indicating a logical 1, T2 as the latest time for a low - high transition.
T3	0.85ms	The earliest time it is safe to sample the signal line to determine its state.
T4	1.25ms	The latest time it is safe to sample the signal line to determine its state.
T5	1.3ms	T5 as the earliest time that a device is allowed to return to a high impedance state(logical 0).

Ts	Time (ms)	The bit start event.
T6	1.7ms	T6 as the latest time that a device is allowed to return to a high impedance state(logical 0).
T7	2.05ms	T7 as the earliest time for the start of a following bit.
	2.4ms	As a nominal data bit period.
T8	2.75ms	T8 as the latest time for the start of a following bit.

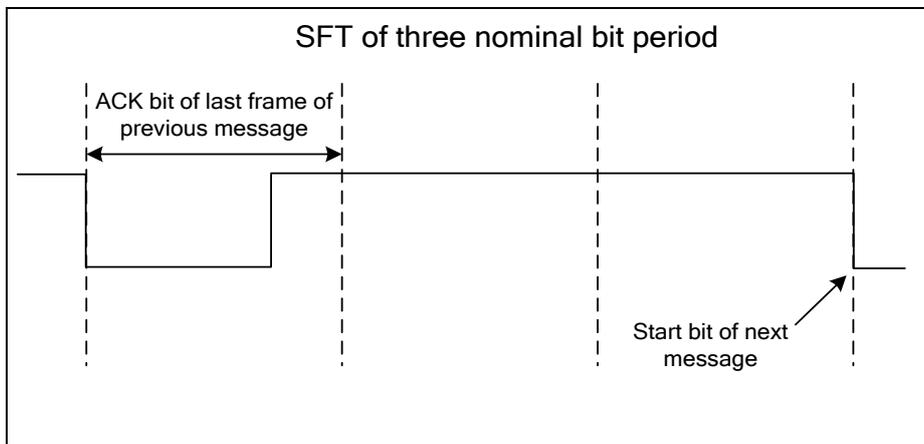
### 20.3.4. Arbitration

CEC line arbitration starts from the front edge of the start bit to the end of the Initiator address bits among the Header Block. In the meantime the Initiator should monitor the CEC line. During this period, if low impedance is detected when sending high impedance state then it should assume that it has lost the arbitration.



Before attempting to transmit or re-transmit a frame, a device shall ensure that the CEC line has been inactive for a number of bit periods.

This signal free time is defined as the time since the start of the final bit of the previous frame.



The length of the required signal free time depends on the current status of the control signal line and the initiating device. If SFT=0x0, the HDMI-CEC controller's SFT will perform as table below:

Precondition	Signal Free Time (nominal data bit periods)
Present Initiator wants to send another message immediately after its previous message	≥7
New Initiator wants to send a message	≥5
Previous attempt to send message unsuccessful	≥3

This means that there is an opportunity for other devices to gain access to the CEC line during the periods mentioned above to send their own messages after the current device has finished sending its current message.

If SFT is not 0x0, the corresponding user configure SFT will be performed.

### 20.3.5. SFT option bit description

SFT option bit support another way for saving bus inactive time through setting more SFT counter's start time point.

When SFTOPT = 0, the SFT timer will start at the time STAOM bit asserted when the controller is in the idle state.

When SFTOPT = 1, the SFT timer will start at the time CEC bus is in idle state and the SFT time will be saved if you configure the STAOM after SFT done because the controller will start transmit without any latency.

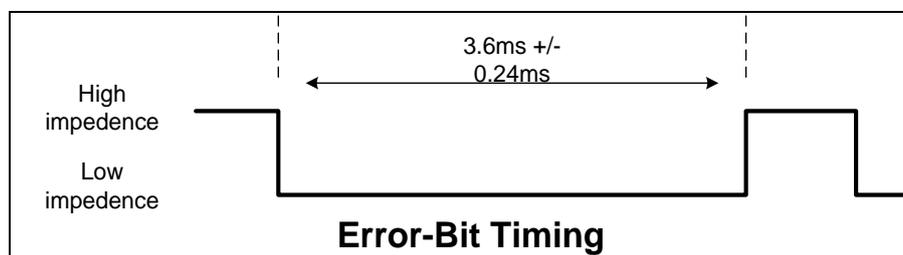
When SFTOPT = 1, some other cases will also start the SFT counter:

- In case of regular TX/RX complete(TEND/REND asserted)
- In case of transmission not complete such as the time TERR, TAERR or TU asserted.
- In case of receiving progress, if some error detected and error bit is generated, the SFT timer will start when output error bit finished.

### 20.3.6. Error definition

#### Error-Bit

If some errors are occurred and corresponding generation configure is enabled the HDMI-CEC controller will generate an error bit on the CEC pin for indicating. Error bit period definition is shown as below:



#### Frame error

CEC protocol defines that each frame of message need the acknowledgement to confirm the communication is successful. For broadcast(destination address=0xF), the ACK bit should be logic 1 and for singlecast(destination address<0xF), the ACK bit should be logic 0, otherwise

the frame error occurs(TAERR/RAE flag asserted).

Another frame error situation is that the CEC bus pin voltage is different from CEC pad when HDMI-CEC controller is under initiator state(TERR flag asserted).

### **Bit rising error(BRE)**

BRE flag can be asserted if rising edge detected during the BRE checking window and BRE flag will also generate CEC interrupt if the BREIE=1.

If BRES=1, the controller will stop receiving message and if BREG=1 the error-bit will be generated.

If BRES=1 in broadcast the BRE flag asserted, the error bit will be generated to notify the initiator the error. If you do not want to generate the error bit for BRE detection you can configure the BREG=0 and BCNG=1.

**Note:** The BRES=0 and BREG=1 configuration must be avoided.

### **Bit period short error(BPSE)**

BPSE is set when the period of the neighboring falling edge is shorter than expected. BPSE flag will also generate CEC interrupt if the BPSEIE=1.

If the BPSE flag asserted, the error bit will must be generated except one of the cases below:

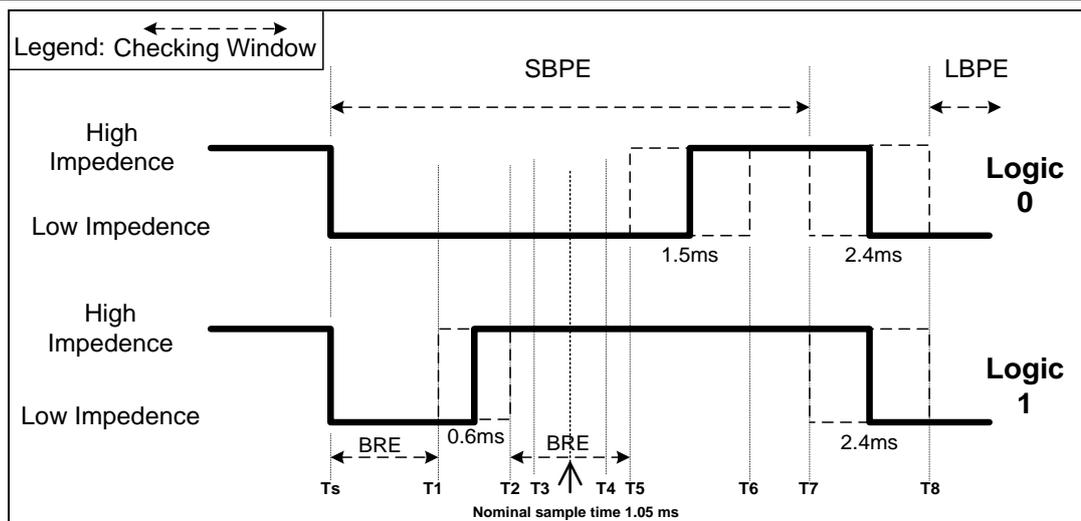
- 1) BCNG = 1
- 2) LMEN = 1
- 3) Receiving broadcast

### **Bit period long error(BPLE)**

BPLE is set when the period of the neighboring falling edge is longer than expected. BPLE flag will also generate CEC interrupt if the BPLEIE=1.

When BPLE asserted, controller will stop receiving message and generate error bit if in one of the cases below:

- 1) BPLEG=1 in both singlecast and broadcast
- 2) BCNG=0 in broadcast



**Table 20-2. Error Handling Timing Parameter Table**

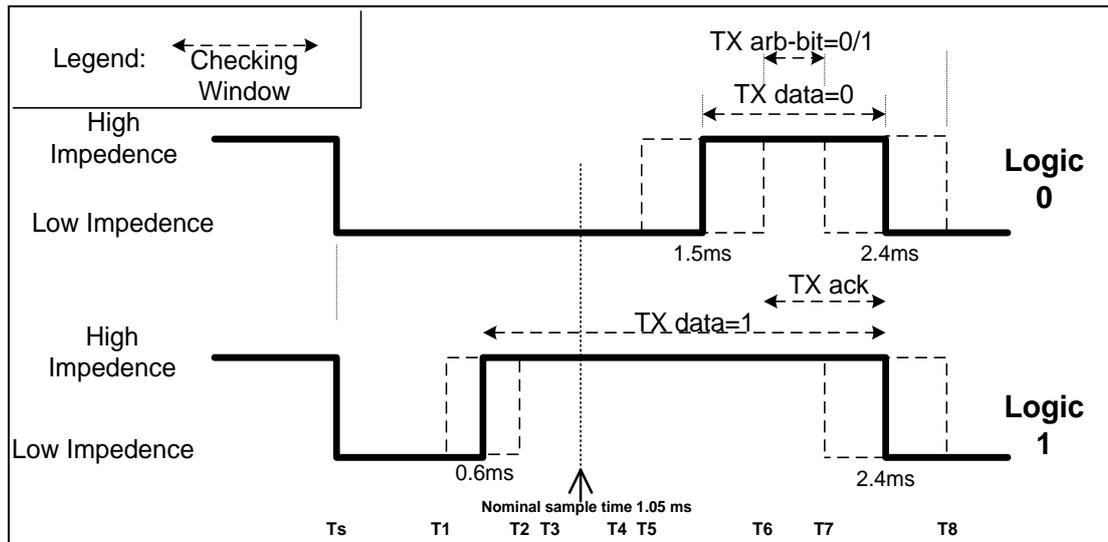
Symbol	RTOL	Time(ms)	Description
$T_s$	-	0ms	The bit start event.
$T_1$	1	0.3ms	When indicating a logical 1, $T_1$ as the earliest time for a low - high transition.
	0	0.4ms	
$T_2$	0	0.8ms	When indicating a logical 1, $T_2$ as the latest time for a low - high transition.
	1	0.9ms	
$T_3$	-	0.85ms	The earliest time it is safe to sample the signal line to determine its state.
$T_4$	-	1.25ms	The latest time it is safe to sample the signal line to determine its state.
$T_5$	1	1.2ms	$T_5$ as the earliest time that a device is allowed to return to a high impedance state(logical 0).
	0	1.3ms	
$T_6$	0	1.7ms	$T_6$ as the latest time that a device is allowed to return to a high impedance state(logical 0).
	1	1.8ms	
$T_7$	1	1.85ms	$T_7$ as the earliest time for the start of a following bit.
	0	2.05ms	
		2.4ms	As a nominal data bit period.
$T_8$	0	2.75ms	$T_8$ as the latest time for the start of a following bit.
	1	2.95ms	

### Transmission error detection(TERR)

The TERR is set when the initiator find low impedance on the CEC bus when it is transmitting high impedance. TERR will also generate CEC interrupt if TERRIE=1.

When TERR asserted the transmission is aborted and the software can retry the transmission.

TERR check window is depending on the different bit state of the frame shown as below:



**Table 20-3. TERR Timing Parameter Table**

Symbol	RTOL	Time(ms)	Description
Ts	-	0ms	The bit start event.
T1	1	0.3ms	The earliest time for a low - high transition when indicating a logical 1.
	0	0.4ms	
T2	0	0.8ms	The latest time for a low - high transition when indicating a logical 1.
	1	0.9ms	
T3	-	0.85ms	The earliest time it is safe to sample the signal line to determine its state.
T4	-	1.25ms	The latest time it is safe to sample the signal line to determine its state.
T5	1	1.2ms	The earliest time a device is permitted return to a high impedance state (logical 0).
	0	1.3ms	
T6	0	1.7ms	The latest time a device is permitted return to a high impedance state (logical 0).
	1	1.8ms	
T7	1	1.85ms	The earliest time for the start of a following bit.
	0	2.05ms	
		2.4ms	The nominal data bit period.
T8	0	2.75ms	The latest time for the start of a following bit.
	1	2.95ms	

### 20.3.7. HDMI-CEC interrupt

There 13 interrupts in HDMI-CEC controller are made up of corresponding flag and interrupt enable bit:

No.	Interrupt event in HDMI-CEC	Event flag	Interrupt enable bit
1	Arbitration fail	ARBF	ARBFIE

No.	Interrupt event in HDMI-CEC	Event flag	Interrupt enable bit
2	TX Byte Request	TBR	TBRIE
3	Transmission end	TEND	TENDIE
4	Transmit Byte buffer underrun	TU	TUIE
5	Transmit error	TERR	TERRIE
6	Transmit acknowledge error	TAERR	TAERRIE
7	Byte Received	BR	BRIE
8	Reception end	REND	RENDIE
9	Receive Overrun	RO	ROIE
10	Bit rising error	BRE	BREIE
11	Bit Period Short Error	BPSE	BPSEIE
12	Bit Period Long Error	BPLE	BPLEIE
13	RX acknowledge error	RAE	RAEIE

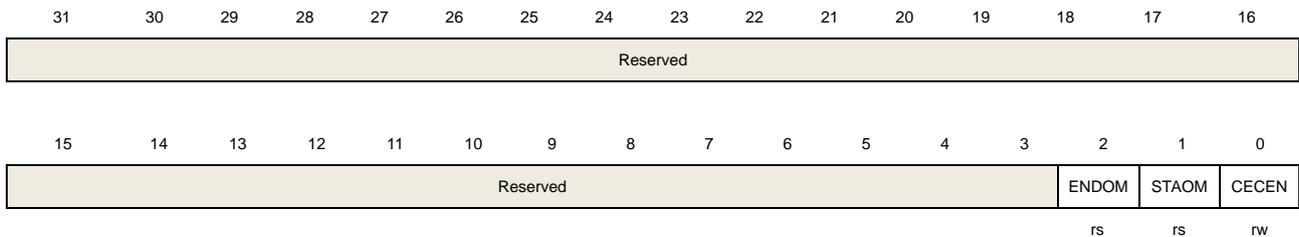
**Note:** Any HDMI-CEC interrupt will wake up the chip from Deep-sleep Mode.

## 20.4. Register definition

### 20.4.1. Control register (CEC\_CTL)

Address offset: 0x00

Reset value: 0x0000 0000



Bits	Fields	Descriptions
31:3	Reserved	Must be kept at reset value
2	ENDOM	<p>ENDOM bit value in the next frame in Transmit mode.</p> <p>ENDOM can only be set by software when CECEN=1. ENDOM is cleared by hardware in the same situation of clearing STAOM.</p> <p>0: Next frame send 0 in ENDOM bit position</p> <p>1: Next frame send 1 in ENDOM bit position</p>
1	STAOM	<p>Start of sending a message.</p> <p>STAOM can only be set by software when CECEN=1. STAOM is cleared by hardware if any of these flags asserted: TEND, TU, TAERR, TERR or CECEN is cleared.</p> <p>If the message consists of only header frame, ENDOM should be set before configuring header frame in TDATA. After setting STAOM, the SFT counter will start and when SFT is done the Start-bit will performance on CEC line. Software can abort sending the message through clearing CECEN bit under STAOM=1.</p> <p>0: No CEC transmission is on-going</p> <p>1: CEC transmission is pending or executing</p>
0	CECEN	<p>Enable/disable HDMI-CEC controller.</p> <p>CECEN bit is configured by software.</p> <p>0: Disable HDMI-CEC controller. Abort any sending message state and clear ENDOM/STAOM</p> <p>1: Enable CEC controller and go into receiving state if STAOM=0</p>

### 20.4.2. Configuration register (CEC\_CFG)

Address offset: 0x04

Reset value: 0x0000 0000

**Note:** This register can only be write when CECEN=0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LMEN		OAD [14:0]													
rw		rw													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							SFTOPT	BCNG	BPLEG	BREG	BRES	RTOL	SFT[2:0]		
							rw	rw	rw	rw	rw	rw	rw		

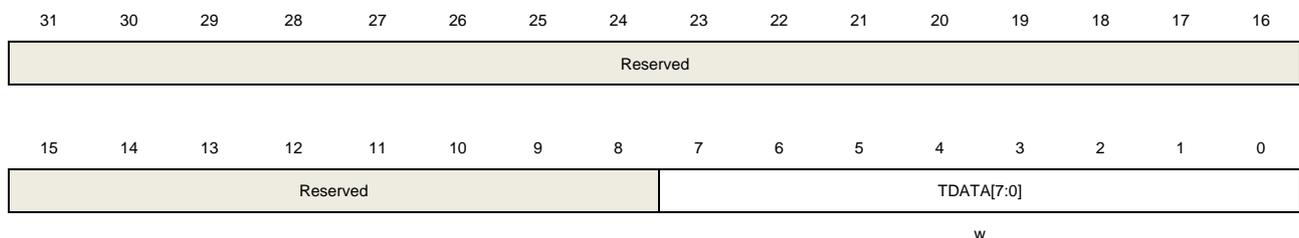
Bits	Fields	Descriptions
31	LMEN	<p>Listen Mode Enable Bit</p> <p>This bit is set and cleared by software.</p> <p>0: Only receive broadcast and singlecast in OAD address with appropriate ACK</p> <p>1: Receive broadcast and singlecast in OAD address with appropriate ACK and receive message whose destination address is not in OAD without feedback ACK</p>
30:16	OAD[14:0]	<p>Own Address</p> <p>Each bit of OAD represents one destination address. For example: if OAD[0]=1, the controller will receive the message being sent to address 0x0. This means the controller can be configured to have multiple own addresses. Broadcast message is always received.</p> <p>After received destination address(last 4 bit of HEADER frame),if it is valid in the OAD, the controller will feedback with positive acknowledge ,if it is not valid in the OAD and LMEN=1,the controller will receive the message with no acknowledge, if it is not valid in the OAD and LMEN=0,the controller will not receive the message.</p>
15:9	Reserved	Must be kept at reset value
8	SFTOPT	<p>The SFT start option bit</p> <p>This bit is set and cleared by software.</p> <p>0: SFT counter starts counting when STAOM is asserted</p> <p>1: SFT counter starts automatically after transmission/reception enabled</p>
7	BCNG	<p>Do not generate an Error-bit in broadcast message</p> <p>This bit is set and cleared by software.</p> <p>0: In broadcast mode, BRE and BPLE will generate an Error-bit on CEC line and if LMEN=1, BPSE will also generate an Error-bit</p> <p>1: Error-bit is not generated in the same condition as above</p>
6	BPLEG	<p>Generate an Error-bit when detected BPLE in singlecast</p> <p>This bit is set and cleared by software.</p> <p>0: Not generate an Error-bit on CEC line when detected BPLE in singlecast</p> <p>1: Generate an Error-bit on CEC line when detected BPLE in singlecast</p>
5	BREG	<p>Generate an Error-bit when detected BRE in singlecast</p> <p>This bit is set and cleared by software.</p> <p>0: Not generate an Error-bit on CEC line when detected BRE in singlecast</p> <p>1: Generate an Error-bit on CEC line when detected BRE in singlecast</p>

4	BRES	<p>Whether stop receive message when detected BRE</p> <p>This bit is set and cleared by software.</p> <p>0: Do not stop reception for BRE and data bit is sampled at nominal time(1.05ms)</p> <p>1: Stop reception for BRE</p>
3	RTOL	<p>Reception bit timing tolerance</p> <p>This bit is set and cleared by software.</p> <p>0: Standard bit timing tolerance</p> <p>1: Extended bit timing tolerance</p>
2:0	SFT[2:0]	<p>Signal Free Time</p> <p>This bit is set and cleared by software.</p> <p>If SFT=0x0, the SFT time will perform as HDMI-CEC protocol description and if not, the SFT time is fixed configured by software. The start point is the falling edge of the ACK bit.</p> <p>0x0:</p> <ul style="list-style-type: none"> <li>- 3x Standard data-bit period if SFT counter is start because of unsuccessful transmission(ARBF=1,TERR=1,TU=1 or TAERR=1)</li> <li>- 5x Standard data-bit period if CEC controller is the new initiator</li> <li>- 7x Standard data-bit period if CEC controller has successful completed transmission</li> </ul> <p>0x1: 1.5x nominal data bit periods</p> <p>0x2: 2.5x nominal data bit periods</p> <p>0x3: 3.5x nominal data bit periods</p> <p>0x4: 4.5x nominal data bit periods</p> <p>0x5: 5.5x nominal data bit periods</p> <p>0x6: 6.5x nominal data bit periods</p> <p>0x7: 7.5x nominal data bit periods</p>

### 20.4.3. Transmit data register (CEC\_TDATA)

Address offset: 0x08

Reset value: 0x0000 0000



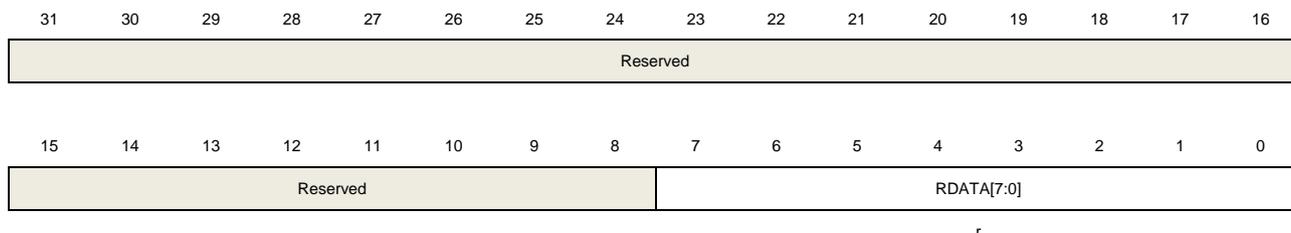
Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value
7:0	TDATA[7:0]	Transmit data register

These bits are write only and contain the data byte to be transmit.

## 20.4.4. Receive data register (CEC\_RDATA)

Address offset: 0xC

Reset value: 0x0000 0000

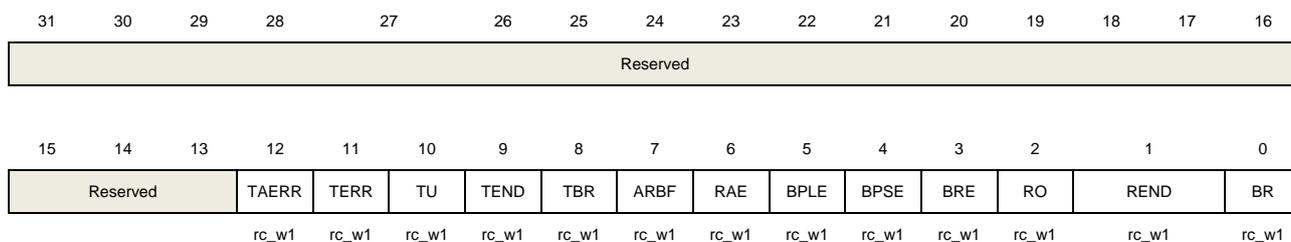


Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value
7:0	RDATA[7:0]	Receive data register These bits are read only and contain the last data byte which has been received from the CEC line.

## 20.4.5. Interrupt Flag Register (CEC\_INTF)

Address offset: 0x10

Reset value: 0x0000 0000



Bits	Fields	Descriptions
31:13	Reserved	Must be kept at reset value
12	TAERR	Transmit ACK Error flag. This bit is set by hardware and cleared by software writing 1. The ACK bit is received 1 in singlecast and is received 0 in broadcast will assert the flag. TAERR will stop sending message and clear STAOM and ENDOM.
11	TERR	Transmit Error This bit is set by hardware and cleared by software writing 1. TERR is asserted if the controller is in initiator state and the CEC line is low impedance but it does not pull it down. TERR will stop sending message and clear

STAOM and ENDOM.		
10	TU	<p>Transmit data buffer underrun</p> <p>This bit is set by hardware and cleared by software writing 1.</p> <p>TU is asserted if the software does not write data before sending the next byte. TU will stop sending message and clear STAOM and ENDOM.</p>
9	TEND	<p>Transmit successfully end</p> <p>This bit is set by hardware and cleared by software writing 1.</p> <p>TEND is asserted if the all frames of the message are successfully transmitted. TEND will clear STAOM and ENDOM bit.</p>
8	TBR	<p>Transmit Byte data request</p> <p>This bit is set by hardware and cleared by software writing 1.</p> <p>TBR is asserted when the 4th bit of current frame is transmitted and software should write data into TDATA within 6 nominal data-bit periods</p>
7	ARBF	<p>Arbitration fail</p> <p>This bit is set by hardware and cleared by software writing 1.</p> <p>ARBF is asserted when either situation is occurs: external CEC device pull down the CEC line for sending start bit when controller is in SFT state or the controller and CEC device sending the start bit at the same time but the controller's initiator address priority is lower.</p> <p>If ARBF is asserted, the controller will get into reception state and after finish receiving the message the controller will retry to send message. During receiving and sending message, the STAOM will keep set.</p>
6	RAE	<p>Receive ACK Error</p> <p>This bit is set by hardware and cleared by software writing 1.</p> <p>RAE is asserted if ACK=0 in broadcast or ACK=1 in singlecast under LMEN=1 and destination address is not in OAD.</p> <p>RAE will stop receiving message.</p>
5	BPLE	<p>Bit Period Long Error</p> <p>This bit is set by hardware and cleared by software writing 1.</p> <p>BPLE is asserted when the data-bit is out of the maximum period. BPLE will stop receiving message and generate an error-bit if BPLEG=1 in singlecast or BCNG=0 in broadcast.</p>
4	BPSE	<p>Bit Period Short Error</p> <p>This bit is set by hardware and cleared by software writing 1.</p> <p>BPSE is asserted if a data-bit period is less than the minimal period.</p>
3	BRE	<p>Bit Rising Error</p> <p>This bit is set by hardware and cleared by software writing 1.</p> <p>BRE is asserted if the rising edge in a period is occurs in unexpected time.</p>
2	RO	<p>Receive Overrun</p>

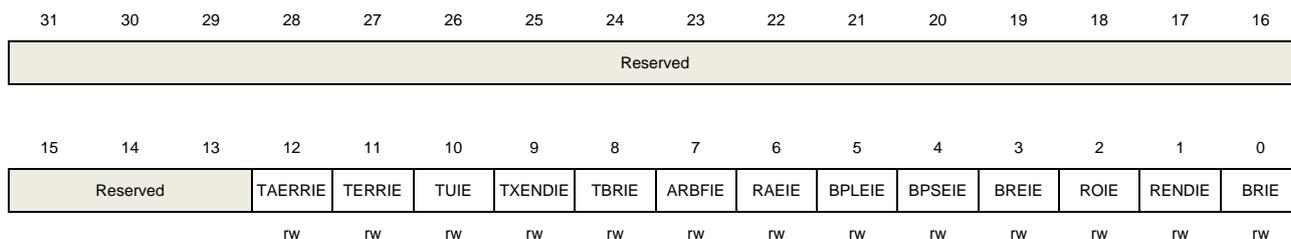
This bit is set by hardware and cleared by software writing 1.  
 RO is asserted when a new byte is received and BR is also set.  
 RO will stop receiving message and send an incorrect ACK bit.

- 1      **REND**      End of Reception  
 This bit is set by hardware and cleared by software writing 1.  
 REND is asserted if the controller received the whole message with feedback correct ACK. REND is asserted at the same time of BR.
- 0      **BR**      Byte received  
 This bit is set by hardware and cleared by software writing 1.  
 BR is asserted if the controller received the whole message with feedback correct ACK. When BR is asserted, the RDATA is valid.

## 20.4.6. Interrupt enable register (CEC\_INTEN)

Address offset: 0x14

Reset value: 0x0000 0000



Bits	Fields	Descriptions
31:13	Reserved	Must be kept at reset value
12	TAERRIE	TAERR Interrupt Enable. This bit is set and cleared by software. 0: TAERR interrupt disable 1: TAERR interrupt enable
11	TERRIE	TERR Interrupt Enable. This bit is set and cleared by software. 0: TERR interrupt disable 1: TERR interrupt enable
10	TUIE	TU Interrupt Enable. This bit is set and cleared by software. 0: TU interrupt disable 1: TU interrupt enable
9	TENDIE	TEND Interrupt Enable. This bit is set and cleared by software.

		0: TEND interrupt disable 1: TEND interrupt enable
8	TBRIE	TBR Interrupt Enable. This bit is set and cleared by software. 0: TBR interrupt disable 1: TBR interrupt enable
7	ARBFIE	ARBF Interrupt Enable. This bit is set and cleared by software. 0: ARBF interrupt disable 1: ARBF interrupt enable
6	RAEIE	RAE Interrupt Enable. This bit is set and cleared by software. 0: RAE interrupt disable 1: RAE interrupt enable
5	BPLEIE	BPLE Interrupt Enable. This bit is set and cleared by software. 0: BPLE interrupt disable 1: BPLE interrupt enable
4	BPSEIE	BPSE Interrupt Enable. This bit is set and cleared by software. 0: BPSE interrupt disable 1: BPSE interrupt enable
3	BREIE	BRE Interrupt Enable. This bit is set and cleared by software. 0: BRE interrupt disable 1: BRE interrupt enable
2	ROIE	RO Interrupt Enable. This bit is set and cleared by software. 0: RO interrupt disable 1: RO interrupt enable
1	RENDIE	REND Interrupt Enable. This bit is set and cleared by software. 0: REND interrupt disable 1: REND interrupt enable
0	BRIE	BR Interrupt Enable. This bit is set and cleared by software. 0: BR interrupt disable 1: BR interrupt enable



## 21. Touch sensing interface(TSI)

### 21.1. Overview

Touch Sensing Interface (TSI) provides a convenient solution for touch keys, sliders and capacitive proximity sensing applications. The controller builds on charge transfer method. Placing a finger near fringing electric fields adds capacitance to the system and TSI is able to measure this capacitance change using charge transfer method.

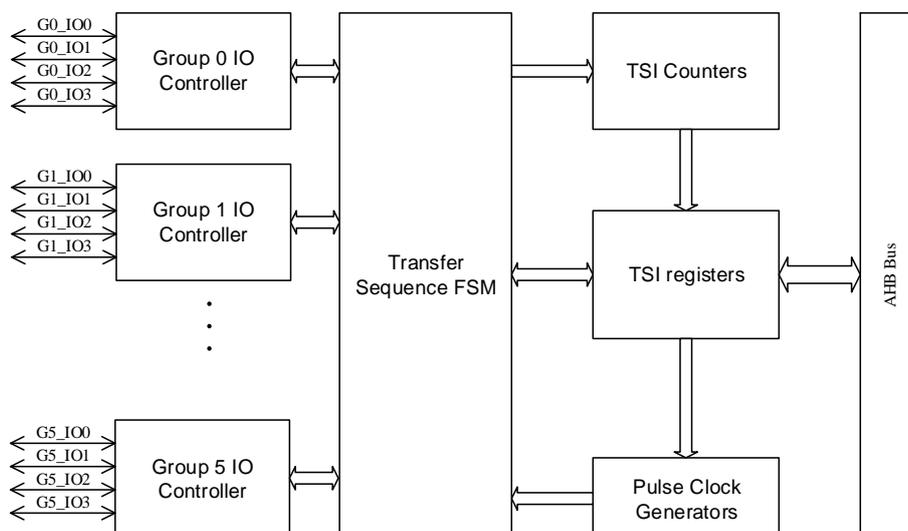
### 21.2. Characteristics

- Transfer sequence fully controlled by hardware
- 6 fully parallel groups implemented
- 18 IOs configurable for capacitive sensing Channel Pins and 6 for Sample Pins
- Configurable transfer sequence frequency
- Possible to implement the user specific transfer sequences
- Sequence end and error flags / configurable interrupts
- Spread spectrum function implemented

### 21.3. Function overview

#### 21.3.1. TSI block diagram

Figure 21-1. Block diagram of TSI module

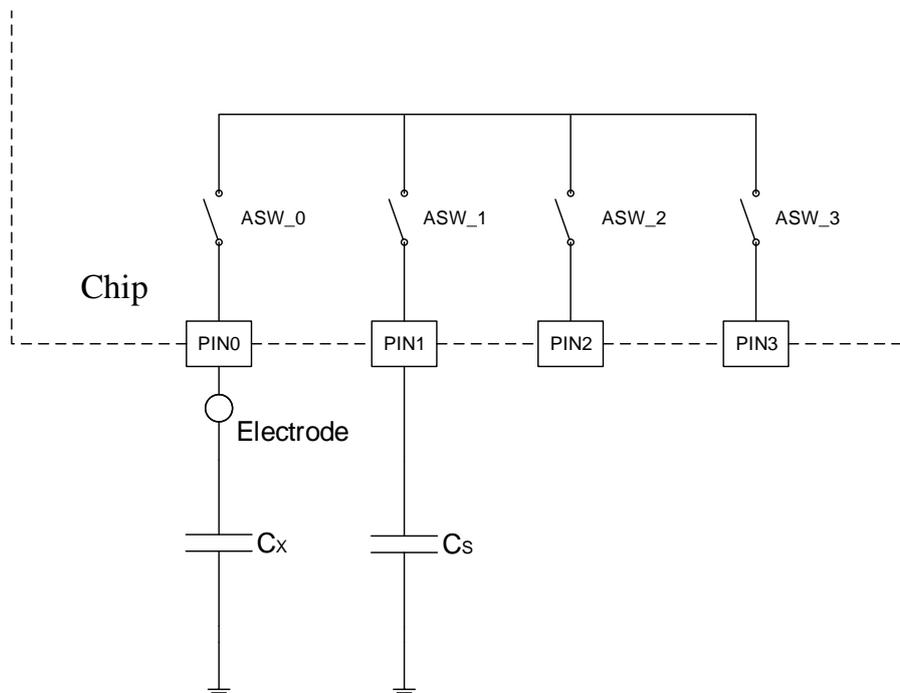


### 21.3.2. Touch sensing technique overview

There are different technologies for touch sensing, such as optical, resistive, capacitive, strain, etc. Detecting the change of a system is the key problem and goal in these technologies. The TSI module is designed to use charge transfer method which detects the capacitive change of an electrode when touched by or a finger close to it. In order to detect the capacitive change, TSI performs a charge transfer sequence including several charging, transfer steps. The number of these steps indicates the capacitance of an electrode. So the application is able to detect the change of capacitance by monitoring the step number of each transfer sequence.

As shown in Figure 21-2, there are 4 PINs in one group and each PIN has an analog switch connected to a common point which is the key component to implement charge transfer. There should be a sample pin and one or more channel pin(s) configured in one group. In Figure 21-2, PIN0 is a channel pin and PIN1 is a sample pin while PIN2 and PIN3 are unused. An electrode connecting PIN0 is designed on PCB board. The A sample capacitor  $C_s$  connected to sample pin PIN1 is also required. Now the capacitance of the channel pin PIN0 includes  $C_x$  and the capacitance introduced by the electrode, so capacitance of PIN0 increases when a finger is touching while the capacitance of PIN1 remains unchanged. Thus, the finger's touching can be detected if the capacitance of PIN0 can be measured. In TSI module, a charge-transfer sequence is performed to measure the capacitance of the channel pin(s) in a group, which will be detailed in next section.

**Figure 21-2. Block diagram of Sample pin and Channel Pin**



### 21.3.3. Charge transfer sequence

In order to measure the capacitance of a channel pin, charge transfer sequence is performed in chip. The sequence shown in Table 21-1 is described based on the connection of Figure

21-2, i.e. PIN0 is channel pin and PIN1 is sample pin.

**Table 21-1. Pin and analog switch state in a charge-transfer sequence**

Step	Name	ASW_0	ASW_1	Pin0	Pin1
1	Discharge	Close	Close	Input Floating	Pull Down
2	Buffer Time1	Open	Open	Input Floating	Input Floating
3	Charge	Open	Open	Output High	Input Floating
4	Extend Charge	Open	Open	Output High	Input Floating
5	Buffer Time2	Open	Open	Input Floating	Input Floating
6	Charge Transfer	Close	Close	Input Floating	Input Floating
7	Buffer Time3	Open	Open	Input Floating	Input Floating
8	Compare	Open	Open	Input Floating	Input Floating

### 1. Discharge

Both  $C_x$  and  $C_s$  are discharged by closing ASW\_0 and ASW\_1 and configuring PIN1 to pull down. This step is the initial operation for a correct charge transfer sequence and is performed by software before starting a charge transfer sequence. Discharging time in this step should be guaranteed to ensure that the voltage of  $C_x$  and  $C_s$  are discharged to zero.

### 2. Buffer Time1

Buffer time with ASW\_0 and ASW\_1 open, PIN0 is configured to input floating.

### 3. Charge

Channel pin PIN0 is configured to output high, in order to charge  $C_x$ . ASW\_0 and ASW\_1 remain open during this step. The charging time should be configured (see Register Section for detail) to ensure that the voltage of  $C_x$  is charged to  $V_{DD}$ .

### 4. Extend Charge

This is an optional step in a charge-transfer sequence and the behavior of all pins and analog switches in this step is the same as Step 3. The only difference between this and step 3 is the duration time, which is configurable in TSI registers. The duration of this step changes in each loop of a charge-transfer sequence, spreading the spectrum.

**5. Buffer Time2**

Buffer time with ASW\_0 and ASW\_1 open, PIN0 is configured to input floating.

**6. Charge transfer**

ASW\_0 and ASW\_1 are closed and PIN0 is configured to input floating to transfer charge from  $C_x$  to  $C_s$ . The transfer time should be configured (see Register Section for detail) to ensure the full transfer after that the voltage of  $C_x$  and  $C_s$  will be equal.

**7. Buffer Time3**

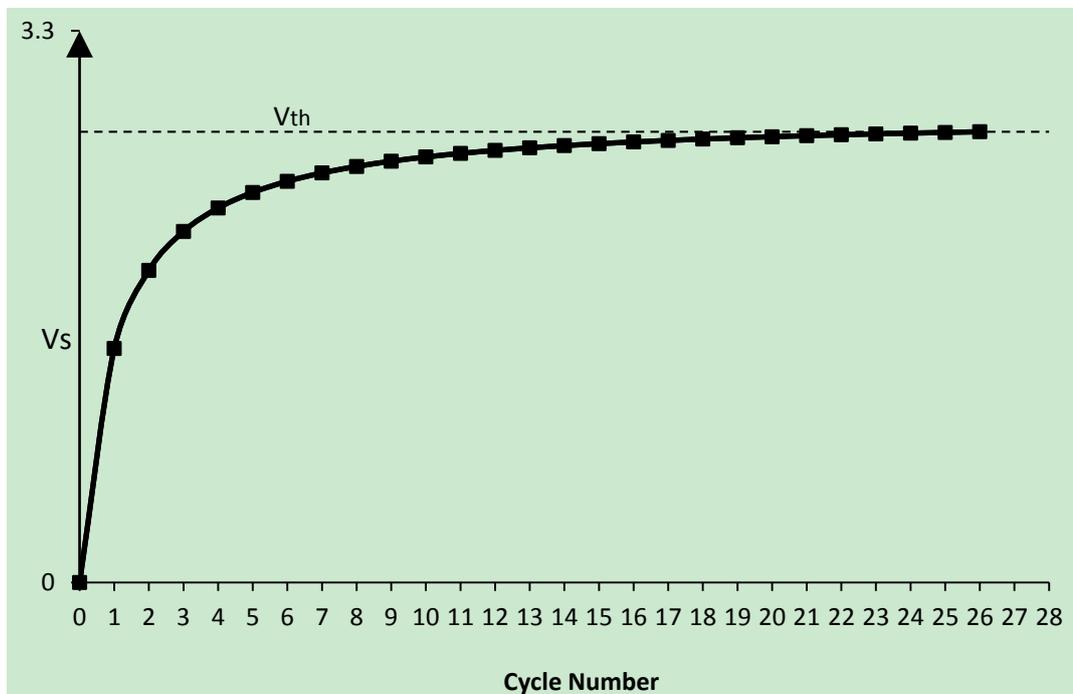
Buffer time with ASW\_0 and ASW\_1 open, PIN0 is configured to input floating.

**8. Compare**

ASW\_0, ASW\_1 and PIN0 remain the configuration of Step7. At this step, the voltage of sample pin PIN1 is compared to a threshold called  $V_{th}$ . If voltage of PIN1 is lower than  $V_{th}$ , the sequence returns to Step2 and continues, otherwise, the sequence ends.

The voltage of sample pin  $V_s$  is zero after initial step and increases after each charge cycle, as shown in Figure 21-3. A larger  $C_x$  will cause a greater increase during a cycle. The sequence stops when  $V_s$  reaches  $V_{th}$ . Each group has a counter which records the number of cycles performed on it to reach  $V_{th}$ . At the end of charge-transfer sequence, the group counter is read out to estimate the  $C_x$ , i.e. a smaller counter values indicates a larger  $C_x$ .

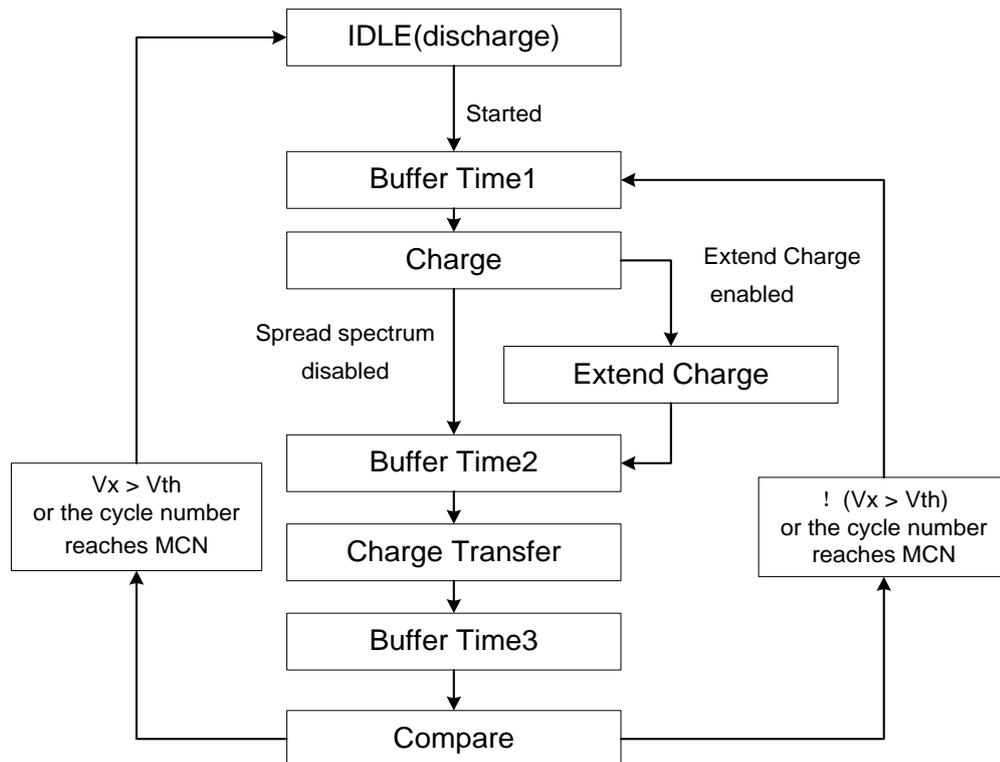
**Figure 21-3. Voltage of a sample pin during charge-transfer sequence**



### 21.3.4. Charge transfer sequence FSM

A hardware FSM is designed in chip to perform the charge transfer sequence described in the previous section as shown in Figure 21-4.

Figure 21-4. FSM flow of a charge-transfer sequence



This FSM remains in IDLE state after reset. There are 2 kinds of start condition defined by TRGMOD bit in TSI\_CTL register:

**TRGMOD = 0:** Software Trigger Mode. In this mode, the FSM starts after TSIS bit in TSI\_CTL register is written 1 by software.

**TRGMOD = 1:** Hardware Trigger Mode. In this mode the FSM starts when a falling or rising edge on TSITG pin is detected.

Once started, the FSM runs following the flow described in Figure 21-4. The FSM leaves a state if the duration time of this state reaches defined value, and goes into the next state.

The Extend Charge state is present only if the ECEN bit is set in TSI\_CTL register. This state is designed to implement spread spectrum function.

In comparing state, the FSM compares voltage of the sample pin in every enabled group and the threshold voltage. If all sample pins' voltage reach the threshold, FSM returns IDLE state and stops, otherwise, it returns to Buffer Time 1 state and begins the next cycle.

As shown in Figure 21-4, after 27 cycles,  $V_s$  (the voltage of sample pin) reaches  $V_{th}$  (the threshold voltage).

There is also a max cycle number defined by MCN in TSI\_CTL register. When the cycle number reaches MCN, FSM returns to IDLE state and stops after Compare State, whether  $V_s$  reaches  $V_{th}$  or not.

### 21.3.5. Clock and duration time of states

There are 3 clocks in TSI module: HCLK, CTCLK and ECCLK. HCLK is system clock and drives TSI's register and FSM. CTCLK, which is divided from HCLK with division factor defined by register CTCDIV is the clock used for calculating the duration time of the charge state and Charge Transfer state. ECCLK, which is divided from HCLK with division factor defined by register ECCDIV is the clock used to calculate the maximum duration time of Extend Charge state.

The duration time of each state except state Extend Charge state is fixed in each loop according to the configuration of the register.

The duration time of Buffer Time1, Buffer Time2 and Buffer Time3 are fixed to 2 HCLK periods. The duration time of Charge state and Charge Transfer state is defined by CDT and CTDT bits (see TSC\_CTL register section for detail).

The duration time of Extend Charge state changes in each cycle of the charge-transfer FSM and the maximum duration time are defined by ECDDT[6:0] in TSI\_CTL register.

The duration time of Extend Charge state in each cycle is presented in Table 21-2.

**Table 21-2. Duration time of Extend Charge state in each cycle**

Cycle Number	Number of ECCLKs in Extend Charge state
1	0
2	1
...	
ECDDT	ECDDT-1
ECDDT+1	ECDDT
ECDDT+2	ECDDT+1
ECDDT+3	ECDDT
ECDDT+4	ECDDT-1
...	...
2*ECDDT+1	2
2*ECDDT+2	1
2*ECDDT+3	0
2*ECDDT+4	1
2*ECDDT+5	2
...	...

### 21.3.6. PIN mode control of TSI

There are 4 pins in each group and each of these pins is able to be used as a sample pin or channel pin. Only one pin in a group should be configured as sample pin, and channel pins can be more than one. The sample pin and channel pin(s) should not be configured as the same pin in any case.

When a PIN is configured in GPIO (see chapter GPIO) used by TSI, the pin's mode is controlled by TSI. Generally, each pin has 3 modes: input, output high and output low.

The mode of a channel pin or a sample pin during a charge-transfer sequence is described in Table 21-1 in which PIN0 represents a channel pin and PIN1 represents a sample pin, i.e. the charge-transfer FSM take control of these channels or sample pins' mode and the states of related analog switches when the sequence is on-going. When the sequence is in IDLE state, PINMOD bit in TSI\_CTL register defines the mode of these pins. Pins that are configured in GPIO used by TSI but neither sample nor channel in TSI register is called free pins whose mode is defined by PINMOD bit in TSI\_CTL, too.

### 21.3.7. ASW and hysteresis mode

A channel or sample pin's analog switch is controlled by charge-transfer sequence when FSM is running, as shown in Table 21-1. When the FSM is IDLE, these pins' analog switches are controlled by GxPy bits in TSI\_ASW register. All free pin's analog switches are controlled by GxPy bits too.

TSI takes control of the analog switches when FSM is IDLE, even if these pins are not configured to be used by TSI in GPIO. The user is able to perform user-defined charge-transfer sequence by writing GxPy bits to control these analog switches, while controlling pin mode directly in GPIO.

Each TSI pin's hysteresis mode is configurable by GxPy bit in TSI\_PHM register.

### 21.3.8. TSI operation flow

The normal operation flow of TSI is listed below:

System initialization, such as system clock configuration, TSI related GPIO configuration, etc. Program TSI\_CTL, TSI\_CHCFG, TSI\_INTEN, TSI\_SAMP\_CFG and GEx bits of TSI\_GCTL register according to demand.

Enable TSI by setting TSIEN bit in TSI\_CTL register.

Optional for software trigger mode: program TSIEN bit to start charging transfer sequence. In hardware trigger mode, TSI is started by falling/rising edge on the trigger pin.

Wait for the CTCF or MNERR flag in TSI\_INTF and clear these flags by writing TSI\_INTC.

Read out the CYCN bits in TSI\_GxCYCN registers.

### 21.3.9. TSI flags and interrupts

**Table 21-3. TSI errors and flags**

Flag Name	Description	Cleared by
-----------	-------------	------------

CTCF	TSI stops because all enabled samplers' sample pins reach $V_{th}$ .	CCTCF bit in TSI_INTC
MNERR	TSI stops because the cycle number reaches the maximum value.	CMNERR bit in TSI_INTC

## 21.3.10. TSI GPIOs

**Table 21-4. TSI pins**

TSI Group	TSI Pins	GPPIN pins
TSI_GRP0	PIN0	PA0
	PIN1	PA1
	PIN2	PA2
	PIN3	PA3
TSI_GRP1	PIN0	PA4
	PIN1	PA5
	PIN2	PA6
	PIN3	PA7
TSI_GRP2	PIN0	PC5
	PIN1	PB0
	PIN2	PB1
	PIN3	PB2
TSI_GRP3	PIN0	PA9
	PIN1	PA10
	PIN2	PA11
	PIN3	PA12
TSI_GRP4	PIN0	PB3
	PIN1	PB4
	PIN2	PB6
	PIN3	PB7
TSI_GRP5	PIN0	PB11
	PIN1	PB12
	PIN2	PB13
	PIN3	PB14

## 21.4. Register definition

### 21.4.1. Control register (TSI\_CTL)

Address offset: 0x00

Reset value: 0x0000 0000

This register can be accessed by word(32-bit).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CDT[3:0]				CTDT[3:0]				ECDT[6:0]						ECEN	
rw				rw				rw						rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ECDIV	CTCDIV[2:0]			Reserved				MCN[2:0]		PINMOD	EGSEL	TRGMO D	TSIS	TSIEN	
rw	rw							rw		rw	rw	rw	rw	rw	

Bits	Fields	Descriptions
31:28	CDT[3:0]	Charge State Duration Time CDT[3:0] is set and clear by software. These bits controls the duration time of Charge State in a charge-transfer sequence. 0000: $1 \times t_{CTCLK}$ 0001: $2 \times t_{CTCLK}$ 0010: $3 \times t_{CTCLK}$ .... 1111: $16 \times t_{CTCLK}$
27:24	CTDT[3:0]	Charge Transfer State Duration Time CTDT[3:0] is set and clear by software. These bits control the duration time of Charge Transfer State in a charge-transfer sequence. 0000: $1 \times t_{CTCLK}$ 0001: $2 \times t_{CTCLK}$ 0010: $3 \times t_{CTCLK}$ .... 1111: $16 \times t_{CTCLK}$
23:17	ECDT[6:0]	Extend Charge State Maximum Duration Time ECDT[6:0] is set and clear by software. These bits control the maximum Duration Time duration time of Extend Charge Transfer State in a charge-transfer sequence. Extend Charge State is only present when ECEN bit in TSI_CTL register is set. 0000000: $1 \times t_{ECCLK}$ 0000001: $2 \times t_{ECCLK}$ 0000010: $3 \times t_{ECCLK}$

		....
		11111111: $128 \times t_{ECCLK}$
16	ECEN	Extend Charge State Enable. 0: Extend Charge disabled 1: Extend Charge enabled
15	ECDIV	ECCLK clock division factor. ECCLK in TSI is divided from HCLK and ECDIV defines the division factor. 0: $f_{ECCLK} = f_{HCLK}$ 1: $f_{ECCLK} = 0.5f_{HCLK}$
14:12	CTCDIV[2:0]	CTCLK clock division factor. CTCLK in TSI is divided from HCLK and CTCDIV defines the division factor. 000: $f_{CTCLK} = f_{HCLK}$ 001: $f_{CTCLK} = f_{HCLK}/2$ 010: $f_{CTCLK} = f_{HCLK}/4$ 011: $f_{CTCLK} = f_{HCLK}/8$ .... 111: $f_{CTCLK} = f_{HCLK}/128$
11:8	Reserved	Must be kept at reset value
7:5	MCN[2:0]	Max cycle number of a sequence MCN [2:0] defines the max cycle number of a charge-transfer sequence FSM which stops after reaching this number. 000: 255 001: 511 010: 1023 011: 2047 100: 4095 101: 8191 110: 16383 111: Reserved
4	PINMOD	Pin mode This bit defines a TSI pin's mode when charge-transfer sequence is IDLE. 0: TSI pin will output low when IDLE 1: TSI pin will keep input mode when IDLE
3	EGSEL	Edge selection This bit defines the edge type in hardware trigger mode. 0: Falling edge 1: Rising edge
2	TRGMOD	Trigger mode selection 0: Software Trigger Mode, sequence will start after TSIS bit is set. 1: Hardware Trigger Mode, sequence will start after a falling/rising edge on trigger

pin detected.

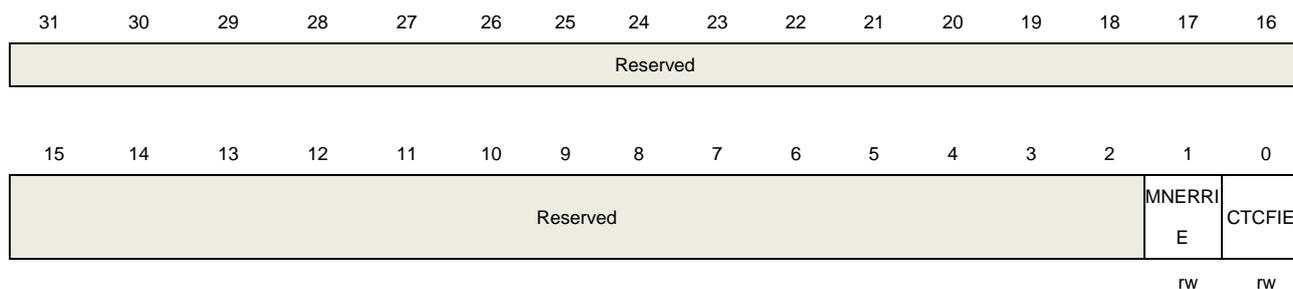
1	TSIS	<p>TSI start</p> <p>This bit is set by software to start a charge-transfer sequence in software trigger mode and reset by hardware when the sequence stops. After setting this bit, software can reset it to stop the started sequence manually.</p> <p>0: TSI is not started 1: TSI is started.</p>
0	TSIEN	<p>TSI enable</p> <p>0: TSI module is enabled 1: TSI module is disabled</p>

## 21.4.2. Interrupt enable register (TSI\_INTEN)

Address offset: 0x04

Reset value: 0x0000 0000

This register can be accessed by word(32-bit).



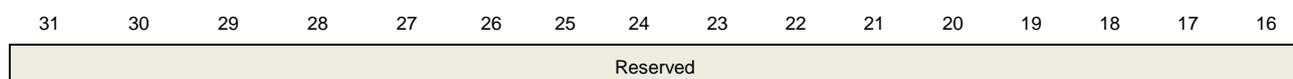
Bits	Fields	Descriptions
31:2	Reserved	Must be kept at reset value
1	MNERRIE	<p>Max Cycle Number Error Interrupt Enable</p> <p>0: MNERR interrupt is disabled 1: MNERR interrupt is enabled</p>
0	CTCFIE	<p>Charge-transfer complete flag Interrupt Enable</p> <p>0: CTCF interrupt is disabled 1: CTCF interrupt is enabled</p>

## 21.4.3. Interrupt flag clear register (TSI\_INTC)

Address offset: 0x08

Reset value: 0x0000 0000

This register can be accessed by word(32-bit).



15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													CMNER	CCTCF	
													R		
													w	w	

Bits	Fields	Descriptions
31:2	Reserved	Must be kept at reset value
1	CMNERR	Clear max cycle number error 0: Reserved 1: Clear MNERR
0	CCTCF	Clear charge-transfer complete flag 0: Reserved 1: Clear CTCF

#### 21.4.4. Interrupt flag register (TSI\_INTF)

Address offset: 0x0C

Reset value: 0x0000 0000

This register can be accessed by word(32-bit).

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													MNERR	CTCF	
													r	r	

Bits	Fields	Descriptions
31:2	Reserved	Must be kept at reset value
1	MNERR	Max Cycle Number Error This bit is set by hardware after charge-transfer sequence stops because it reaches the max cycle number defined by MCN[2:0]. This bit is cleared by writing 1 to CMNERR bit in TSI_INTC register. 0: No Max Count Error 1: Max Count Error
0	CTCF	Charge-Transfer complete flag This bit is set by hardware after charge-transfer sequence stops because all enabled group's sample pins reach the threshold voltage or because the cycle number reaches the value defined by MCN[2:0]. This bit is cleared by writing 1 to

CCTCF bit in TSI\_INTC register.

0: Charge-Transfer not complete

1: Charge-Transfer complete

## 21.4.5. Pin hysteresis mode register (TSI\_PHM)

Address offset: 0x10

Reset value: 0x0000 0000

This register can be accessed by word(32-bit).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								G5P3	G5P2	G5P1	G5P0	G4P3	G4P2	G4P1	G4P0
								r/w							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
G3P3	G3P2	G3P1	G3P0	G2P3	G2P2	G2P1	G2P0	G1P3	G1P2	G1P1	G1P0	G0P3	G0P2	G0P1	G0P0
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

Bits	Fields	Descriptions
31:24	Reserved	Must be kept at reset value
23:0	GxPy	Pin hysteresis mode This bit is set and cleared by software. 0: Pin GxPy Schmitt trigger hysteresis mode disabled 1: Pin GxPy Schmitt trigger hysteresis mode enabled

## 21.4.6. Analog switch register (TSI\_ASW)

Address offset: 0x18

Reset value: 0x0000 0000

This register can be accessed by word(32-bit).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								G5P3	G5P2	G5P1	G5P0	G4P3	G4P2	G4P1	G4P0
								r/w							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
G3P3	G3P2	G3P1	G3P0	G2P3	G2P2	G2P1	G2P0	G1P3	G1P2	G1P1	G1P0	G0P3	G0P2	G0P1	G0P0
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

Bits	Fields	Descriptions
31:24	Reserved	Must be kept at reset value
23:0	GxPy	Analog switch state. This bit is set and cleared by software.

0: Analog switch of GxPy is open  
 1: Analog switch of GxPy is closed

## 21.4.7. Sample configuration register (TSI\_SAMPCFG)

Address offset: 0x20  
 Reset value: 0x0000 0000

This register can be accessed by word(32-bit).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								G5P3	G5P2	G5P1	G5P0	G4P3	G4P2	G4P1	G4P0
								rw							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
G3P3	G3P2	G3P1	G3P0	G2P3	G2P2	G2P1	G2P0	G1P3	G1P2	G1P1	G1P0	G0P3	G0P2	G0P1	G0P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:24	Reserved	Must be kept at reset value
23:0	GxPy	Sample pin mode This bit is set and cleared by software. 0: Pin GxPy is not a sample pin 1: Pin GxPy is a sample pin

## 21.4.8. Channel configuration register (TSI\_CHCFG)

Address offset: 0x28  
 Reset value: 0x0000 0000

This register can be accessed by word(32-bit).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								G5P3	G5P2	G5P1	G5P0	G4P3	G4P2	G4P1	G4P0
								rw							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
G3P3	G3P2	G3P1	G3P0	G2P3	G2P2	G2P1	G2P0	G1P3	G1P2	G1P1	G1P0	G0P3	G0P2	G0P1	G0P0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:24	Reserved	Must be kept at reset value
23:0	GxPy	Channel pin mode This bit is set and cleared by software.

0: Pin GxPy is not a channel pin

1: Pin GxPy is a channel pin

### 21.4.9. Group control register (TSI\_GCTL)

Address offset: 0x30

Reset value: 0x0000 0000

This register can be accessed by word(32-bit).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved										GC5	GC4	GC3	GC2	GC1	GC0
										r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										GE5	GE4	GE3	GE2	GE1	GE0
										rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:24	Reserved	Must be kept at reset value
21:16	GCx	Group complete This bit is set by hardware when charge-transfer sequence for an enabled group is complete. It is cleared by hardware when a new charge-transfer sequence starts. 0: Charge-transfer for group x is not complete 1: Charge-transfer for group x is complete
15:6	Reserved	Must be kept at reset value
5:0	GEx	Group enable This bit is set and cleared by software. 0: Group x is disabled 1: Group x is enabled

### 21.4.10. Group x cycle number registers (TSI\_GxCYCN) (x= 0..5)

Address offset: 0x30 + 0x04 \*(x + 1)

Reset value: 0x0000 0000

This register can be accessed by word(32-bit).

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		CYCN[13:0]													
rw															

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:14	Reserved	Must be kept at reset value
13:0	CYCN[13:0]	Cycle number These bits reflect the cycle number for a group as soon as a charge-transfer sequence completes. They are cleared by hardware when a new charge-transfer sequence starts.

## 22. Universal Serial Bus full-speed device interface (USBD)

The USBD is only available on GD32F150 series.

### 22.1. Overview

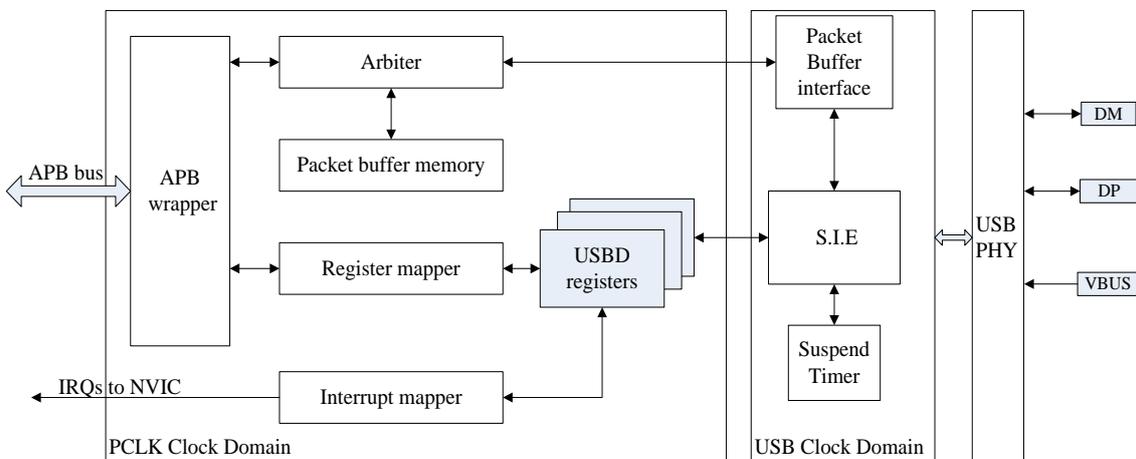
The Universal Serial Bus full-speed device interface (USBD) module provides a device solution for implementing a USB 2.0 full-speed compliant peripheral. It contains a full-speed internal USB PHY and no more external PHY chip is needed. USBD supports all the four types of transfer (control, bulk, interrupt and isochronous) defined in USB 2.0 protocol.

### 22.2. Main features

- USB 2.0 full-speed device controller.
- Support up to 8 configurable bidirectional endpoints.
- Support double-buffered bulk/isochronous endpoints.
- Each endpoint supports control, bulk, isochronous or interrupt transfer types (exclude endpoint 0, endpoint 0 only support control transfer).
- Support USB suspend/resume operations.
- Shared dedicated 512-byte SRAM used for data packet buffer with CAN.
- Integrated USB PHY.

### 22.3. Block diagram

Figure 22-1. USBD block diagram



## 22.4. Signal description

**Table 22-1. USB D signal description**

I/O port	Type	Description
VBUS	Input	Bus power port
DM	Input/Output	Differential D-
DP	Input/Output	Differential D+

**Note:** As soon as the USB D is enabled, these pins are connected to the USB D internal transceiver automatically.

## 22.5. Clock configuration

According to the USB standard definition, the USB full-speed module adopt fixed 48MHz clock. It is necessary to configure two clock for using USB D, one is the USB controller clock, its frequency must be configured to 48MHz, and the other one is the APB1 to USB interface clock which is also APB1 bus clock, its frequency can be above or below 48MHz.

**Note:** In order to meet the system requirements of packet buffer interface and USB data transfer rate, the frequency of the APB1 bus clock must be greater than 24MHz, so as to avoid data buffer overflow and underflow.

48MHz clock of USB controller can be generated by dividing MCU internal or external crystal oscillator by a programmable prescaler, then multiplying the frequency through PLL.

- Regard two frequency division of 8MHz internal oscillator as the input of the PLL, then 12 frequencies doubling the clock.
- Regard 8MHz external oscillator as the input of the PLL, firstly frequency doubling, then adopt USB frequency divider to divide frequency.

When the USB clock is generated by external crystal, only 4 USB frequency prescaler can be used as 1, 1.5, 2 and 2.5. Thus, for obtaining 48MHz clock, PLL frequencies doubling could be configured as 48MHz, 72MHz.

**Note:** Regardless of using internal or external crystal oscillator to generate USB clock, the clock accuracy must reach  $\pm 500\text{ppm}$ . If the accuracy of the USB clock cannot meet the condition, data transfer may not conform to the requirements of the USB specification, and even it may cause USB not working directly.

## 22.6. Function overview

### 22.6.1. USB endpoints

USBBD supports 8 USB endpoints that can be individually configured.

Each endpoint supports:

- Single/Double buffer (endpoint 0 can't use double buffer).
- One endpoint buffer descriptor.
- Programmable buffer starting address and buffer length.
- Configurable response to a packet.
- Control transfer (endpoint 0 only).

#### Endpoint buffer

The function of the device operation is to transfer a request in the memory image to and from the Universal Serial Bus. To efficiently manage USB endpoint communications, USBBD implements a dedicated data packet buffer of 512-bytes SRAM memory accessed directly by the USB peripheral. It is mapped to the APB1 peripheral memory, from 0x4000 6000 to 0x4000 6400. The total capacity is 1KB, but USBBD uses actually only 512 bytes for the bus width reason.

Each endpoint can be associated with one or two data packet buffers used to store the current data payload. The bidirectional endpoint has usually two buffers, one is used for transmission and the other one is for reception. The mono-directional endpoint only has one buffer for data operation.

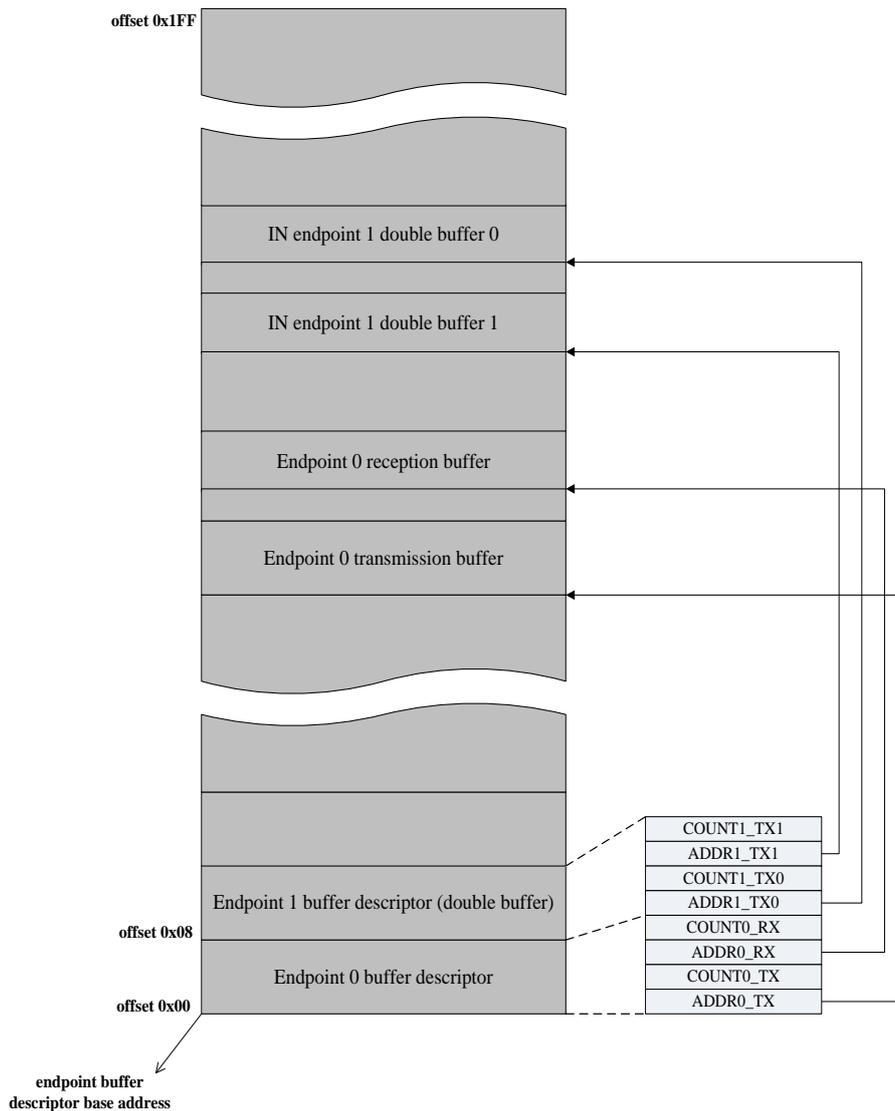
**Note:** The USBBD and CAN share the dedicated 512-byte SRAM memory.

#### Endpoint buffer descriptor table

To indicate where the endpoint-related buffer is located, how large it is or how many bytes must be transmitted, USBBD implements an endpoint buffer descriptor table which defines the buffer address and length and which is also located in the endpoint data packet buffer. The endpoint buffer descriptor is used as a communication port between the application firmware and the SIE in system memory. Every endpoint direction requires two 16-bit words buffer descriptor. Therefore, each table entry includes 4 16-bit words (Tx and Rx two direction) and is aligned to 8-byte boundary. When an endpoint is double-buffered, the SIE will use the two buffers in ping-pong fashion. The endpoint buffer descriptor table is pointed to by the USBBD endpoint buffer address register.

The relationship between endpoint buffer descriptor table entries and packet buffer areas is depicted in [Figure 22-2. An example with buffer descriptor table usage \(USBBD\\_BADDR = 0\).](#)

Figure 22-2. An example with buffer descriptor table usage (USB\_D\_BADDR = 0)



**Note:** This figure is not drawn on the actual scale, and it is addressed through the USB bus 16-bit mode.

### Double-buffered endpoints

The double-buffered feature is used to improve bulk transfer performance. To implement the new flow control scheme, the USB peripheral should know which packet buffer is currently in use by the application software, so to be aware of any conflict. Since in the USB\_D\_EPxCS register, there are two data toggle bits (TX\_DTG and RX\_DTG) but only one is used by USB\_D for hardware data handling (due to the unidirectional constraint required by double-buffering feature), the other one can be used by the application software to show which buffer it is currently using. This new buffer flag is called software buffer bit (SW\_BUF). In [Table 22-2. Double-buffering buffer flag definition](#), the correspondence between USB\_D\_EPxCS register bits and DTG/SW\_BUF definition is explained.

**Table 22-2. Double-buffering buffer flag definition**

Buffer flag	Tx endpoint	Rx endpoint
DTG	TX_DTG (USBD_EPxCS bit 6)	RX_DTG (USBD_EPxCS bit 14)
SW_BUF	USBD_EPxCS bit 14	USBD_EPxCS bit 6

The DTG bit and the SW\_BUF bit are responsible for the flow control. When a transfer completes, the USB peripheral toggle the DTG bit; when the data have been copied, the application software need to toggle the SW\_BUF bit. Except for the first time, if the value of DTG bit is equal to the SW\_BUF's, the transfer will pause, and the host is NAK. When the two bits are not equal, the transfer resume.

**Table 22-3. Double buffer usage**

Endpoint Type	DTOG	SW_BUF	Packet buffer used by the USB peripheral	Packet buffer used by the application software
OUT	0	1	EPxRBADDR/EPxRBCNT buffer description table locations.	EPxTBADDR/EPxTBCNT buffer description table locations.
	1	0	EPxTBADDR/EPxTBCNT buffer description table locations.	EPxRBADDR/EPxRBCNT buffer description table locations.
IN	0	1	EPxTBADDR/EPxTBCNT buffer description table locations.	EPxRBADDR/EPxRBCNT buffer description table locations.
	1	0	EPxRBADDR/EPxRBCNT buffer description table locations.	EPxTBADDR/EPxTBCNT buffer description table locations.

### Endpoint memory requests arbitration

As the USBD is connected to the APB1 bus through an APB1 interface, so USB APB1 interface will accept memory requests coming from the APB1 bus and from the USB interface. The arbiter will resolve the conflicts by giving priority to APB1 accesses, while always reserving half of the memory bandwidth to complete all USB transfers. This time-duplex scheme implements a virtual dual-port SRAM that allows memory access, when an USB transaction is happening. Multiword APB1 transfers of any length are also allowed by this scheme.

## 22.6.2. Operation procedure

### USB transaction process

After the endpoint is configured and a transaction is required, the hardware will detect the token packet. When a token is recognized by the USBD, the data transfer is performed. When

all the data has been transferred, the proper handshake packet over the USB is generated or expected according to the direction of the transfer.

After the transaction process is completed, an endpoint-specific interrupt is generated. In the interrupt routine, the application can process it accordingly.

Transaction formatting is performed by the hardware, including CRC generation and checking.

Once the endpoint is enabled, endpoint control and status register, buffer address and COUNT field should not be modified by the application software. When the data transfer operation is completed, notified by a STIF interrupt event, they can be accessed again to re-enable a new operation.

### **IN transaction**

When a configured and valid endpoint receives an IN token packet, it will send the data packet to the host. If the endpoint is not valid, a NAK or STALL handshake is sent according to the endpoint status.

In the data packet transfer process, a configured data PID will be sent firstly, then the actual data in endpoint buffer memory is loaded into the output shift register to be transmitted. After the data are sent, the computed CRC will be sent by hardware.

When receiving the ACK sent from the host, then the USB peripheral will toggle the data PID and set the endpoint status to be NAK. At the same time, the successful transfer interrupt will be triggered. In the interrupt service routine, application fill the data packet memory with data, start next transfer by re-enable the endpoint by setting the endpoint status VALID.

### **OUT and SETUP transaction**

USB handles these two tokens more or less in the same way, the differences in the handling of SETUP packets will be detailed in the following section about control transfer.

After the received endpoint is configured and enabled, host will send OUT/SETUP token to the device. When receiving the token, USB will access the endpoint buffer descriptor to initialize the endpoint buffer address and length. Then the received data bytes subsequently are packed in words (LSB mode) and transferred to the endpoint buffer. When detecting the end of data packet, the computed CRC and received CRC are compared. If no errors occur, an ACK handshake packet is sent to the host.

When the transaction is completed correctly, USB will toggle the data PID and set the endpoint status to be NAK. Then the endpoint successful transfer interrupt will be triggered by hardware. In the interrupt service routine, the application can get the transaction type and read the received data from the endpoint buffer. After the received data is processed, the application should initiate further transactions by setting the endpoint status valid.

If any error happens during reception, the USB set the error interrupt bit and still copy data into the packet memory buffer, but will not send the ACK packet. The USB itself can recover from reception errors and continue to handle next transfer. The USB never override outside

the data buffer, which is controlled by the internal register configured. The received 2-byte CRC is also copied to the packet memory buffer, immediately following data bytes. If the length of data is greater than actually allocated length, the excess data are not copied. This is a buffer overrun situation. A STALL handshake is sent, and this transaction fails.

If an addressed endpoint is not valid, a NAK or STALL handshake packet is sent instead of the ACK, according to the endpoint status and no data is written to the endpoint data buffers.

If the length of the data packet payload (actual number of bytes used by the application) is greater than the allocated endpoint data buffer, the USBD detects a buffer overrun condition. In this case, a STALL handshake is sent instead of the usual ACK to notify the problem to the host, no interrupt is generated and the transaction is considered failed.

### Control transfers

Control transfers require that a SETUP transaction be started from the host to a device to describe the type of control access that the device should perform. The SETUP transaction is followed by zero or more control DATA transactions that carry the specific information for the requested access. Finally, a STATUS transaction completes the control transfer and allows the endpoint to return the status of the control transfer to the client software. After the STATUS transaction for a control transfer is completed, the host can advance to the next control transfer for the endpoint.

USB always use endpoint 0 in two directions as default control endpoint to handle control transfers. It is aware of the number and direction of data stages by interpreting the contents of SETUP transaction, and is required to set the unused direction endpoint 0 status to STALL except the last data stage.

At the last data stage, the application software set the opposite direction endpoint 0 status to NAK. This will keep the host waiting for the completion of the control operation. If the operation completes successfully, the software will change NAK to VALID, otherwise to STALL. If the status stage is an OUT, the STATUS\_OUT bit should be set, so that a status transaction with non-zero data will be answered STALL to indicate an error happen.

As USB specification states, a device must answer a SETUP packet with an ACK handshake. The USB doesn't allow a control endpoint to answer with a NAK or STALL handshake packet to a SETUP token when device aborts a previously issued command to start the new one eventually.

When the configured control endpoint 0 receives a SETUP token, the USBD accepts the data, performing the required data transfers and sends back an ACK handshake. If that endpoint has a previously issued successful receive interrupt request not yet acknowledged by the application (i.e. RX\_ST bit is still set from a previously completed reception), the USBD discards the SETUP transaction and does not answer with any handshake packet regardless of its state, simulating a reception error and forcing the host to send the SETUP token again. This is done to avoid losing the notification of a SETUP transaction addressed to the same endpoint immediately following the transaction, which triggered the successful receive

interrupt.

### **Isochronous transfers**

Isochronous transfers can guarantee constant data rate and bounded latency, but do not support data retransmission in response to errors on the bus. A receiver can determine that a transmission error occurred. The low-level USB protocol does not allow handshakes to be returned to the transmitter of an isochronous pipe. Normally, handshakes would be returned to tell the transmitter whether a packet was successfully received or not. Consequently, the isochronous transaction does not have a handshake phase, and have no ACK packet after the data packet. Data toggling is not supported, and DATA0 PID is only used to start a data packet.

The isochronous endpoint status only can be set DISABLED and VALID, any other value is illegal. The application software can implement double-buffering to improve performance. By swapping transmission and reception data packet buffer on each transaction, the application software can copy the data into or out of a buffer, at the same time the USB peripheral handle the data transmission or reception of data in another buffer. The DTOG bit indicates which buffer that the USB peripheral is currently using.

The application software initializes the DTOG according to the first buffer to be used. At the end of each transaction, the RX\_ST or TX\_ST bit is set, depending on the enabled direction regardless of CRC errors or buffer-overflow conditions (if errors occur, the ERRIF bit will be set). At the same time, The USB peripheral will toggle the DTOG bit, but will not affect the STAT bit.

### **22.6.3. USB events and interrupts**

Each USB action is always initiated by the application software, driven by one USB interrupt or event. After system reset, the application needs to wait for a succession of USB interrupts and events.

#### **Reset events**

##### **System and power-on reset**

Upon system and power-on reset, the application software should first provide all required clock to the USB module and interface, then de-assert its reset signal so to be able to access its registers, last switch on the analog part of the device related to the USB transceiver.

The USB firmware should do as follows:

- Reset CLOSE bit in USBD\_CTL register.
- Wait for the internal reference voltage to be stable.
- Clear SETRST bit in USBD\_CTL register.
- Clear the IFR register to remove the spurious pending interrupt and then enable other unit.

### USB reset (RESET interrupt)

When this event occurs, the USB peripheral status is the same as the moment system reset.

The USB firmware should do as follows:

- Set USBEN bit in AR register to enable USB module in 10ms.
- Initialize the USBD\_EP0CS register and its related packet buffers.

### Suspend and resume events

The USB module can be forced to place in low-power mode (SUSPEND mode) by writing in the USB control register (USBD\_CTL) whenever required. At this time, all static power dissipation is avoided and the USB clock can be slowed down or stopped. It will be resumed when detect activity at the USB bus while in low-power mode.

The USB protocol insists on power management by the USB device. This becomes even more important if the device draws power from the bus (bus-powered device). The following constraints should be met by the bus-powered device.

- A device in the non-configured state should draw a maximum of 100mA from the USB bus.
- A configured device can draw only up to what is specified in the Max Power field of the configuration descriptor. The maximum value is 500mA.
- A suspended device should draw a maximum of 500uA.

A device will go into the suspend state if there is no activity on the USB bus for more than 3ms. A suspended device wakes up, if RESUME signaling is detected.

USBD also supports software initiated remote wakeup. To initiate remote wakeup, the application software must enable all clocks and clear the suspend bit after MCU is waked up. This will cause the hardware to generate a remote wakeup signal upstream.

Setting the SETSPS bit to 1 enables the suspend mode, and it will disable the check of SOF reception. Setting the LOWM bit to 1 will shut down the static power consumption in the analog USB transceivers, but the RESUME signal is still able to be detected.

### USB Interrupts

USBD has three interrupts: low-priority interrupt, high-priority interrupt and wakeup interrupt. Software can configure these interrupts to route the interrupt condition to these entries in the NVIC table. An interrupt will be generated when both the interrupt status bit and the corresponding interrupt enable bit are set. The interrupt status bit is set by hardware if the interrupt condition occurs (irrespective of the interrupt enable bit).

- Low-priority interrupt (Channel 37): triggered by all USB events.
- High-priority interrupt (Channel 38): triggered only by a correct transfer event for isochronous and double-buffer bulk transfer.
- Wakeup interrupt (Channel 42): triggered by the wakeup events.

## 22.6.4. Operation guide

This section describes the operation guide for USBD.

### USB D register initialization sequence

1. Clear the CLOSE bit in USBD\_CTL register, then clear the SETRST bit.
2. Clear USBD\_INTF register to remove any spurious pending interrupt.
3. Program USBD\_BADDR register to set endpoint buffer base address.
4. Set USBD\_CTL register to enable interrupts.
5. Wait for the reset interrupt (RSTIF).
6. In the reset interrupt, initialize default control endpoint 0 to start enumeration process and program USBD\_ADDR to set the device address to 0 and enable USB module function.
7. Configure endpoint 0 and prepare to receive SETUP packet.

### Endpoint initialization sequence

1. Program USBD\_EPxTBADDR or USBD\_EPxRBADDR registers with transmission or reception data buffer address.
2. Program the EP\_CTL and EP\_KCTL bits in USBD\_EPxCS register to set endpoint type and buffer kind according to the endpoint usage.
3. If the endpoint is a single buffer endpoint:
  - 1) Initialize the endpoint data toggle bit by programming the TX\_DTG or RX\_DTG bit in USBD\_EPxCS register, but endpoint 0 needs to set them to 1 and 0 respectively for control transfer.
  - 2) Configure endpoint status by programming the TX\_STA bit or RX\_STA bit in USBD\_EPxCS register, but both of them are set to '10 (NAK) if use endpoint 0 to initialize the control transfer.

If the endpoint is a double buffer endpoint:

- 1) Both transmission and reception toggle fields need to be programmed. If the endpoint is a Tx endpoint, clear the TX\_DTG and RX\_DTG bit in USBD\_EPxCS register, or if endpoint is a Rx endpoint, it needs to toggle TX\_DTG bit.
- 2) Program USBD\_EPxTBCNT and USBD\_EPxRBCNT register to set transfer data bit count.
- 3) Endpoint transmission and reception status both need to be configured. If the endpoint is a Tx endpoint, set the TX\_STA bit to be NAK and RX\_STA bit to be DISABLED, or the endpoint is a Rx endpoint, set the RX\_STA bit to be VALID and TX\_STA bit to be

DISABLED.

## SETUP and OUT data transfers

1. Program USBD\_EPxRBCNT register to set BLKNUM and RXCNT filed, these filed defines the endpoint buffer length.
2. Configure the endpoint status to be VALID to enable the endpoint to receive data by programming USBD\_EPxCS register.
3. Wait for successful transfer interrupt (STIF).
4. In the interrupt handler, application can get the transaction type by reading the STEUP bit in USBD\_EPxCS register. Then application will read the data payload from the endpoint data buffer with the start address defined in USBD\_EPxRBAR register. Last application will interpret the data and process the corresponding transaction.

## IN data transfers

1. Program USBD\_EPxTBCNT register to set TCNT filed, this filed defines the endpoint buffer length.
2. Configure the endpoint status to be VALID to enable the endpoint to transmit data by programming USBD\_EPxCS register.
3. Wait for successful transfer interrupt (STIF).
4. In the interrupt handler, application needs to update user buffer length and location pointer. Then application fill the endpoint buffer with user buffer data. Last application will configure the endpoint status to be VALID to start next transfer.

## 22.7. Register definition

### 22.7.1. USB control register (USB\_CTL)

Address offset: 0x40

Reset value: 0x0003

This register can be accessed by half-word (16-bit) or word (32-bit)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STIE	PMOUIE	ERRIE	WKUPIE	SPSIE	RSTIE	SOFIE	ESOFIE	Reserved			RSREQ	SETSPS	LOWM	CLOSE	SETRST
rw	rw	rw	rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits	Fields	Descriptions
15	STIE	Successful transfer interrupt enable.  0: Successful transfer interrupt disabled. 1: Interrupt generated when STIF bit in USB_CTL register is set.
14	PMOUIE	Packet memory overrun/underrun interrupt enable.  0: No interrupt generated when packet memory overrun / underrun. 1: Interrupt generated when PMOUIF bit in USB_CTL register is set.
13	ERRIE	Error interrupt enable.  0: Error interrupt disabled 1: Interrupt generated when ERRIF bit in USB_CTL register is set.
12	WKUPIE	Wakeup interrupt enable  0: Wakeup interrupt disabled 1: Interrupt generated when WKUPIF bit in USB_CTL register is set.
11	SPSIE	Suspend state interrupt enable  0: Suspend state interrupt disabled 1: Interrupt generated when SPSIF bit in USB_I2R register is set.
10	RSTIE	USB reset interrupt enable.  0: USB reset interrupt disabled 1: Interrupt generated when RSTIF bit in USB_CTL register is set.
9	SOFIE	Start of frame interrupt enable  0: Start of frame interrupt disabled 1: Interrupt generated when SOFIF bit in USB_CTL register is set.
8	ESOFIE	Expected start of frame interrupt enable

		0: Expected start of frame interrupt disabled 1: Interrupt generated when ESOFIF bit in USBD_INTF register is set.
7:5	Reserved	Must be kept at reset value
4	RSREQ	Resume request  The software set a resume request to the USB host, and the USB host should drive the resume sequence according the USB specifications  0: No resume request 1: Send resume request.
3	SETSPS	Set suspend  The software should set suspend state when SPSIF bit in USBD_INTF register is set.  0: Not set suspend state. 1: Set suspend state.
2	LOWM	Low-power mode  When set this bit, the USB goes to low-power mode at suspend state. If resume from suspend state, the hardware reset this bit.  0: No effect 1: Go to low-power mode at suspend state.
1	CLOSE	Close state  When this bit is set, the USBD goes to close state, and completely close the USBD and disconnected from the host.  0: Not in close state 1: In close state.
0	SETRST	Set reset  When this bit is set, the USBD peripheral should be reset.  0: No reset 1: A reset generated.

## 22.7.2. USBD interrupt flag register (USB\_D\_INTF)

Address offset: 0x44

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STIF	PMOUIF	ERRIF	WKUIF	SPSIF	RSTIF	SOFIF	ESOFIF	Reserved			DIR	EPNUM[3:0]			
r	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0				r	r			

Bits	Fields	Descriptions
15	STIF	Successful transfer interrupt flag This bit set by hardware when a successful transaction completes
14	PMOUIF	Packet memory overrun/underrun interrupt flag This bit set by hardware to indicate that the packet memory is inadequate to hold transfer data. The software writes 0 to clear this bit.
13	ERRIF	Error interrupt flag This bit set by hardware when an error happens during transaction. The software writes 0 to clear this bit.
12	WKUIF	Wakeup interrupt flag This bit set by hardware in the SUSPEND state to indicate that activity is detected. The software writes 0 to clear this bit.
11	SPSIF	Suspend state interrupt flag When no traffic happen in 3ms, hardware set this bit to indicate a SUSPEND request. The software writes 0 to clear this bit.
10	RSTIF	USB reset interrupt flag Set by hardware when the USB RESET signal is detected. The software writes 0 to clear this bit.
9	SOFIF	Start of frame interrupt flag Set by hardware when a new SOF packet arrives, The software writes 0 to clear this bit.
8	ESOFIF	Expected start of frame interrupt flag Set by the hardware to indicate that a SOF packet is expected but not received. The software writes 0 to clear this bit.
7:5	Reserved	Must be kept at reset value
4	DIR	Direction of transaction Set by the hardware to indicate the direction of the transaction 0: OUT type 1: IN type
3:0	EPNUM[3:0]	Endpoint Number Set by the hardware to identify the endpoint which the transaction is directed to

### 22.7.3. USB status register (USB\_STAT)

Address offset: 0x48

Reset value: 0x0XXX where X is undefined

This register can be accessed by half-word (16-bit) or word (32-bit)

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

RX_DP	RX_DM	LOCK	SOFLN[1:0]	FCNT[10:0]
r	r	r	r	r

Bits	Fields	Descriptions
15	RX_DP	Receive data + line status Represent the status on the DP line
14	RX_DM	Receive data - line status Represent the status on the DM line
13	LOCK	Locked the USB Set by the hardware indicate that at the least two consecutive SOF have been received
12:11	SOFLN[1:0]	SOF lost number Increment every ESOFIF happens by hardware Cleared once the reception of SOF
10:0	FCNT[10:0]	Frame number counter The Frame number counter incremented every SOF received.

## 22.7.4. USB device address register (USB\_DADDR)

Address offset: 0x4C

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)

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

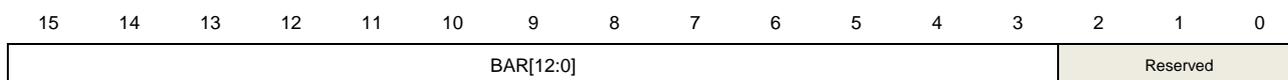
Bits	Fields	Descriptions
15:8	Reserved	Must be kept at reset value
7	USBEN	USB device enable Set by software to enable the USB device  0: The USB device disabled. No transactions handled. 1: The USB device enabled.
6:0	USB DAR[6:0]	USB device address After bus reset, the address is reset to 0x00. If the enable bit is set, the device will respond on packets for function address DEV_ADDR

## 22.7.5. USB buffer address register (USB\_BADDR)

Address offset: 0x50

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



rw

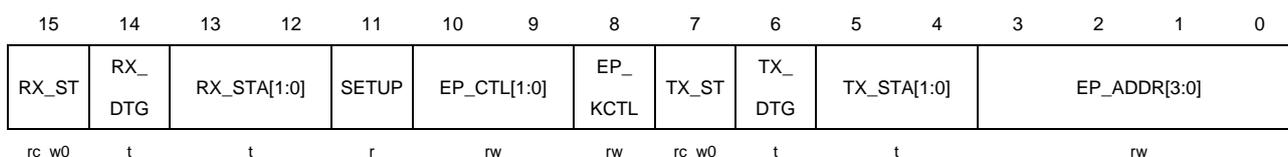
Bits	Fields	Descriptions
15:3	BAR[12:0]	Buffer address Start address of the allocation buffer(512byte on-chip SRAM), used for buffer descriptor table, packet memory
2:0	Reserved	Must be kept at reset value

### 22.7.6. USB endpoint x control and status register (USB\_EPxCS), x=[0..7]

Address offset: 0x00 to 0x1C

Reset value: 0x0000

This register can be accessed by half-word (16-bit) or word (32-bit)



Bits	Fields	Descriptions
15	RX_ST	Reception successful transferred Set by hardware when a successful OUT/SETUP transaction complete Cleared by software by writing 0
14	RX_DTG	Reception data PID toggle This bit represent the toggle data bit (0=DATA0,1=DATA1)for non-isochronous endpoint Used to implement the flow control for double-buffered endpoint Used to swap buffer for isochronous endpoint
13:12	RX_STA[1:0]	Reception status bits Toggle by writing 1 by software Remain unchanged by writing 0 Refer to the table below
11	SETUP	Setup transaction completed Set by hardware when a SETUP transaction completed.
10:9	EP_CTL[1:0]	Endpoint type control Refer to the table below
8	EP_KCTL	Endpoint kind control

The exact meaning depends on the endpoint type  
Refer to the table below

7	TX_ST	<p>Transmission successful transfer</p> <p>Set by hardware when a successful IN transaction complete</p> <p>Clear by software</p>
6	TX_DTG	<p>Transmission data PID toggle</p> <p>This bit represent the toggle data bit (0=DATA0,1=DATA1)for non-isochronous endpoint</p> <p>Used to implement the flow control for double-buffered endpoint</p> <p>Used to swap buffer for isochronous endpoint</p>
5:4	TX_STA[1:0]	<p>Status bits, for transmission transfers</p> <p>Refer to the table below</p>
3:0	EP_ADDR	<p>Endpoint address</p> <p>Used to direct the transaction to the target endpoint</p>

**Table 22-4. Reception status encoding**

RX_STA[1:0]	Meaning
00	<b>DISABLED:</b> ignore all reception requests of this endpoint
01	<b>STALL:</b> STALL handshake status
10	<b>NAK:</b> NAK handshake status
11	<b>VALID:</b> enable endpoint for reception

**Table 22-5. Endpoint type encoding**

EP_CTL[1:0]	Meaning
00	<b>BULK:</b> bulk endpoint
01	<b>CONTROL:</b> control endpoint
10	<b>ISO:</b> isochronous endpoint
11	<b>INTERRUPT:</b> interrupt endpoint

**Table 22-6. Endpoint kind meaning**

EP_CTL[1:0]		EP_KCTL Meaning
00	BULK	DBL_BUF
01	CONTROL	STATUS_OUT

**Table 22-7. Transmission status encoding**

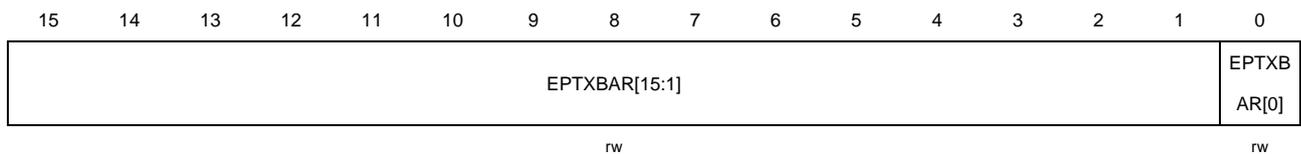
TX_STA[1:0]	Meaning
00	<b>DISABLED:</b> ignore all transmission requests of this endpoint
01	<b>STALL:</b> STALL handshake status
10	<b>NAK:</b> NAK handshake status
11	<b>VALID:</b> enable endpoint for transmission

**22.7.7. USBD endpoint x transmission buffer address register (USBD\_EPxTBADDR), x=[0..7]**

Address offset:  $[USB\_BADDR] + x * 16$

USB local address:  $[USB\_BADDR] + x * 8$

This register can be accessed by half-word (16-bit) or word (32-bit)



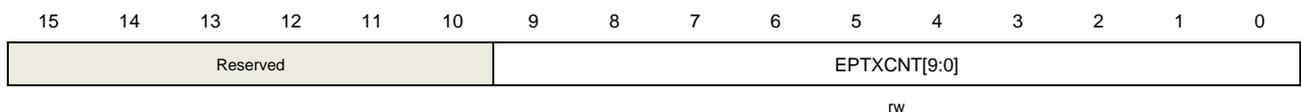
Bits	Fields	Descriptions
15:1	EPTXBAR[15:1]	Endpoint transmission buffer address Start address of the packet buffer containing data to be sent when receive next IN token
0	EPTXBAR[0]	Must be set to 0

**22.7.8. USBD endpoint x transmission buffer byte count register (USBD\_EPxTBCNT), x=[0..7]**

Address offset:  $[USB\_BADDR] + x * 16 + 4$

USB local Address:  $[USB\_BADDR] + x * 8 + 2$

This register can be accessed by half-word (16-bit) or word (32-bit)



Bits	Fields	Descriptions
15:10	Reserved	Must be kept at reset value
9:0	EPTXCNT[9:0]	Endpoint transmission byte count The number of bytes to be transmitted at next IN token

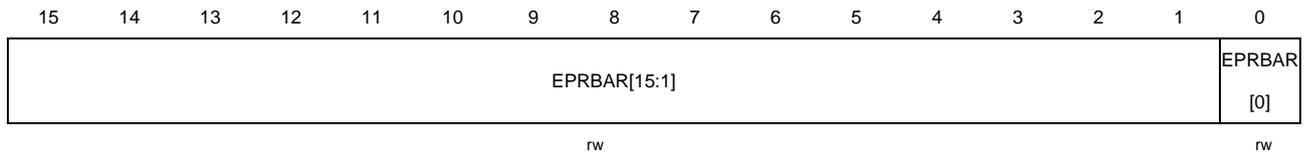
## 22.7.9. USB D endpoint x reception buffer address register

### (USB\_D\_EPxRBADDR), x=[0..7]

Address offset: [USB\_D\_BADDR] + x \* 16 + 8

USB local Address: [USB\_BADDR] + x \* 8 + 4

This register can be accessed by half-word (16-bit) or word (32-bit)



Bits	Fields	Descriptions
15:1	EPRBAR[15:1]	Endpoint reception buffer address  Start address of packet buffer containing the data received by the endpoint at the next OUT/SETUP token
0	EPRBAR[0]	Must be set to 0

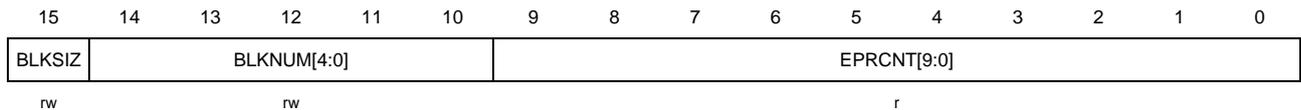
## 22.7.10. USB D endpoint x reception buffer byte count register

### (USB\_D\_EPxRBCNT), x=[0..7]

Address offset: [USB\_D\_BADDR] + x \* 16 + 12

USB local Address: [USB\_BADDR] + x \* 8 + 6

This register can be accessed by half-word (16-bit) or word (32-bit)



Bits	Fields	Descriptions
15	BLKSIZ	Block size  0: block size is 2 bytes 1: block size is 32 bytes
14:10	BLKNUM[4:0]	Block number  The number of blocks allocated to the packet buffer
9:0	EPRCNT[9:0]	Endpoint reception byte count  The number of bytes to be received at next OUT/SETUP token

## 23. Segment LCD controller (SLCD)

This chapter applies to GD32F170xx and GD32F190xx devices.

### 23.1. Overview

The SLCD controller directly drives LCD displays by creating the AC segment and common voltage signals automatically. It can drive the monochrome passive liquid crystal display (LCD) which composed of a plurality of segments (pixels or complete symbols) that can be converted to visible or invisible. The SLCD controller can support up to 32 segments and 8 commons.

### 23.2. Characteristics

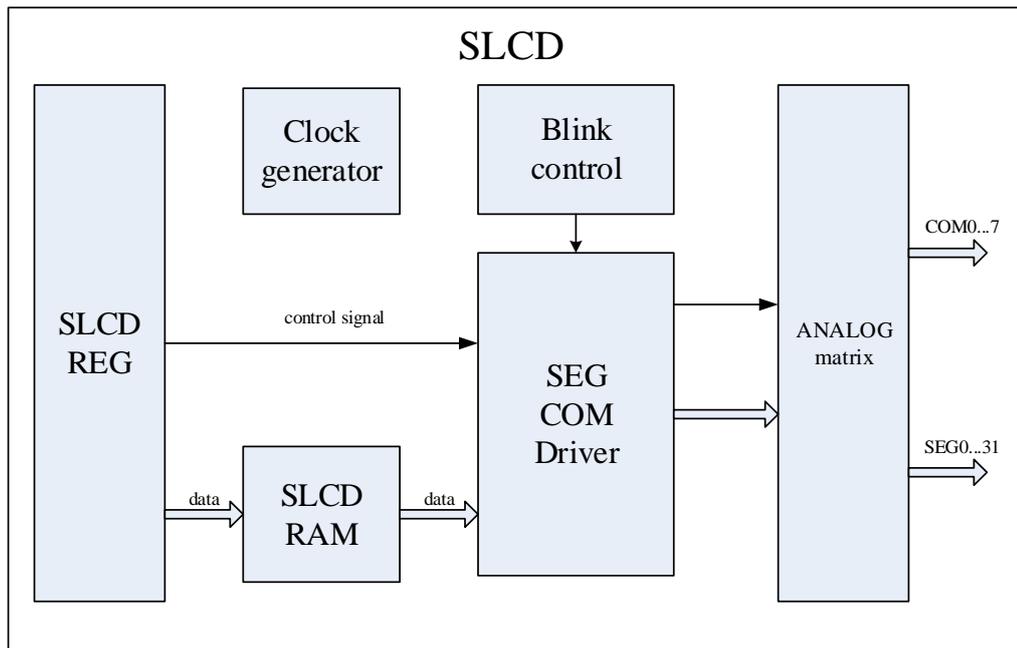
- Configurable frame frequency
- Blinking of individual segments or all segments
- Supports Static, 1/2, 1/3, 1/4, 1/6 and 1/8 duty
- Supports 1/2, 1/3 and 1/4 bias
- Double buffer up to 8x32 bits registers to store SLCD\_DATAx
- The contrast can also be adjusted by configuring dead time
- VSLCD rails decoupling capability

### 23.3. Function overview

#### 23.3.1. SLCD Architecture

The block diagram of the SLCD controller is shown as follows.

Figure 23-1. SLCD Block Diagram



The SLCD REG is the register of SLCD controller, which configured by APB bus, and generate interrupt to CPU. It includes SLCD\_CTL, SLCD\_CFG, SLCD\_STAT, SLCD\_STATC, SLCD\_DATAx registers.

The Clock generator generates SLCD clock from input clock. The SLCD clock drives the blink control and SEG/COM driver. The Blink control generates blink frequency and blink pixels. The SEG/COM driver generates segment and common signals to ANALOG matrix. The ANALOG matrix implements segment and common voltages.

### 23.3.2. Clock generator

SLCD input clock is the same as RTCCLK, 3 different clock sources: LXTAL, IRC40K and HXTAL divided by 32 can be selected by RTCSRC bits in RCU\_BDCTL register. The input clock frequency varies from 32KHz to 1MHz.

The SLCD controller uses the input clock signal from the integrated clock divider to generate the timing for common and segment lines. The SLCD clock frequency is selected with the PSC and DIV bits in SLCD\_CTL registers. The resulting SLCD clock frequency is calculated by:

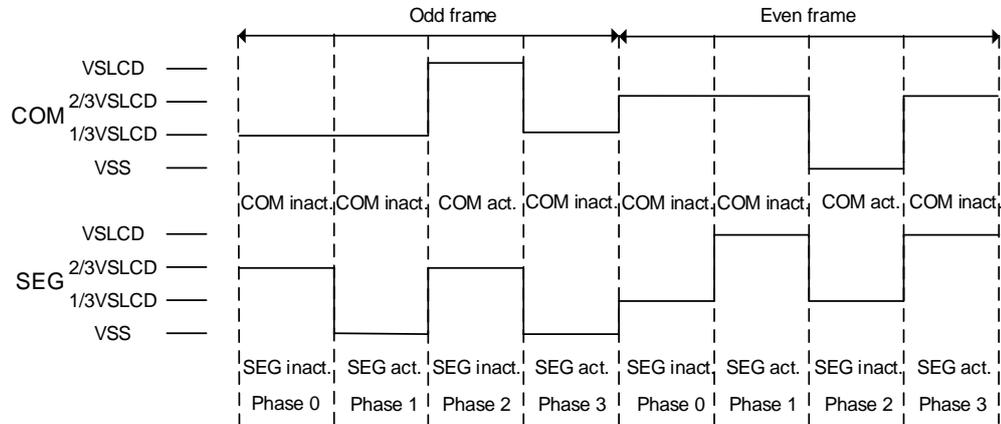
$$f_{\text{SLCD}} = \frac{f_{\text{in\_clk}}}{2^{\text{PSC}} \times (\text{DIV} + 16)}$$

The SLCD clock is the time base for the SLCD controller. The frequency of SLCD clock is equivalent to the phase frequency. One SLCD frame is one odd frame or one even frame, both of them have several phases as many as active common terminals. The duty is the Number defined as 1/ (the number of common terminals on a given SLCD display). So the frame frequency is calculated by:

$$f_{\text{frame}} = f_{\text{SLCD}} \times \text{Duty}$$

The SOF bit in SLCD\_STAT register is set by the hardware at the start of the frame, and the SLCD interrupt is executed if the SOFIE bit in SLCD\_CFG is set. SOF is cleared by writing 1 to the SOFC bit in SLCD\_STATC register.

**Figure 23-2. 1/3 Bias, 1/4 Duty**



### 23.3.3. Blink control

The SLCD controller also supports blinking. The blinking mode controlled by BLKMOD bits in SLCD\_CFG register. BLKMOD = 01 allows to blink individual segment on SEG0 with COM0, with BLKMOD = 10 all commons on SEG0 are blinking, with BLKMOD = 11 all segments with all commons are blinking, and with BLKMOD = 00 blinking is disabled.

The blink frequency is generated from SLCD clock and selected with BLKDIV bit in SLCD\_CFG registers. The resulting BLINK frequency is calculated by:

$$f_{\text{BLINK}} = \frac{f_{\text{SLCD}}}{2^{(\text{BLKDIV}+3)}}$$

After a blinking mode (BLKMOD = 01, 10 or 11) is selected, the enabled segments or all segments go blank at the next frame boundary and stay off for half a BLKCLK period. Then they go active at the next frame boundary and stay on for another half BLKCLK period before they go blank again at a frame boundary.

### 23.3.4. SEG/COM Driver

SEG/COM Driver generates segments and commons signals.

#### BIAS generator:

The bias is selected by setting BIAS bits in SLCD\_CTL registers. It has its max amplitude VSLCD or VSS only in the corresponding phase of a frame cycle. The odd frame voltage and even frame voltage are shown as follows:

**Table 23-1. The odd frame voltage**

BIAS	Static	1/2 bias	1/3 bias	1/4 bias
COM active	VSLCD	VSLCD	VSLCD	VSLCD

BIAS	Static	1/2 bias	1/3 bias	1/4 bias
COM inactive	/	1/2 VSLCD	1/3 VSLCD	1/4 VSLCD
SEG active	VSS	VSS	VSS	VSS
SEG inactive	VSLCD	VSLCD	2/3 VSLCD	1/2 VSLCD

**Table 23-2. The even frame voltage**

BIAS	Static	1/2 bias	1/3 bias	1/4 bias
COM active	VSS	VSS	VSS	VSS
COM inactive	/	1/2 VSLCD	2/3 VSLCD	3/4 VSLCD
SEG active	VSLCD	VSLCD	VSLCD	VSLCD
SEG inactive	VSS	VSS	1/3 VSLCD	1/2 VSLCD

### COM signal:

The common signal is selected by DUTY bits in SLCD\_CTL register. When DUTY is 000, static duty selected. Only COM[0] used and only one phase in odd frame or even frame, so COM[0] driver active signal always. When DUTY is 001, only COM[1:0] and 2 phases used. When DUTY is 010, only COM[2:0] and 3 phases used. When DUTY is 011, only COM[3:0] and 4 phases used. When DUTY is 100, COM[7:0] and 8 phases used. When DUTY is 101, COM[5:0] and 6 phases used. The all common signal driver is shown as follows:

**Table 23-3. The all common signal driver**

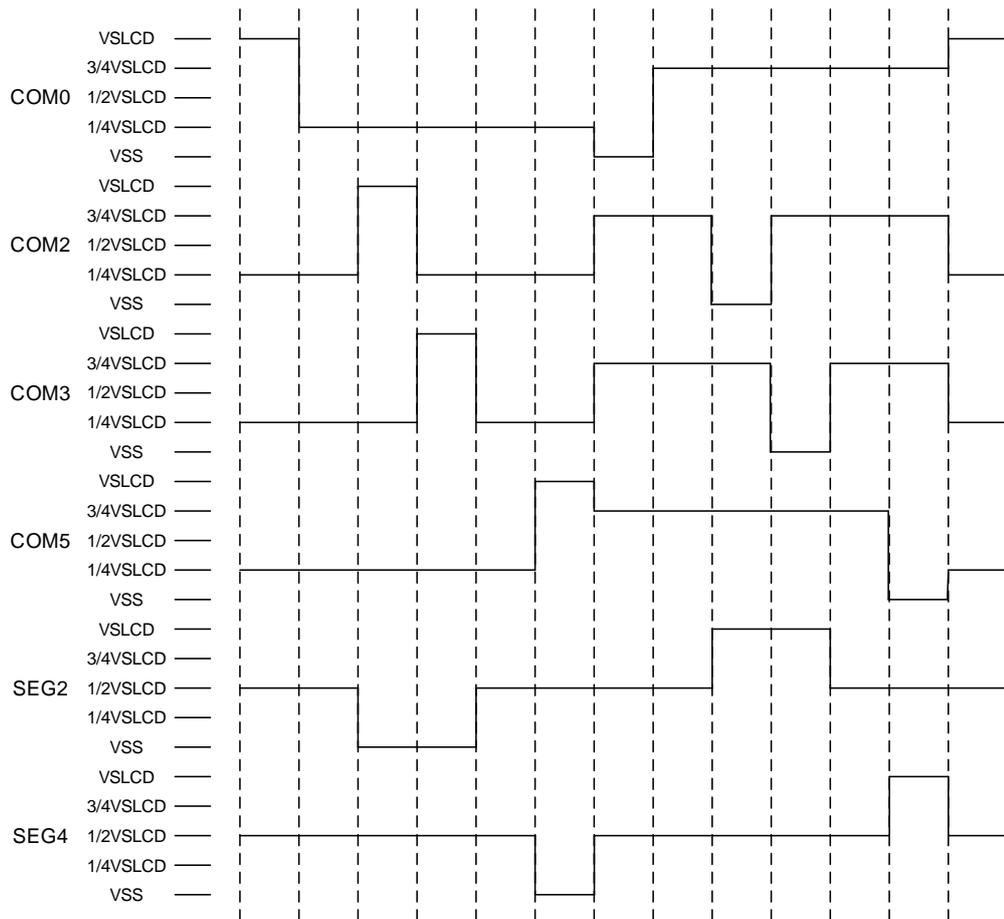
phase	1	2	3	4	5	6	7	8
COM0	active	inactive						
COM1	inactive	active	inactive	inactive	inactive	inactive	inactive	inactive
COM2	inactive	inactive	active	inactive	inactive	inactive	inactive	inactive
COM3	inactive	inactive	inactive	active	inactive	inactive	inactive	inactive
COM4	inactive	inactive	inactive	inactive	active	inactive	inactive	inactive
COM5	inactive	inactive	inactive	inactive	inactive	active	inactive	inactive
COM6	inactive	inactive	inactive	inactive	inactive	inactive	active	inactive
COM7	inactive	active						

### SEG signal:

The segment signals are read from SLCD\_DATAx registers. The segment signals data are SLCD\_DATAx when phase x. When the value is 1, the corresponding segment drives active signal. When the value is 0, the corresponding segment drives inactive signal.

For example, if the application need to active the pixel COM2 SEG2, COM3 SEG2, and COM5 SEG4. It should set the bit2 in the SLCD\_DATA2, the bit2 in the SLCD\_DATA3, and the bit4 in the SLCD\_DATA5. Then the SEG2 signal will active at the third and fourth phase of each odd and even frame, the SEG4 signal will active at the sixth phase of each odd and even frame. The active and inactive voltages are shown in Table 23-1 and Table 23-2. The segment signals show in figure 23-3 is the result to the above configuration when bias is 1/4 and duty is 1/6.

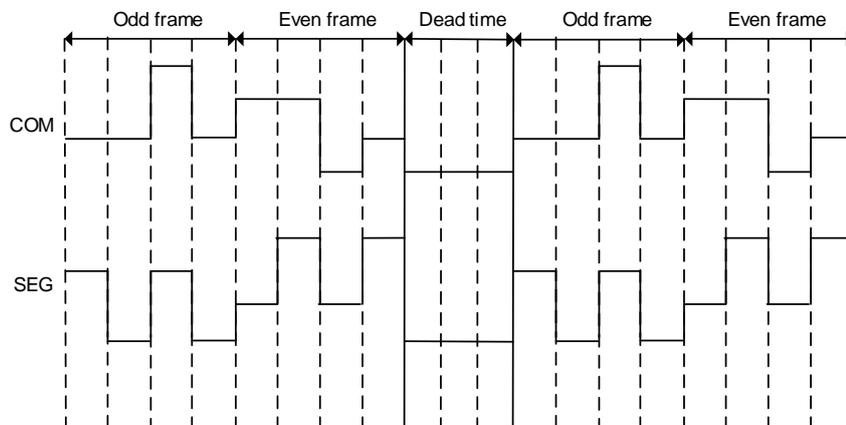
**Figure 23-3. 1/4 Bias, 1/6 Duty**



**DEAD time:**

The dead time is using DTD bits in SLCD\_CFG register. It inserts VSS after each even frame. The number of phase inserted is defined by DTD bits. The application can adjust the contrast according to the configuration of dead time.

**Figure 23-4. SLCD dead time (1/3 Bias, 1/4 Duty)**



### 23.3.5. Double buffer memory

The double buffer memory is used to ensure the coherency of the displayed information.

The application access the first buffer according to modify the SLCD\_DATAx registers. After writing the displayed information into the SLCD\_DATAx registers, the application need to set the UPRF bit in SLCD\_STAT register, then the hardware will transfer the data from the first buffer to the second buffer, during this time, the UPRF keeps set and the SLCD\_DATAx registers are write protected. When the transfer is completed, the UPRF is cleared and the UPRF is set by the hardware, an interrupt will be generated if the UPDIE is set. The segment signal is driven by the data in the second buffer, so the displayed information will not be influenced by writing SLCD\_DATAx.

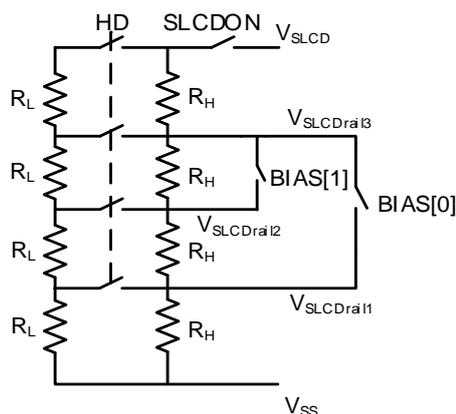
If the UPRF is set when the display is disabled (SLCDON = 0), the transfer will not occur until the SLCDON is set.

### 23.3.6. ANALOG matrix

The analog matrix supplies SLCD voltage. The SLCD voltage levels are generated by the VSLCD pin or by the internal voltage step-up converter (depending on the VSRC bit in the SLCD\_CTL register). When using the internal voltage, the VSLCD value can be selected from VSLCD0 to VSLCD7 by the CONR[2:0] bits in the SLCD\_CFG register (Refer to the product datasheet for the VSLCDx values). The application can adjust the contrast according to the change of VSLCD value. The other way to adjust the contrast is by using the deadtime.

The analog matrix supplies intermediate voltage levels (1/3 VSLCD, 2/3 VSLCD or 1/4 VSLCD, 2/4 VSLCD, 3/4 VSLCD) between VSS and VSLCD through an internal resistor divider network as shown in the figure 23-5.

**Figure 23-5. Resistr divider network**



During the transitions, the low value resistors ( $R_L$ ) are switched on to increase the current in order to quickly reach the static state. Then the low value resistors ( $R_L$ ) are switched off, the high value resistors ( $R_H$ ) are used to reduce the power. The length of the time during  $R_L$  is switched on depend on the PULSE[2:0] bits in the SLCD\_CFG register. The  $R_L$  can be always switched on according to setting the HDEN bit in the SLCD\_CFG register.

## External decoupling:

The VSLCD intermediate voltage rails (VSLCDrail1, VSLCDrail2, VSLCDrail3 in the figure 23-5) can be connect to the GPIOs by configuring the SLCD\_DECA bits of the SYSCFG\_CFG1 register. Adding decoupling capacitors on these GPIOs helps to get a steady voltage resulting in a higher contrast. Thus the PULSE[2:0] is allowed to select low value to reduce the power.

**Note:** When using VSLCDrail function, GPIO should be configured to analog input mode. The segment AFIO and VSLCDrail decoupling functions cannot be used simultaneously.

**Table 23-4. VSLCDrail connect pins**

	BIAS			Pin
	1/2	1/3	1/4	
<b>VSLCDrail1</b>	1/2VSLCD	1/3VSLCD	1/4VSLCD	PB12
<b>VSLCDrail2</b>	1/2VSLCD	2/3VSLCD	1/2VSLCD	PB2
<b>VSLCDrail3</b>	1/2VSLCD	2/3VSLCD	3/4VSLCD	PB0

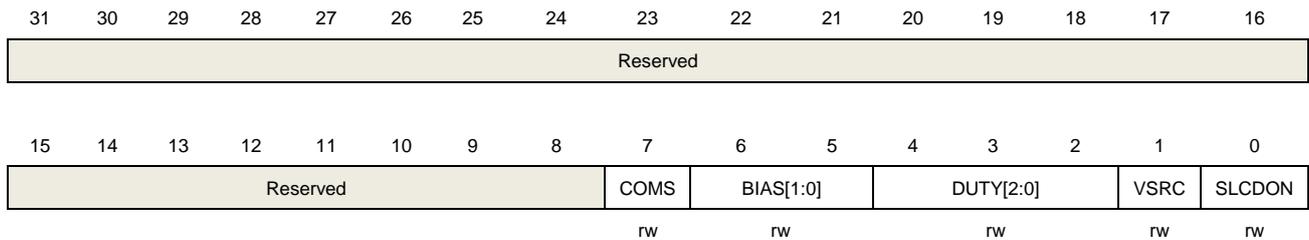
## 23.4. Register definition

### 23.4.1. Control register (SLCD\_CTL)

Address offset: 0x00

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)



Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value
7	COMS	common/segment pad select This bit is used to common/segment pad selection. When duty selects 1/8 or 1/6, SLCD_COM[7:4] pad is always select SLCD_COM[7:4] function whatever this bit is set or reset. 0: SLCD_COM[7:4] pad select SLCD_COM[7:4] 1: SLCD_COM[7:4] pad select SLCD_SEG[31:28]
6:5	BIAS[1:0]	Bias select Bias is the number of voltage levels used when driving a SLCD. It is defined as 1/(number of voltage levels used to drive an SLCD display - 1). 00: 1/4 Bias (5 voltage levels: VSS, 1/4VSLCD, 1/2VSLCD, 3/4VSLCD, VSLCD) 01: 1/2 Bias (3 voltage levels: VSS, 1/2VSLCD, VSLCD) 10: 1/3 Bias (4 voltage levels: VSS, 1/3VSLCD, 2/3VSLCD, VSLCD) 11: Reserved
4:2	DUTY[2:0]	Duty select These bits determine the duty cycle. Duty is the number defined as 1/(number of common terminals on a given SLCD display). 000: Static duty 001: 1/2 duty 010: 1/3 duty 011: 1/4 duty 100: 1/8 duty 101: 1/6 duty 110: Reserved 111: Reserved

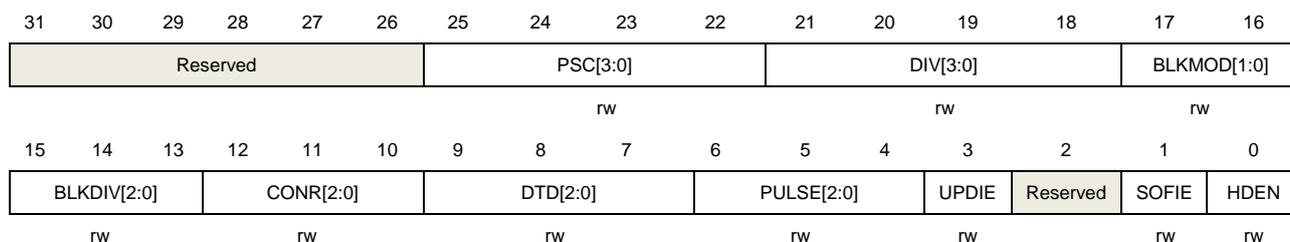
1	VSRC	SLCD Voltage source Set this bit determines which is the SLCD voltage source. 0: Internal source 1: External source (VSLCD pin)
0	SLCDON	SLCD controller start Set this bit by software to start SLCD controller. Clear this bit by software to stop SLCD controller and the SLCD controller stop at the beginning of the next frame. 0: SLCD Controller stop 1: SLCD Controller start

### 23.4.2. Configuration register (SLCD\_CFG)

Address offset: 0x04

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)



Bits	Fields	Descriptions
31:26	Reserved	Must be kept at reset value
25:22	PSC[3:0]	SLCD clock prescaler Set these bits define the prescaler of SLCD clock. 0000: $f_{PSC} = f_{in\_clk}$ 0001: $f_{PSC} = f_{in\_clk}/2$ 0010: $f_{PSC} = f_{in\_clk}/4$ ... 1111: $f_{PSC} = f_{in\_clk}/32768$
21:18	DIV[3:0]	SLCD clock divider Set these bits define the division factor of the DIV divider. 0000: $f_{SLCD} = f_{PSC}/16$ 0001: $f_{SLCD} = f_{PSC}/17$ 0010: $f_{SLCD} = f_{PSC}/18$ ... 1111: $f_{SLCD} = f_{PSC}/31$
17:16	BLKMOD[1:0]	Blink mode 00: No Blink

		01: Blink on SEG[0], COM[0] (1 pixel)
		10: Blink on SEG[0], all COMs (up to 8 pixels depending on the programmed duty)
		11: Blink on all SEGs and all COMs (all pixels)
15:13	BLKDIV[2:0]	Blink frequency divider
		000: $f_{BLINK} = f_{SLCD}/8$
		001: $f_{BLINK} = f_{SLCD}/16$
		010: $f_{BLINK} = f_{SLCD}/32$
		011: $f_{BLINK} = f_{SLCD}/64$
		100: $f_{BLINK} = f_{SLCD}/128$
		101: $f_{BLINK} = f_{SLCD}/256$
		110: $f_{BLINK} = f_{SLCD}/512$
		111: $f_{BLINK} = f_{SLCD}/1024$
12:10	CONR[2:0]	Contrast ratio
		When choosing the internal voltage source (VSRC=0), these bits specify the VSLCD voltage. It ranges from VSLCD0 to VSLCD7 (typical 2.75 V to 5.20V), Refer to the product datasheet for the VSLCDx values. When choosing the external voltage source (VSRC=1), these bits is invalid.
		000: VSLCD0
		001: VSLCD1
		010: VSLCD2
		011: VSLCD3
		100: VSLCD4
		101: VSLCD5
		110: VSLCD6
		111: VSLCD7
9:7	DTD[2:0]	Dead time duration
		Set these bits configure the length of the dead time between frames.
		000: No dead time
		001: 1 phase dead time
		010: 2 phase dead time
		...
		111: 7 phase dead time
6:4	PULSE[2:0]	Pulse ON duration
		Set these bits define the pulse duration in terms of PSC pulses.
		000: 0
		001: $1/f_{PSC}$
		010: $2/f_{PSC}$
		011: $3/f_{PSC}$
		100: $4/f_{PSC}$
		101: $5/f_{PSC}$
		110: $6/f_{PSC}$

111: 7/fPSC

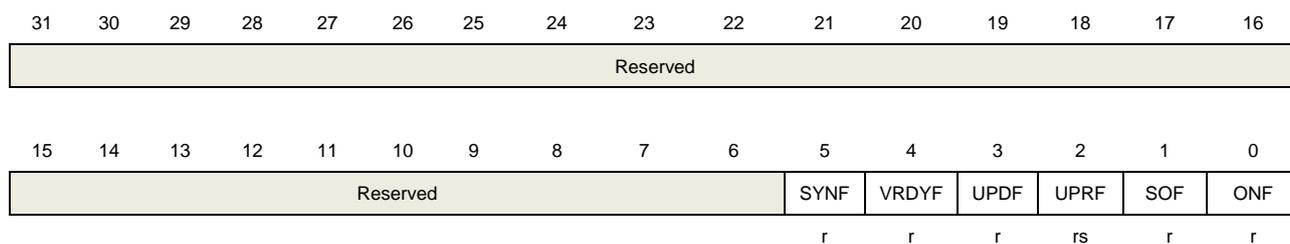
3	UPDIE	SLCD update done interrupt enable This bit is set and cleared by software. 0: SLCD Update Done interrupt disabled 1: SLCD Update Done interrupt enabled
2	Reserved	Must be kept at reset value
1	SOFIE	Start of frame interrupt enable This bit is set and cleared by software. 0: SLCD Start of Frame interrupt disabled 1: SLCD Start of Frame interrupt enabled
0	HDEN	High drive enable This bit is set and cleared by software. 0: Permanent high drive disabled. The time during which RL is enabled is configured by the PULSE[2:0]. 1: Permanent high drive enabled. RL is always switched on, and the PULSE[2:0] is invalid.

### 23.4.3. Status flag register (SLCD\_STAT)

Address offset: 0x08

Reset value: 0x0000 0020

This register has to be accessed by word(32-bit)



Bits	Fields	Descriptions
31:6	Reserved	Must be kept at reset value
5	SYNF	SLCD_CFG register synchronization flag This bit is set when SLCD_CFG register update to SLCD clock domain, and It is cleared by hardware when writing to the SLCD_CFG register. 0: SLCD_CFG Register not yet synchronized 1: SLCD_CFG Register synchronized to SLCD clock domain

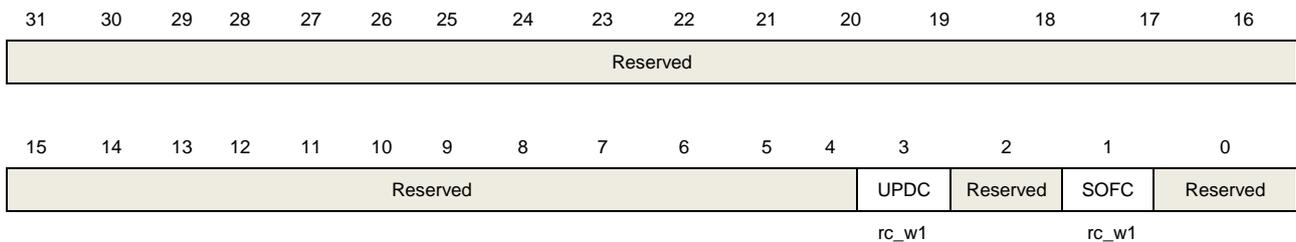
4	VRDYF	SLCD voltage ready flag This bit is set and cleared by the hardware according to the SLCD voltage. 0: SLCD voltage is not ready 1: Step-up converter is enabled and ready to provide the correct voltage
3	UPDF	Update SLCD data done flag This bit is set by hardware when update SLCD data done. It is cleared by writing 1 to the UPDC bit in the SLCD_STATC register. 0: No effect 1: SLCD data update done
2	UPRF	Update SLCD data request flag After modifying the the first buffer by the SLCD_DATAx registers, the application should set this bit to transfer the data to the second buffer. This bit stays set until the transfer is complete, the SLCD_DATAx register is write protected during this time. 0: No effect 1: Request SLCD data update
1	SOF	Start of frame flag This bit is set by hardware at the beginning of a new frame. It is cleared by writing 1 to the SOFC bit in the SLCD_STATC register. 0: No effect 1: Start of Frame flag
0	ONF	SLCD controller on flag This bit is set by hardware when SLCDON is set to 1, it is cleared by hardware after the SLCDON is cleared and the last frame is displayed. 0: SLCD Controller disabled. 1: SLCD Controller enabled

### 23.4.4. Status flag clear register (SLCD\_STATC)

Address offset: 0x0C

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)



Bits	Fields	Descriptions
31:4	Reserved	Must be kept at reset value

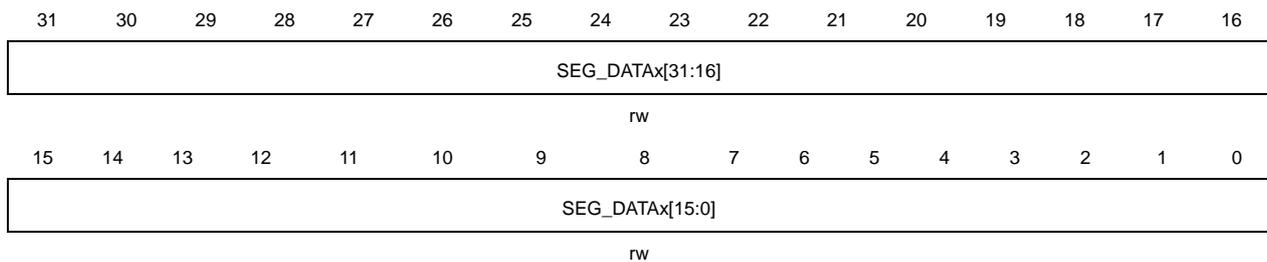
3	UPDC	SLCD data update done clear bit Set this bit to clear the UPDF flag in SLCD_STAT register. 0: No effect 1: Clear UPDF flag
2	Reserved	Must be kept at reset value
1	SOFC	Start of frame flag clear Set this bit to clear the SOF flag in the SLCD_STAT register. 0: No effect 1: Clear SOF flag
0	Reserved	Must be kept at reset value

### 23.4.5. Display data registers (SLCD\_DATAx, x=0~7)

Address offset:  $0x14+0x08*(x+1)$

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)



Bits	Fields	Descriptions
31:0	SEG_DATAx[31:0]	Each bit corresponds to one pixel to display. 0: Pixel inactive 1: Pixel active

## 24. Operational amplifiers (OPA)/ Programmable current and voltage reference (IVREF)

This chapter applies to GD32F170xx and GD32F190xx devices.

### 24.1. Operational amplifiers (OPA)

#### 24.1.1. Overview

There are three operational amplifiers in the MCU and the operational amplifiers can be able to route to external or internal follower.

When an operational amplifier is enabled, there will be a corresponding ADC channel used to output measurement outcome.

#### 24.1.2. Characteristics

- Rail-to-rail input and output voltage range
- Low input bias current
- Low input offset voltage
- Low power mode

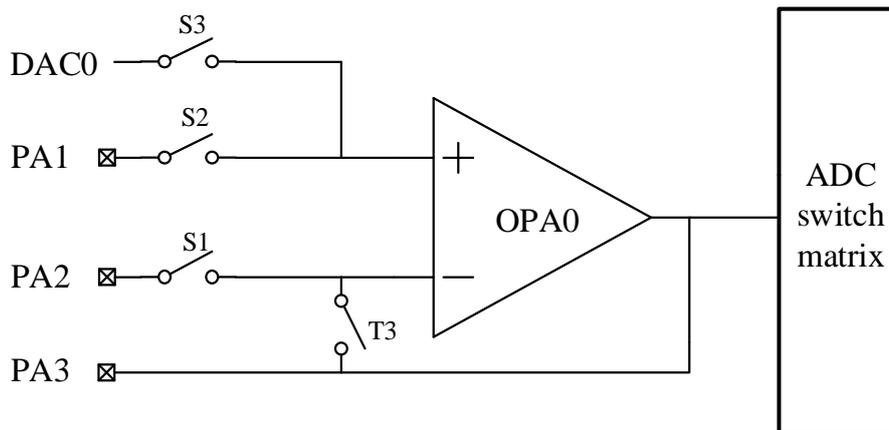
#### 24.1.3. Function overview

##### Signal routing

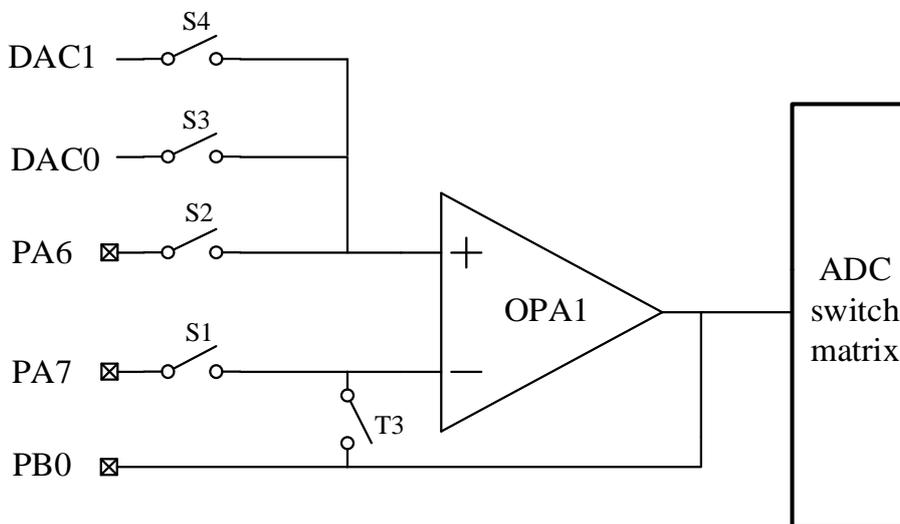
The connections with dedicated I/O are listed below:

- OPA0\_VINP—>PA1
- OPA0\_VINM—>PA2
- OPA0\_VOUT—>PA3
- OPA1\_VINP—>PA6
- OPA1\_VINM—>PA7
- OPA1\_VOUT—>PB0
- OPA2\_VINP—>PC1
- OPA2\_VINM—>PC2
- OPA2\_VOUT—>PC3

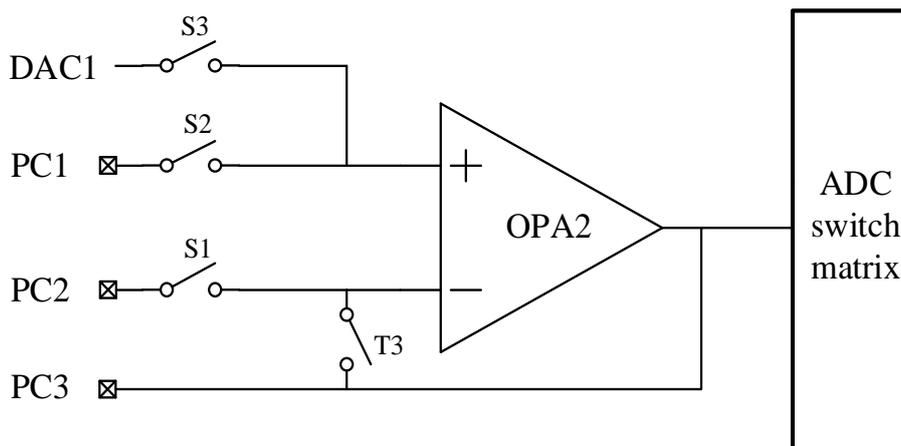
**Figure 24-1. OPA0 Signal Route**



**Figure 24-2. OPA1 Signal Route**



**Figure 24-3. OPA2 Signal Route**



OPA\_CTL register can select the routing for the three operational amplifiers. For OPA0-2, S3/S4 is used to connect DAC0/DAC1 to their positive input. The OPAx\_PD bit can be used to power down all operational amplifiers.

## Calibration

Before leaving factory, the default factory trimming values are stored in nonvolatile memory and used to initialize the operational amplifier offset. But also the chip support software using the user value to trim the operational amplifier offset.

During trimming operation, all switches related to the inputs of each operational amplifier must be disconnected.

Each operational amplifier has two operation modes: normal mode and low-power mode. The different mode can configure different user trimming value in different register. The trimming value is a 30-bit value for each mode of each operational amplifier. Writing 1 to OT\_USER bit will switch the trimming value to user configured values. This will take effect for all the operational amplifiers.

Following steps are the commendatory procedure for either one of operational amplifier:

- 1). Configure control register to disconnected all the switches to the operational amplifier
- 2). Set the OT\_USER bit to 1
- 3). Choose one calibration mode in below table:

For example: Operational amplifier-1, Normal mode offset cal low

Configure like this:

OPA0PD=0, OPA0LPM=0, OPA0CAL\_H=0, OPA0CAL\_L=1

- 4). Write trimming values in OPA0\_TRIM\_LOW. The writing value should be start from 00000b to the first value that causes the OPA0CALOUT bit set to 1.

**Note:** After writing trimming values into trimming register and before check CALOUT bit, software should wait for the  $t_{offtrimmax}$  delay specified in datasheet.

When got the correct trimming values, the values must be kept in trimming register.

Repeat step 3 and 4 for complete the whole operational amplifier calibration procedure:

- Normal mode calibration high,
- Low-power mode calibration low,
- Low-power mode calibration high.

**Note:** During the whole process of calibration, the external connection of operational amplifier output must not pull-up or pull-down current higher than 500uA.

**Table 24-1. Operating mode and calibration**

Mode	Control Bits				Output	
	OPAxPD	OPAxLPM	OPAxCAL_H	OPAxCAL_L	V <sub>OUT</sub>	CALOUT
Normal	0	0	0	0	analog	0
			1	1		
Low-power	0	1	0	0	analog	0
			1	1		
Power-down	1	X	X	X	Z	0

<b>Offset cal-high</b>	0	X	1	0	analog	X
<b>Offset cal-low</b>	0	X	0	1	analog	X

## 24.1.4. Register definition

### Control register (OPA\_CTL)

Address offset: 0x5C

Reset value: 0x0001 0101

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OPA2CAL OUT	OPA1CAL OUT	OPA0CAL OUT	OPA_ RANG E	S4OPA 1	Reserved			OPA2 LPM	OPA2 CAL_H	OPA2CA L_L	S3O PA2	S2O PA2	S1O PA2	T3O PA2	OPA2 PD
r	r	r	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
OPA1LP M	OPA1CAL _H	OPA1CAL _L	S3OP A1	S2OP A1	S1OP A1	T3OP A1	OPA 1PD	OPA0 LPM	OPA0CA L_H	OPA0 CAL_L	S3O PA0	S2O PA0	S1O PA0	T3O PA0	OPA0 PD
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31	OPA2CALOUT	OPA2 calibration output When in calibration mode, if offset is trimmed the flag will assert.
30	OPA1CALOUT	OPA1 calibration output When in calibration mode, if offset is trimmed the flag will assert.
29	OPA0CALOUT	OPA0 calibration output When in calibration mode, if offset is trimmed the flag will assert.
28	OPA_RANGE	Power supply range This bit can be set and cleared by software when the operational amplifiers are in powered down. It select the operational amplifier power supply range for stability 0: Low range ( $V_{DDA} < 3.3\text{ V}$ ) 1: High range ( $V_{DDA} > 3.3\text{ V}$ )
27	S4OPA1	S4 switch enable for OPA1 0: S4 switch opened 1: S4 switch closed
26:24	Reserved	Must be kept at reset value
23	OPA2LPM	OPA2 low power mode 0: OPA2 low power mode off 1: OPA2 low power mode on
22	OPA2CAL_H	OPA2 offset calibration for N diff 0: OPA2 offset calibration for N diff OFF 1: OPA2 offset calibration for N diff ON if OPA2CAL_L = 0

21	OPA2CAL_L	OPA2 offset calibration for P diff 0: OPA2 offset calibration for P diff OFF 1: OPA2 offset calibration for P diff ON if OPA2CAL_H = 0
20	S3OPA2	S3 switch enable for OPA2 0: S3 switch opened 1: S3 switch closed
19	S2OPA2	S2 switch enable for OPA2 0: S2 switch opened 1: S2 switch closed
18	S1OPA2	S1 switch enable for OPA2 0: S1 switch opened 1: S1 switch closed
17	T3OPA2	T3 switch enable for OPA2 0: T3 switch opened 1: T3 switch closed
16	OPA2PD	OPA2 power down 0: OPA2 enabled 1: OPA2 disabled
15	OPA1LPM	OPA1 low power mode 0: OPA1 low power mode off 1: OPA1 low power mode on
14	OPA1CAL_H	OPA1 offset calibration for N diff 0: OPA1 offset calibration for N diff OFF 1: OPA1 offset calibration for N diff ON if OPA1CAL_L = 0
13	OPA1CAL_L	OPA1 offset calibration for P diff 0: OPA1 offset calibration for P diff OFF 1: OPA1 offset calibration for P diff ON if OPA1CAL_H = 0
12	S3OPA1	S3 switch enable for OPA1 0: S3 switch opened 1: S3 switch closed
11	S2OPA1	S2 switch enable for OPA1 0: S2 switch opened 1: S2 switch closed
10	S1OPA1	S1 switch enable for OPA1 0: S1 switch opened 1: S1 switch closed
9	T3OPA1	T3 switch enable for OPA1 0: T3 switch opened

		1: T3 switch closed
8	OPA1PD	OPA1 power down 0: OPA1 enabled 1: OPA1 disabled
7	OPA0LPM	OPA0 low power mode 0: OPA0 low power mode off 1: OPA0 low power mode on
6	OPA0CAL_H	OPA0 offset calibration for N diff 0: OPA0 offset calibration for N diff OFF 1: OPA0 offset calibration for N diff ON if OPA0CAL_L = 0
5	OPA0CAL_L	OPA0 offset calibration for P diff 0: OPA0 offset calibration for P diff OFF 1: OPA0 offset calibration for P diff ON if OPA0CAL_H = 0
4	S3OPA0	S3 switch enable for OPA0 0: S3 switch opened 1: S3 switch closed
3	S2OPA0	S2 switch enable for OPA0 0: S2 switch opened 1: S2 switch closed
2	S1OPA0	S1 switch enable for OPA0 0: S1 switch opened 1: S1 switch closed
1	T3OPA0	T3 switch enable for OPA0 0: T3 switch opened 1: T3 switch closed
0	OPA0PD	OPA0 power down 0: OPA0 enabled 1: OPA0 disabled

### Bias trimming register for normal mode (OPA\_BT)

Address offset: 0x60

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OT_USE R	Reserved	OA2_TRIM_HIGH[4:0]				OA2_TRIM_LOW[4:0]				OA1_TRIM_HIGH[4:1]					
w		rw				rw				rw					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OA1_ TRIM_HI GH[0]	OA1_TRIM_LOW[4:0]				OA0_TRIM_HIGH[4:0]				OA0_TRIM_LOW[4:0]						
rw	rw				rw				rw						

Bits	Fields	Descriptions
31	OT_USER	This bit is set and cleared by software; it is always read as 0. It is used to select if the OPAx offset is trimmed by the preset factory-programmed trimming values or the user programmed trimming value. 0: Trim the OPA offset using default factory values 1: Trim the OPA offset using user programmed values
30	Reserved	Must be kept at reset value
29:25	OA2_TRIM_HIGH[4:0]	OPA2, normal mode 5-bit offset trim value for NMOS pairs
24:20	OA2_TRIM_LOW[4:0]	OPA2, normal mode 5-bit offset trim value for PMOS pairs
19:15	OA1_TRIM_HIGH[4:0]	OPA1, normal mode 5-bit offset trim value for NMOS pairs
14:10	OA1_TRIM_LOW[4:0]	OPA1, normal mode 5-bit offset trim value for PMOS pairs
9:5	OA0_TRIM_HIGH[4:0]	OPA0, normal mode 5-bit offset trim value for NMOS pairs
4:0	OA0_TRIM_LOW[4:0]	OPA0, normal mode 5-bit offset trim value for PMOS pairs

### Bias trimming register for low power mode (OPA\_LPBT)

Address offset: 0x64

Reset value: 0x0000 0000

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		OA2_TRIM_LP_HIGH[4:0]				OA2_TRIM_LP_LOW[4:0]				OA1_TRIM_LP_HIGH[4:1]					
		rw				rw				rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OA1_ TRIM_ LP_HIGH[0]	OA1_TRIM_LP_LOW[4:0]				OA0_TRIM_LP_HIGH[4:0]				OA0_TRIM_LP_LOW[4:0]						
rw	rw				rw				rw						

Bits	Fields	Descriptions
31:30	Reserved	Must be kept at reset value
29:25	OA2_TRIM_LP_HIG H[4:0]	OPA2, low-power mode 5-bit offset trim value for NMOS pairs
24:20	OA2_TRIM_ LP_LOW[4:0]	OPA2, low-power mode 5-bit offset trim value for PMOS pairs

19:15	OA1_TRIM_ LP_HIGH[4:0]	OPA1, low-power mode 5-bit offset trim value for NMOS pairs
14:10	OA1_TRIM_ LP_LOW[4:0]	OPA1, low-power mode 5-bit offset trim value for PMOS pairs
9:5	OA0_TRIM_ LP_HIGH[4:0]	OPA0, low-power mode 5-bit offset trim value for NMOS pairs
4:0	OA0_TRIM_ LP_LOW[4:0]	OPA0, low-power mode 5-bit offset trim value for PMOS pairs

## 24.2. Programmable Current and Voltage Reference (IVREF)

### 24.2.1. Overview

An application programmable current reference module is included in the MCU.

When users want to use current reference, there are two different running modes are supplied.

One mode named Low Power Mode and another one named High Current Mode.

The difference between two modes is the current step and maximum current.

The former's (LPM) step current is 1uA and the latter's (HCM) 8uA.

The former's (LPM) maximum current is 63uA and the latter's (HCM) 504uA.

There is still a voltage reference module in the MCU which can provide a 2.5V voltage.

### 24.2.2. Characteristics

Current Reference feature:

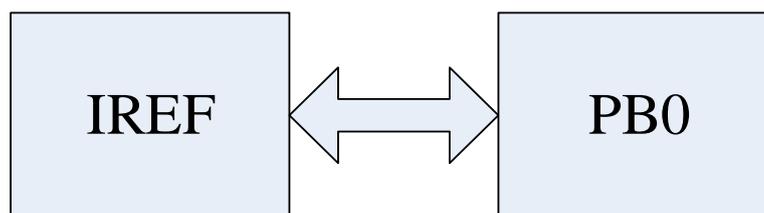
- Programmable current
- Programmable Source or sink
- Low Power Mode and High Current Mode

Voltage Reference feature:

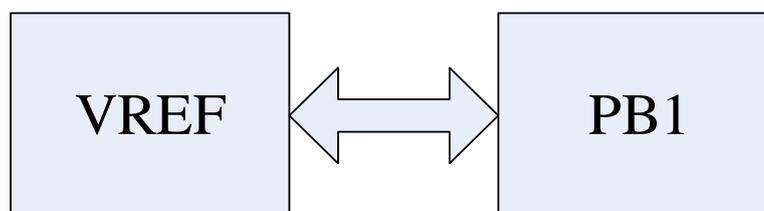
- User programmable trim

### 24.2.3. Function overview

#### Signal routing



When IREF is used, the PB0 pin should be configured to analog input mode.



When VREF is used, the PB1 pin should be configured as an analog I/O and external VREF decoupling capacitors are recommended.

### **User Trimming**

User can trim the IREF outputting current by programming IREF\_TRIM and can trim the voltage by programming VREF\_TRIM.

## 24.2.4. Register definition

### Control register (IVREF\_CTL)

Address offset: 0x300

Reset value: 0x1000 0F00

This register has to be accessed by word (32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
VREN	DECAP	Reserved	VPT[4:0]				Reserved								
rw	rw		rw												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CREN	SSEL	Reserved	CPT[4:0]				SCMOD	Reserved	CSDT[5:0]						
rw	rw		rw				rw		rw						

Bits	Fields	Descriptions
31	VREN	Voltage Reference Enable 0: Disable Voltage Reference 1: Enable Voltage Reference
30	DECAP	Disconnect external capacitor 0: Connect external capacitor 1: Disconnect external capacitor
29	Reserved	Must be kept at reset value
28:24	VPT[4:0]	Voltage precision trim 00000: -6.4% 00001: -6.0% .... 11110: +5.6% 11111: +6%
23:16	Reserved	Must be kept at reset value
15	CREN	Current Reference Enable. 0: Disable current reference 1: Enable current reference
14	SSEL	Step selection 0: Low power, 1uA step 1: High current, 8uA step
13	Reserved	Must be kept at reset value
12:8	CPT[4:0]	Current precision trim 00000: -15% ....

		11111: +16%
7	SCMOD	Sink current mode 0: Source current 1: Sink current
6	Reserved	Must be kept at reset value
5:0	CSDT[5:0]	Current step data 000000: Default value. 000001: Step * 1 .... 111111: Step * 63

## 25. Controller area network (CAN)

This chapter applies to GD32F170xx and GD32F190xx devices.

### 25.1. Overview

CAN bus (for controller area network) is a bus standard designed to allow microcontrollers and devices to communicate with each other without a host computer.

The Basic Extended CAN (CAN), interfaces the CAN network. It supports the CAN protocols version 2.0A and B. The CAN interface handles the transmission and the reception of CAN frames fully autonomously. The CAN provides 28 scalable/configurable identifier filter banks. The filters are for selecting the incoming messages the software needs and discarding the others. Three transmit mailboxes are provided to the software for setting up messages. The transmission Scheduler decides which mailbox has to be transmitted first. Three complete messages can be stored in each FIFO. The FIFOs are managed completely by hardware. Two receive FIFOs are used by hardware to store the incoming messages. The CAN controller also provides all hardware functions for supporting the time-triggered communication option for safety-critical applications.

### 25.2. Characteristics

- Supports CAN protocols version 2.0A, B
- Baud rates up to 1 Mbit/s
- Supports the time-triggered communication
- Maskable interrupts

#### Transmission

- 3 transmit mailboxes
- Prioritization of messages
- Time Stamp on SOF transmission

#### Reception

- 2 receive FIFOs with 3 messages deep
- 28 scalable/configurable identifier filter banks
- FIFO lock

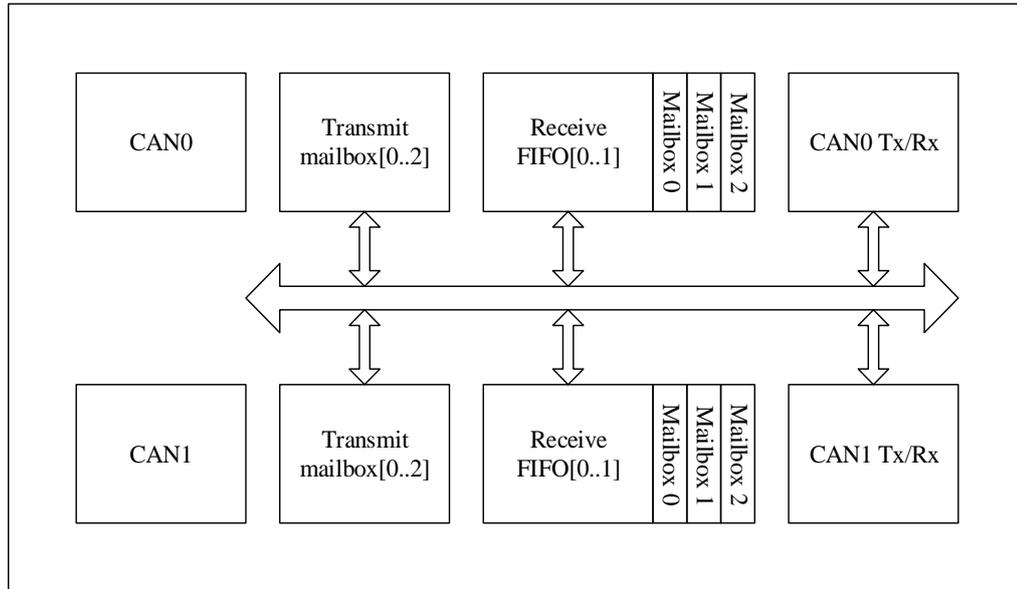
#### Time-triggered communication

- Disable retransmission automatically
- 16-bit free timer
- Time Stamp on SOF reception
- Time Stamp sent in last two data bytes

### 25.3. Function overview

[Figure 25-1. CAN module block diagram](#) shows the CAN block diagram.

**Figure 25-1. CAN module block diagram**



#### 25.3.1. Working mode

The CAN interface has three working modes:

- Sleep working mode
- Initial working mode
- Normal working mode

##### Sleep working mode

Sleep working mode is the default mode after reset. In sleep working mode, the CAN is in the low-power status while the CAN clock is stopped.

When SLPWMOD bit in CAN\_CTL register is set, the CAN enters the sleep working mode. Then the SLPWS bit in CAN\_STAT register is set.

To leave sleep working mode automatically: the AWU bit in CAN\_CTL register is set and the CAN bus activity is detected. To leave sleep working mode by software: clear the SLPWMOD bit in CAN\_CTL register.

Sleep working mode to Initial working mode: Set IWMOD bit and clear SLPWMOD bit in CAN\_CTL register.

Sleep working mode to Normal working mode: Clear IWMOD and SLPWMOD bit in CAN\_CTL

register.

## Initial working mode

When the options of CAN bus communication is needed to be changed, the CAN must enter initial working mode.

When IWMOD bit in CAN\_CTL register is set, the CAN enters the initial working mode. Then the IWS bit in CAN\_STAT register is set.

Initial working mode to sleep working mode: Set SLPWMOD bit and clear IWMOD bit in CAN\_CTL register.

Initial working mode to Normal working mode: Clear IWMOD bit and clear SLPWMOD bit in CAN\_CTL register.

## Normal working mode

The CAN can enter normal working mode and to communicate with other CAN communication nodes.

To enter normal working mode: clear IWMOD and SLPWMOD bit in CAN\_CTL register.

Normal working mode to sleep working mode: Set SLPWMOD bit in CAN\_CTL register and wait the current transmission or reception completed.

Normal working mode to Initial working mode: Set IWMOD bit in CAN\_CTL register, and wait the current transmission or reception completed.

## 25.3.2. Communication modes

The CAN interface has four communication modes:

- Silent communication mode
- Loopback communication mode
- Loopback and silent communication mode
- Normal communication mode

### Silent communication mode

Silent communication mode means reception available and transmission disable.

The Rx pin of the CAN can get the signal from the network and the Tx pin always holds logical one.

When the SCMOD bit in CAN\_BT register is set, the CAN enters the silent communication mode. When it is cleared, the CAN leaves silent communication mode.

Silent communication mode is useful on monitoring the network messages.

### Loopback communication mode

Loopback communication mode means the sending messages are transferred into the reception FIFOs.

Set LCMOD bit in CAN\_BT register to enter loopback communication mode or clear it to leave.

Loopback communication mode is useful on self-test.

### Loopback and silent communication mode

Loopback and silent communication mode means the RX and TX pins are disconnected from the CAN network while the sending messages are transformed into the reception FIFOs.

Set LCMOD and SCMOD bit in CAN\_BT register to enter loopback and silent communication mode or clear them to leave.

Loopback and silent communication mode is useful on self-test. The TX pin holds logical one. The RX pin holds high impedance state.

### Normal communication mode

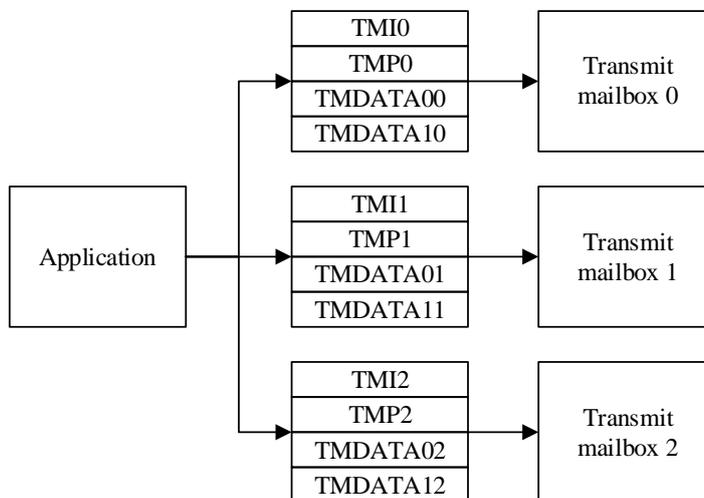
Normal communication mode is the default communication mode unless the LCMOD or SCMOD bit in CAN\_BT register is set.

## 25.3.3. Data transmission

### Transmission register

Three transmit mailboxes are transparent to the application. You can use transmit mailboxes through four registers: CAN\_TMIx, CAN\_TMPx, CAN\_TMDATA0x and CAN\_TMDATA1x. As shown in [Figure 25-2. Transmission register](#).

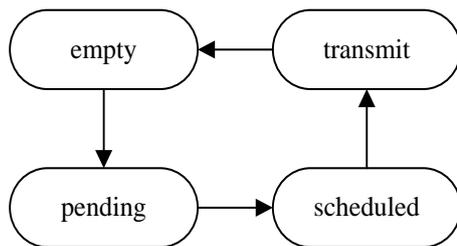
Figure 25-2. Transmission register



### Transmit mailbox state

A transmit mailbox can be used when it is free: **empty** state. If the data is filled in the mailbox, setting TEN bit in CAN\_TMIx register to prepare for starting the transmission: **pending** state. If more than one mailbox is in the pending state, they need schedule the transmission: **scheduled** state. A mailbox with priority enter **transmit** state and start transmitting the message. After the message has been sent, the mailbox is free: **empty** state. As shown in [Figure 25-3. State of transmission mailbox](#).

**Figure 25-3. State of transmission mailbox**



### Transmit status and error

The CAN\_TSTAT register includes the transmit status and error bits: MTF, MTFNERR, MAL, MTE.

- MTF: mailbox transmits finished. Typically, MTF is set when the frame in the transmit mailbox has been sent.
- MTFNERR: mailbox transmits finished and no error. MTFNERR is set when the frame in the transmission mailbox has been sent.
- MAL: mailbox arbitration lost. MAL is set while the frame transmission is failed because of the arbitration lost.
- MTE: mailbox transmits error. MTE is set while the frame transmission is failed because of the detection error of CAN bus.

### Steps of sending a frame

To send a frame through the CAN:

Step 1: Select one free transmit mailbox.

Step 2: Fill four registers with the application's acquirement.

Step 3: Set TEN bit in CAN\_TMIx register.

Step 4: Check the transmit status. Typically, MTF and MTFNERR are set if transmission is successful.

## Transmission options

### Abort

MST bit in CAN\_TSTAT register can abort the transmission.

If the transmission mailbox's state is **pending** or **scheduled**, the abort of transmission can be done immediately.

In the state of **transmit** the abort of transmission has two results. In case of transmission successful, the MTFNERR and MTF in CAN\_TSTAT are set and state changes to **empty**. In case of transmission failed, the state changes to be **scheduled** and then the abort of transmission can be done immediately.

### Priority

When more than one transmit mailbox is pending, the transmission order is given by the TFO bit in CAN\_CTL register.

In case TFO is 1, the three transmit mailboxes work as FIFO.

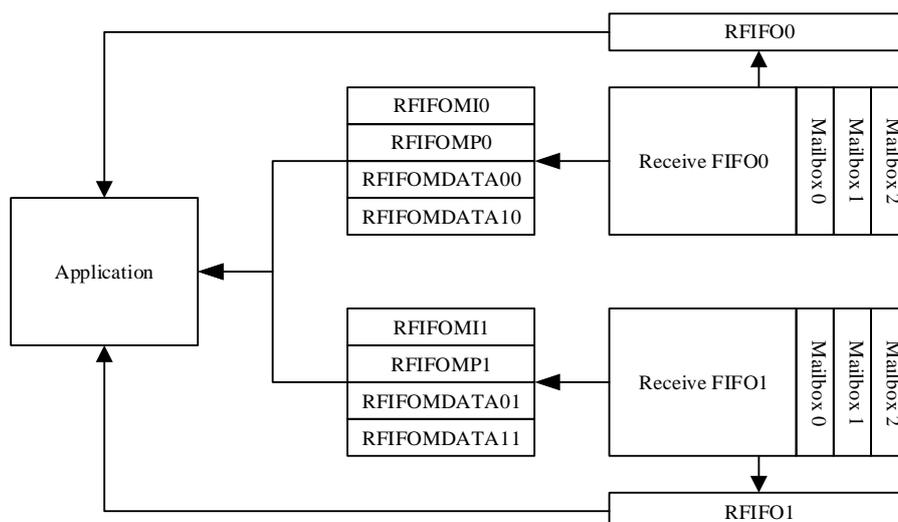
In case TFO is 0, the transmit mailbox with lowest identifier has the highest priority of transmission. If the identifiers are equal, the lower mailbox number will be scheduled first.

## 25.3.4. Data reception

### Reception register

Two receive FIFOs are transparent to the application. You can use receive FIFOs through five registers: CAN\_RFIFOx, CAN\_RFIFOMIx, CAN\_RFIFOMPx, CAN\_RFIFOMDATA0x and CAN\_RFIFOMDATA1x. FIFO's status and operation can be handled by CAN\_RFIFOx register. Reception frame data can be achieved through the registers: CAN\_RFIFOMIx, CAN\_RFIFOMPx, CAN\_RFIFOMDATA0x and CAN\_RFIFOMDATA1x.

Each FIFO consists of three receive mailboxes. As shown in [Figure 25-4. Reception register](#).

**Figure 25-4. Reception register**


## Receive FIFO

Receive FIFO has three mailboxes. The reception frames are stored in the mailbox ordered by the arriving sequence of the frames. First arrived frame can be accessed by application firstly.

The number of frames in the receive FIFO and the status can be accessed by the register CAN\_RFIFO0 and CAN\_RFIFO1.

If at least one frame has been stored in the receive FIFO0. The frame data is placed in the registers (CAN\_RFIFOMI0, CAN\_RFIFOMP0, CAN\_RFIFOMDATA00, CAN\_RFIFOMDATA10). After read the current frame, set RFD bit in CAN\_RFIFO0 to release a frame in the receive FIFO and the software can read the next frame.

## Receive FIFO status

RFL bit in CAN\_RFIFOx register: receive FIFO length. It is 0 when no frame is stored in the reception FIFO and 3 when FIFOx is full.

RFF bit in CAN\_RFIFOx register: the FIFO holds three frames. It indicates FIFOx is full.

RFO bit in CAN\_RFIFOx register: one new frame arrived while the FIFO has hold three frames. It indicates FIFOx is overfull. If the RFOD bit in CAN\_CTL register is set, the new frame is discarded. If the RFOD bit in CAN\_CTL register is reset, the new frame is stored into the receive FIFO and the last frame in the receive FIFO is discarded.

## Steps of receiving a message

Step 1: Check the number of frames in the receive FIFO.

Step 2: Reading CAN\_RFIFOMIx, CAN\_RFIFOMPx, CAN\_RFIFOMDATA0x and CAN\_RFIFOMDATA1x if there is data pending.

Step 3: Set the RFD bit in CAN\_RFIFOx register.

### 25.3.5. Filtering Function

The CAN would receive frames from the CAN bus. If the frame is passed through the filter, it is copied into the receive FIFOs. Otherwise, the frame will be discarded without intervention by the software.

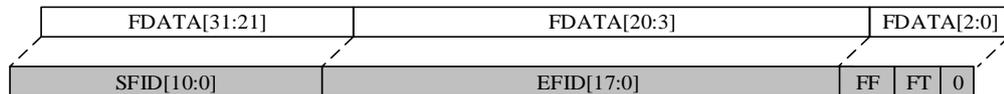
The identifier of frame from the CAN bus takes part in the matching of the filter.

#### Scale

Each filter bank can be configured 32-bit or 16-bit.

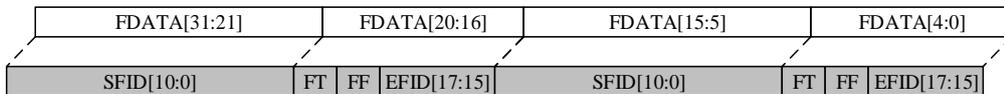
32-bit: SFID[10:0], EFID[17:0], FF and FT bits. As shown in [Figure 25-5. 32-bit filter](#).

**Figure 25-5. 32-bit filter**



16-bit: SFID [10:0], FT, FF and EFID[17:15] bits. As shown in [Figure 25-6. 16-bit filter](#).

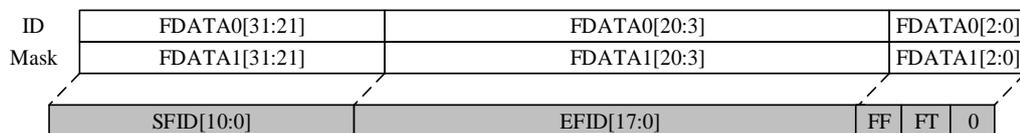
**Figure 25-6. 16-bit filter**



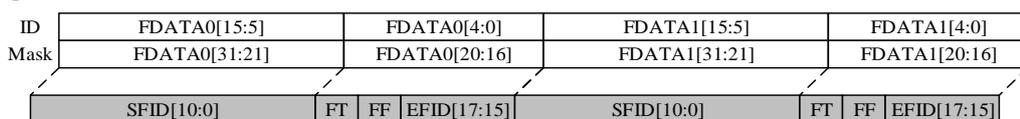
#### Mask mode

In mask mode the identifier registers are associated with mask registers specifying which bits of the identifier are handled as “must match” or as “don’t care”. 32-bit mask mode example is shown in [Figure 25-7. 32-bit mask mode filter](#).

**Figure 25-7. 32-bit mask mode filter**



**Figure 25-8. 16-bit mask mode filter**



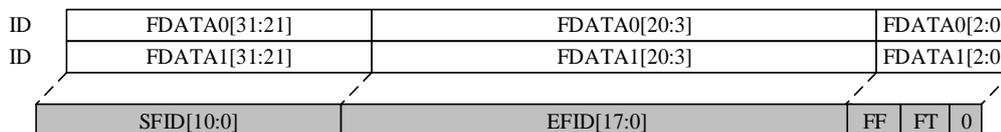
#### List mode

The filter consists of frame identifiers. The filter can decide whether a frame will be discarded or not. When one frame arrived, the filter will check which member can match the identifier of

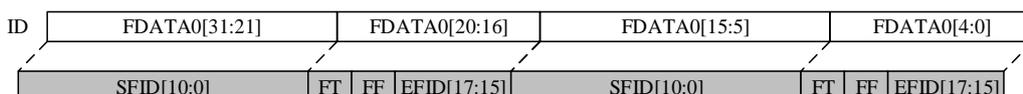
the frame.

32-bit list mode example is shown in [Table 25-1. 32-bit filter number](#).

**Figure 25-9. 32-bit list mode filter**



**Figure 25-10. 16-bit list mode filter**



### Filter number

Each filter within a filter bank is numbered from 0 to a maximum dependent on the mode and the scale of each of the filter banks.

For example, there are two filter banks. Bank 0 is configured as 32-bit mask mode. Bank 1 is configured as 32-bit list mode. The filter number is shown in [Table 25-1. 32-bit filter number](#).

**Table 25-1. 32-bit filter number**

Filter Bank	Filter Data Register	Filter Number
0	F0DATA0-32bit-ID	0
	F0DATA1-32bit-Mask	
1	F1DATA0-32bit-ID	1
	F1DATA1-32bit-ID	2

### Associated FIFO

28 banks can be associated with FIFO0 or FIFO1. If the bank is associated with FIFO 0, the frames passed through the bank will fill the FIFO0.

### Active

The filter bank needs to be configured activation if the application wants the bank working and while filters not used by the application should be left deactivated.

### Filtering index

Each filter number corresponds to a filtering rule. When the frame from the CAN bus passes the filters, a filter number must associate with the frame. The filter number is called filtering index. It stores in the FI bit in CAN\_RFIFOMPx when the frame is read by the application.

Each FIFO numbers the filters within the banks associated with the FIFO itself whatever the bank is active or not.

The example about filtering index is shown in [Table 25-2. Filtering index](#).

**Table 25-2. Filtering index**

Filter Bank	FIFO0	Active	Filter Number	Filter Bank	FIFO1	Active	Filter Number
0	F0DATA0-32bit-ID	Yes	0	2	F2DATA0[15:0]-16bit-ID	Yes	0
	F0DATA1-32bit-Mask				F2DATA0[32:16]-16bit-Mask		
1	F1DATA0-32bit-ID	Yes	1		F2DATA1[15:0]-16bit-ID		1
	F1DATA1-32bit-ID		2		F2DATA1[32:16]-16bit-Mask		
3	F3DATA0[15:0]-16bit-ID	No	3	4	F4DATA0-32bit-ID	No	2
	F3DATA0[32:16]-16bit-Mask				F4DATA1-32bit-Mask		
	F3DATA1[15:0]-16bit-ID		4	5	F5DATA0-32bit-ID	No	3
	F3DATA1[32:16]-16bit-Mask				F5DATA1-32bit-ID		4
7	F7DATA0[15:0]-16bit-ID	No	5	6	F6DATA0[15:0]-16bit-ID	Yes	5
	F7DATA0[32:16]-16bit-Mask		6		F6DATA0[32:16]-16bit-Mask		6
	F7DATA1[15:0]-16bit-ID		7		F6DATA1[15:0]-16bit-ID		7
	F7DATA1[32:16]-16bit-Mask		8		F6DATA1[32:16]-16bit-Mask		8
8	F8DATA0[15:0]-16bit-ID	Yes	9	10	F10DATA0[15:0]-16bit-ID	No	9
	F8DATA0[32:16]-16bit-Mask		10		F10DATA0[32:16]-16bit-Mask		
	F8DATA1[15:0]-16bit-ID		11		F10DATA1[15:0]-16bit-ID		10
	F8DATA1[32:16]-16bit-Mask		12		F10DATA1[32:16]-16bit-Mask		
9	F9DATA0[15:0]-16bit-ID	Yes	13	11	F11DATA0[15:0]-16bit-ID	No	11
	F9DATA0[32:16]-16bit-Mask				13		F11DATA0[32:16]-16bit-Mask
	F9DATA1[15:0]-16bit-ID		14		F11DATA1[15:0]-16bit-ID		13
	F9DATA1[32:16]-16bit-Mask				F11DATA1[32:16]-16bit-Mask		14
12	F12DATA0-32bit-ID	Yes	15	13	F13DATA0-32bit-ID	Yes	15
	F12DATA1-32bit-Mask				F13DATA1-32bit-Mask		16

### Priority

The filters have the priority:

- 32-bit mode is higher than 16-bit mode.

2. List mode is higher than mask mode.
3. Smaller filter index value has the higher priority.

### 25.3.6. Time-triggered communication

The time-triggered CAN protocol is a higher layer protocol on top of the CAN (Controller Area Network) data link layer. Time-triggered communication means that activities are triggered by the elapsing of time segments. In a time-triggered communication system all points of time of message transmission are defined during the development of a system. A time-triggered communication system is ideal for applications in which the data traffic is of a periodic nature.

In this mode, the 16-bit internal counter of the CAN hardware is activated and used to generate the time stamp value stored in the CAN\_RFIFOMPx and CAN\_TMPx registers for reception and transmission respectively. The internal counter is incremented each CAN bit time. The internal counter is captured on the sample point of the SOF (Start of Frame) bit in both reception and transmission.

The automatic retransmission is disabled in the time-triggered CAN communication.

### 25.3.7. Communication parameters

#### Nonautomatic retransmission mode

This mode has been implemented in order to fulfill the requirement of the time-triggered communication option of the CAN standard. To configure the hardware in this mode the ARD bit in the CAN\_CTL register must be set.

In this mode, each transmission is started only once. If the first attempt fails, due to an arbitration loss or an error, the hardware will not automatically restart the frame transmission.

At the end of the first transmission attempt, the hardware considers the request as finished and sets the MTF bit in the CAN\_TSTAT register. The result of the transmission is indicated in the CAN\_TSTAT register by the MTFNERR, MAL and MTE bits.

#### Bit time

On the bit-level the CAN protocol uses synchronous bit transmission. This not only enhances the transmitting capacity but also means that a sophisticated method of bit synchronization is required. While bit synchronization in a character-oriented transmission (asynchronous) is performed upon the reception of the start bit available with each character, a synchronous transmission protocol there is just one start bit available at the beginning of a frame. To ensure the receiver to correctly read the messages, continuous resynchronization is required. Phase buffer segments are therefore inserted before and after the nominal sample point within a bit interval.

The CAN protocol regulates bus access by bit-wise arbitration. The signal propagation from sender to receiver and back to the sender must be completed within one bit-time. For

synchronization purposes a further time segment, the propagation delay segment, is needed in addition to the time reserved for synchronization, the phase buffer segments. The propagation delay segment takes into account the signal propagation on the bus as well as signal delays caused by transmitting and receiving nodes.

The normal bit time simplified by the CAN from the CAN protocol has three segments as follows:

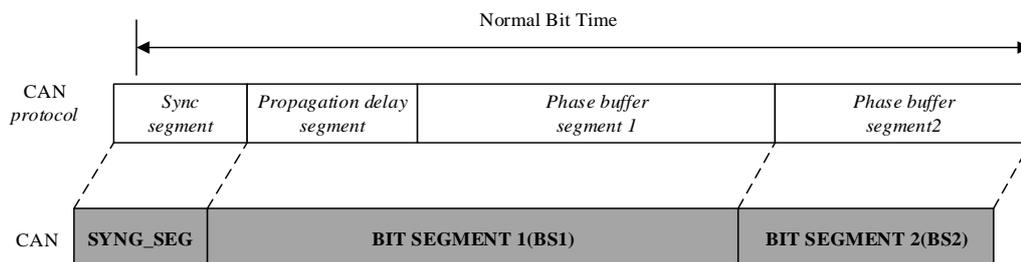
**Synchronization segment (SYNC\_SEG):** a bit change is expected to occur within this time segment. It has a fixed length of one time quantum ( $1 \times t_{CAN}$ ).

**Bit segment 1 (BS1):** defines the location of the sample point. It includes the *Propagation delay segment* and *Phase buffer segment 1* of the CAN standard. Its duration is programmable between 1 and 16 time quanta but may be automatically lengthened to compensate for positive phase drifts due to differences in the frequency of the various nodes of the network.

**Bit segment 2 (BS2):** defines the location of the transmit point. It represents the *Phase buffer segment 2* of the CAN standard. Its duration is programmable between 1 and 8 time quanta but may also be automatically shortened to compensate for negative phase drifts.

The bit time is shown as in the figure below.

**Figure 25-11. The bit time**



The resynchronization Jump Width (SJW) defines an upper bound to the amount of lengthening or shortening of the bit segments. It is programmable between 1 and 4 time quanta.

A valid edge is defined as the first transition in a bit time from dominant to recessive bus level provided the controller itself does not send a recessive bit.

If a valid edge is detected in BS1 instead of SYNC\_SEG, BS1 is extended by up to SJW so that the sample point is delayed.

Conversely, if a valid edge is detected in BS2 instead of SYNC\_SEG, BS2 is shortened by up to SJW so that the transmit point is moved earlier.

## Baud Rate

The CAN's clock derives from the APB1 bus. The CAN calculates its baud rate as follow:

$$BaudRate = \frac{1}{Normal\ Bit\ Time}$$

$$Normal\ Bit\ Time = t_{SYNC\_SEG} + t_{BS1} + t_{BS2}$$

with:

$$t_{SYNC\_SEG} = 1 \times t_q$$

$$t_{BS1} = (1 + BTR.TS1) \times t_q$$

$$t_{BS2} = (1 + BTR.TS2) \times t_q$$

$$t_q = (1 + BTR.BRP) \times t_{PCLK1}$$

### 25.3.8. Error flags

The error management as described in the CAN protocol is handled entirely by hardware using a Transmit Error Counter (TECNT value, in CAN\_ERR register) and a Receive Error Counter (RECNT value, in the CAN\_ERR register), which get incremented or decremented according to the error condition. For detailed information about TECNT and RECNT management, please refer to the CAN standard.

Both of them may be read by software to determine the stability of the network.

Furthermore, the CAN hardware provides detailed information on the current error status in CAN\_ERR register. By means of the CAN\_INTEN register (ERRIE bit, etc.), the software can configure the interrupt generation on error detection in a very flexible way.

#### Bus-Off recovery

The Bus-Off state is reached when TECNT is greater than 255. This state is indicated by BOERR bit in CAN\_ERR register. In Bus-Off state, the CAN is no longer able to transmit and receive messages.

Depending on the ABOR bit in the CAN\_CTL register, CAN will recover from Bus-Off (becomes error active again) either automatically or on software request. But in both cases the CAN has to wait at least for the recovery sequence specified in the CAN standard (128 occurrences of 11 consecutive recessive bits monitored on CAN RX).

If ABOR is set, the CAN will start the recovering sequence automatically after it has entered Bus-Off state.

If ABOR is cleared, the software must initiate the recovering sequence by requesting CAN to enter and to leave initialization mode.

**Note:** If the Bus-off state cannot be recovery, the bus-off interrupt should be enabled and reinit in it.

### 25.3.9. CAN interrupts

Four interrupt vectors are dedicated to CAN. Each interrupt source can be independently enabled or disabled by setting or resetting related bits in CAN\_INTEN.

The interrupt sources can be classified into:

- transmit interrupt
- FIFO0 interrupt
- FIFO1 interrupt
- error and status change interrupt

#### Transmit interrupt

The transmit interrupt can be generated by any of the following conditions and TMEIE bit in CAN\_INTEN register will be set:

- TX mailbox 0 transmit finished: MTF0 bit in the CAN\_TSTAT register is set.
- TX mailbox 1 transmit finished: MTF1 bit in the CAN\_TSTAT register is set.
- TX mailbox 2 transmit finished: MTF2 bit in the CAN\_TSTAT register is set.

#### Receive FIFO0 interrupt

The Receive FIFO0 interrupt can be generated by the following conditions:

- Reception FIFO0 not empty: RFL0 bits in the CAN\_RFIFO0 register are not '00' and RFNEIE0 in CAN\_INTEN register is set.
- Reception FIFO0 full: RFF0 bit in the CAN\_RFIFO0 register is set and RFFIE0 in CAN\_INTEN register is set.
- Reception FIFO0 overrun: RFO0 bit in the CAN\_RFIFO0 register is set and RFOIE0 in CAN\_INTEN register is set.

#### Receive FIFO1 interrupt

The Receive FIFO1 interrupt can be generated by the following conditions:

- Reception FIFO1 not empty: RFL1 bits in the CAN\_RFIFO1 register are not '00' and RFNEIE1 in CAN\_INTEN register is set.
- Reception FIFO1 full: RFF1 bit in the CAN\_RFIFO1 register is set and RFFIE1 in CAN\_INTEN register is set.
- Reception FIFO1 overrun: RFO1 bit in the CAN\_RFIFO1 register is set and RFOIE1 in CAN\_INTEN register is set.

#### Error and working mode change interrupt

The error and working mode change interrupt can be generated by the following conditions:

- Error: ERRIF bit in the CAN\_STAT register and ERRIE bit in the CAN\_INTEN register are set. Refer to ERRIF description in the CAN\_STAT register.
- Wakeup: WUIF bit in the CAN\_STAT register is set and WIE bit in the CAN\_INTEN register is set.

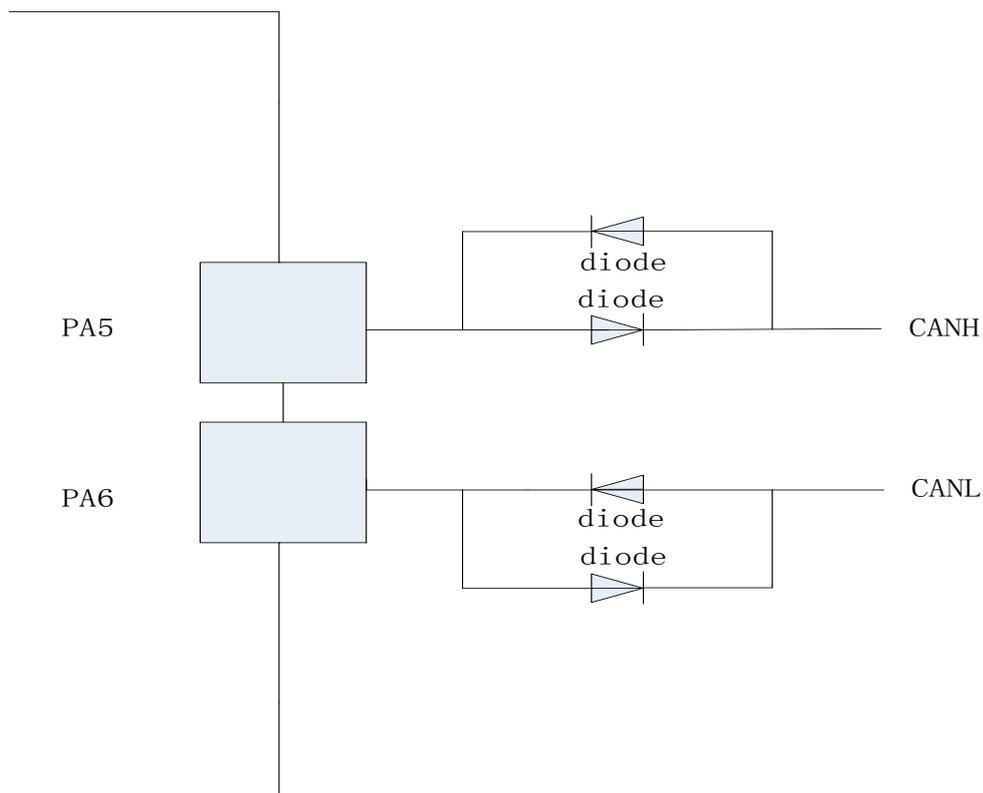
- Enter sleep working mode: SLPIF bit in the CAN\_STAT register is set and SLPWIE bit in the CAN\_INTEN register is set.

### 25.3.10. CAN PHY mode

If set PHYEN bit in CAN\_PHYCTL register, integrated CAN PHY is enabled. At this time the internal transceiver will begin to work. This mode only used for CAN0.

1. The connection outside of MCU chip is refer to the figure as follows:

**Figure 25-12. CAN PHY connection**



2. Configure the PA5/PA6 to analog input mode.
3. Enable the CAN clock.
4. Set PHYEN bit and PDMODE if needed in CAN\_PHYCTL register.
5. Configure CAN registers as usual, same as not in CAN PHY mode.

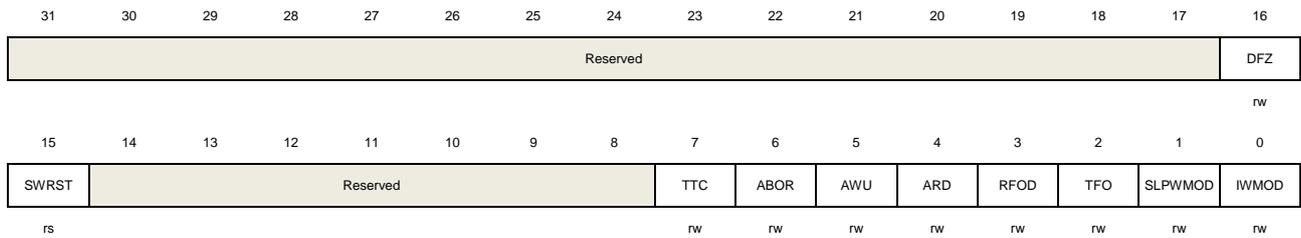
## 25.4. Register definition

### 25.4.1. Control register (CAN\_CTL)

Address offset: 0x00

Reset value: 0x0001 0002

This register has to be accessed by word(32-bit)



Bits	Fields	Descriptions
31:17	Reserved	Must be kept at reset value
16	DFZ	Debug freeze If the CANx_HOLD in DBG_CTL register is set, this bit define the CAN stop for debug or work normal. If the CANx_HOLD in DBG_CTL register is clear, this bit take not effect. 0: CAN reception and transmission working normal even during debug 1: CAN reception and transmission stop working during debug
15	SWRST	Software reset 0: No effect 1: Reset CAN with working mode of sleep. This bit is automatically reset to 0
14:8	Reserved	Must be kept at reset value
7	TTC	Time-triggered communication 0: Disable time-triggered communication 1: Enable time-triggered communication
6	ABOR	Automatic bus-off recovery 0: The bus-off state is left manually by software 1: The bus-off state is left automatically by hardware
5	AWU	Automatic wakeup If this bit is set, the sleep mode left when CAN bus activity detected, and SLPWMOD bit in CAN_CTL register will be cleared automatically. 0: The sleeping working mode is left manually by software 1: The sleeping working mode is left automatically by hardware
4	ARD	Automatic retransmission disable 0: Enable Automatic retransmission

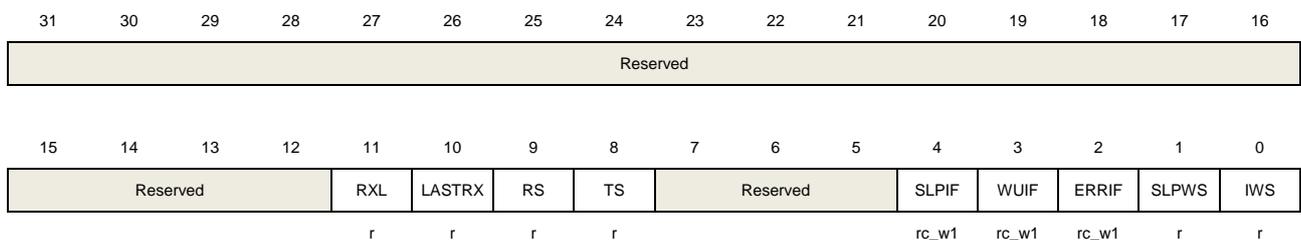
		1: Disable Automatic retransmission
3	RFOD	Receive FIFO overwrite disable 0: Enable receive FIFO overwrite when receive FIFO is full and overwrite the FIFO with the incoming frame 1: Disable receive FIFO overwrite when receive FIFO is full and discard the incoming frame
2	TFO	Transmit FIFO order 0: Order with the identifier of the frame 1: Order with first in and first out
1	SLPWMOD	Sleep working mode If this bit is set by software, the CAN enter sleep working mode after current transmission or reception complete. This bit can cleared by software or hardware. If AWU bit in CAN_CTL register is set, this bit is cleared by hardware when CAN bus activity detected. 0: Disable sleep working mode 1: Enable sleep working mode
0	IWMOD	Initial working mode 0: Disable initial working mode 1: Enable initial working mode

## 25.4.2. Status register (CAN\_STAT)

Address offset: 0x04

Reset value: 0x0000 0C02

This register has to be accessed by word(32-bit)



Bits	Fields	Descriptions
31:12	Reserved	Must be kept at reset value
11	RXL	RX level
10	LASTRX	Last sample value of RX pin
9	RS	Receiving state 0: CAN is not working in the receiving state

		1: CAN is working in the receiving state
8	TS	<p>Transmitting state</p> <p>0: CAN is not working in the transmitting state</p> <p>1: CAN is working in the transmitting state</p>
7:5	Reserved	Must be kept at reset value
4	SLPIF	<p>Status change interrupt flag of sleep working mode entering</p> <p>This bit is set by hardware when entering sleep working mode, and cleared by hardware when the CAN not in sleep working mode. This bit can also cleared by software when write 1 to this bit.</p> <p>0: CAN is not entering the sleep working mode</p> <p>1: CAN is entering the sleep working mode.</p>
3	WUIF	<p>Status change interrupt flag of wakeup from sleep working mode</p> <p>This bit is set when CAN bus activity detected on sleep working mode. This bit can cleared by software when write 1 to this bit.</p> <p>0: Wakeup event is not coming</p> <p>1: Wakeup event is coming</p>
2	ERRIF	<p>Error interrupt flag</p> <p>This bit is set by following event. The BOERR bit in CAN_ERR register is set and BOIE bit in CAN_INTEN register is set. Or the PERR bit in CAN_ERR register is set and PERRIE bit in CAN_INTEN register is set. Or the WERR bit in CAN_ERR register is set and WERRIE bit in CAN_INTEN register is set. Or the ERRN bits in CAN_ERR register are set to 1 to 6 (not 0 and not 7) and ERRNIE in CAN_INTEN register is set. This bit is cleared by software when write 1 to this bit.</p> <p>0: No error interrupt flag</p> <p>1: Any error interrupt flag has happened</p>
1	SLPWS	<p>Sleep working state</p> <p>This bit is set by hardware when the CAN enter sleep working mode after set SLPWMOD bit in CAN_CTL register. If the CAN leave from normal working mode to sleep working mode, it must wait the current frame transmission or reception completed. This bit is cleared by hardware when the CAN leave sleep working mode. Clear SLPWMOD bit in CAN_CTL register or automatically detect the CAN bus activity when AWU bit is set in CAN_CTL register. If leave sleep working mode to normal working mode, this bit will be cleared after receive 11 consecutive recessive bits from the CAN bus.</p> <p>0: CAN is not the state of sleep working mode</p> <p>1: CAN is the state of sleep working mode</p>
0	IWS	<p>Initial working state</p> <p>This bit is set by hardware when the CAN enter initial working mode after set IWMOD bit in CAN_CTL register. If the CAN leave from normal working mode to initial working mode, it must wait the current frame transmission or reception completed. This bit is</p>

cleared by hardware when the CAN leave initial working mode after clear IWMOD bit in CAN\_CTL register. If leave initial working mode to normal working mode, this bit will be cleared after receive 11 consecutive recessive bits from the CAN bus.

0: CAN is not the state of initial working mode

1: CAN is the state of initial working mode

### 25.4.3. Transmit status register (CAN\_TSTAT)

Address offset: 0x08

Reset value: 0x1C00 0000

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TMLS2	TMLS1	TMLS0	TME2	TME1	TME0	NUM[1:0]		MST2	Reserved			MTE2	MAL2	MTFNERR2	MTF2
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
MST1	Reserved			MTE1	MAL1	MTFNERR1	MTF1	MST0	Reserved			MTE0	MAL0	MTFNERR0	MTF0
rs				rc_w1	rc_w1	rc_w1	rc_w1	rs				rc_w1	rc_w1	rc_w1	rc_w1

Bits	Fields	Descriptions
31	TMLS2	Transmit mailbox 2 last sending in transmit FIFO This bit is set by hardware when transmit mailbox 2 has the last sending order in the transmit FIFO with at least two frame are pending.
30	TMLS1	Transmit mailbox 1 last sending in transmit FIFO This bit is set by hardware when transmit mailbox 1 has the last sending order in the transmit FIFO with at least two frame are pending.
29	TMLS0	Transmit mailbox 0 last sending in transmit FIFO This bit is set by hardware when transmit mailbox 0 has the last sending order in the transmit FIFO with at least two frame are pending.
28	TME2	Transmit mailbox 2 empty 0: Transmit mailbox 2 not empty 1: Transmit mailbox 2 empty
27	TME1	Transmit mailbox 1 empty 0: Transmit mailbox 1 not empty 1: Transmit mailbox 1 empty
26	TME0	Transmit mailbox 0 empty 0: Transmit mailbox 0 not empty 1: Transmit mailbox 0 empty

25:24	NUM[1:0]	<p>These bits are the number of the transmit FIFO mailbox in which the frame will be transmitted if at least one mailbox is empty.</p> <p>These bits are the number of the transmit FIFO mailbox in which the frame will be transmitted lastly if all mailboxes are full.</p>
23	MST2	<p>Mailbox 2 stop transmitting</p> <p>This bit is set by the software to stop mailbox 2 transmitting.</p> <p>This bit is reset by the hardware while the mailbox 2 is empty.</p>
22:20	Reserved	Must be kept at reset value
19	MTE2	<p>Mailbox 2 transmit error</p> <p>This bit is set by hardware while the transmit error is occurred. This bit reset by software when write 1 to this bit or MTF2 bit in CAN_TSTAT register. This bit reset by hardware when next transmit start.</p>
18	MAL2	<p>Mailbox 2 arbitration lost</p> <p>This bit is set while the arbitration lost is occurred. This bit reset by software when write 1 to this bit or MTF2 bit in CAN_TSTAT register. This bit reset by hardware when next transmit start.</p>
17	MTFNERR2	<p>Mailbox 2 transmit finished and no error</p> <p>This bit is set when the transmission finished and no error. This bit reset by software when write 1 to this bit or MTF2 bit in CAN_TSTAT register. This bit reset by hardware when the transmission finished with error.</p> <p>0: Mailbox 2 transmit finished with error 1: Mailbox 2 transmit finished and no error</p>
16	MTF2	<p>Mailbox 2 transmit finished</p> <p>This bit set by hardware when the transmission finish or abort. This bit reset by software when write 1 to this bit or TEN bit in CAN_TMI2 is 1.</p> <p>0: Mailbox 2 transmit is progressing 1: Mailbox 2 transmit finished</p>
15	MST1	<p>Mailbox 1 stop transmitting</p> <p>This bit is set by the software to stop mailbox 1 transmitting.</p> <p>This bit is reset by the hardware while the mailbox 1 is empty.</p>
14:12	Reserved	Must be kept at reset value
11	MTE1	<p>Mailbox 1 transmit error</p> <p>This bit is set by hardware while the transmit error is occurred. This bit reset by software when write 1 to this bit or MTF1 bit in CAN_TSTAT register. This bit reset by hardware when next transmit start.</p>
10	MAL1	<p>Mailbox 1 arbitration lost</p> <p>This bit is set while the arbitration lost is occurred. This bit reset by software when write 1 to this bit or MTF1 bit in CAN_TSTAT register. This bit reset by hardware when</p>

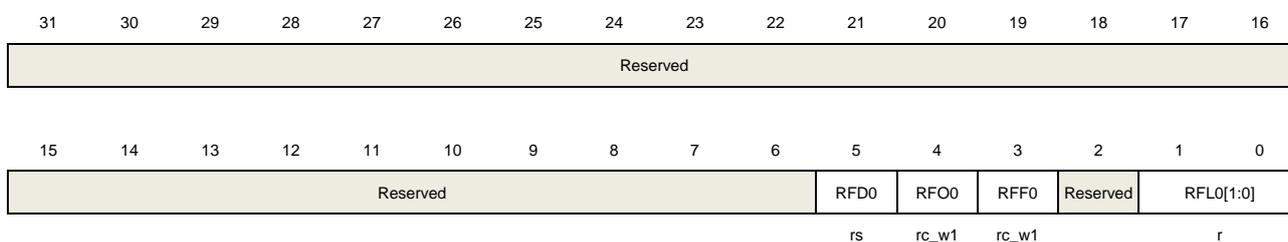
		next transmit start.
9	MTFNERR1	Mailbox 1 transmit finished and no error This bit is set when the transmission finished and no error. This bit reset by software when write 1 to this bit or MTF1 bit in CAN_TSTAT register. This bit reset by hardware when the transmission finished with error. 0: Mailbox 1 transmit finished with error 1: Mailbox 1 transmit finished and no error
8	MTF1	Mailbox 1 transmit finished This bit is set by hardware when the transmission finish or abort. This bit reset by software when write 1 to this bit or TEN bit in CAN_TMI1 is 1. 0: Mailbox 1 transmit is progressing 1: Mailbox 1 transmit finished
7	MST0	Mailbox 0 stop transmitting This bit is set by the software to stop mailbox 0 transmitting. This bit is reset by the hardware while the mailbox 0 is empty.
6:4	Reserved	Must be kept at reset value
3	MTE0	Mailbox 0 transmit error This bit is set by hardware while the transmit error is occurred. This bit reset by software when write 1 to this bit or MTF0 bit in CAN_TSTAT register. This bit reset by hardware when next transmit start.
2	MAL0	Mailbox 0 arbitration lost This bit is set while the arbitration lost is occurred. This bit reset by software when write 1 to this bit or MTF0 bit in CAN_TSTAT register. This bit reset by hardware when next transmit start.
1	MTFNERR0	Mailbox 0 transmit finished and no error This bit is set when the transmission finished and no error. This bit reset by software when write 1 to this bit or MTF0 bit in CAN_TSTAT register. This bit reset by hardware when the transmission finished with error. 0: Mailbox 0 transmit finished with error 1: Mailbox 0 transmit finished and no error
0	MTF0	Mailbox 0 transmit finished This bit is set by hardware when the transmission finish or abort. This bit reset by software when write 1 to this bit or TEN bit in CAN_TMI0 is 1. 0: Mailbox 0 transmit is progressing 1: Mailbox 0 transmit finished

## 25.4.4. Receive message FIFO0 register (CAN\_RFIFO0)

Address offset: 0x0C

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)



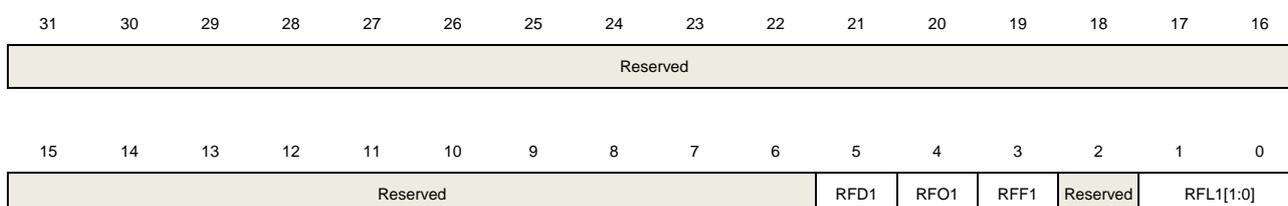
Bits	Fields	Descriptions
31:6	Reserved	Must be kept at reset value
5	RFD0	Receive FIFO0 dequeue This bit is set by the software to start dequeuing a frame from receive FIFO0. This bit is reset by the hardware while the dequeuing is done.
4	RFO0	Receive FIFO0 overfull This bit is set by hardware when receive FIFO0 is overfull and reset by software when write 1 to this bit. 0: The receive FIFO0 is not overfull 1: The receive FIFO0 is overfull
3	RFF0	Receive FIFO0 full This bit is set by hardware when receive FIFO0 is full and reset by software when write 1 to this bit. 0: The receive FIFO0 is not full 1: The receive FIFO0 is full
2	Reserved	Must be kept at reset value
1:0	RFL0[1:0]	Receive FIFO0 length These bits are the length of the receive FIFO0.

## 25.4.5. Receive message FIFO1 register (CAN\_RFIFO1)

Address offset: 0x10

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)



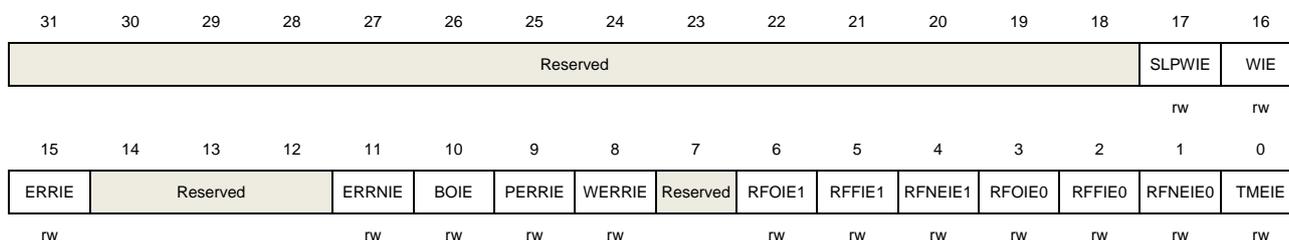
Bits	Fields	Descriptions
31:6	Reserved	Must be kept at reset value
5	RFD1	Receive FIFO1 dequeue This bit is set by the software to start dequeuing a frame from receive FIFO1. This bit is reset by the hardware while the dequeuing is done.
4	RFO1	Receive FIFO1 overfull This bit is set by hardware when receive FIFO1 is overfull and reset by software when write 1 to this bit. 0: The receive FIFO1 is not overfull 1: The receive FIFO1 is overfull
3	RFF1	Receive FIFO1 full This bit is set by hardware when receive FIFO1 is full and reset by software when write 1 to this bit. 0: The receive FIFO1 is not full 1: The receive FIFO1 is full
2	Reserved	Must be kept at reset value
1:0	RFL1[1:0]	Receive FIFO1 length These bits are the length of the receive FIFO1.

### 25.4.6. Interrupt enable register (CAN\_INTEN)

Address offset: 0x14

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)



Bits	Fields	Descriptions
31:18	Reserved	Must be kept at reset value
17	SLPWIE	Sleep working interrupt enable 0: Sleep working interrupt disable 1: Sleep working interrupt enable

16	WIE	Wakeup interrupt enable 0: Wakeup interrupt disable 1: Wakeup interrupt enable
15	ERRIE	Error interrupt enable 0: Error interrupt disable 1: Error interrupt enable
14:12	Reserved	Must be kept at reset value
11	ERRNIE	Error number interrupt enable 0: Error number interrupt disable 1: Error number interrupt enable
10	BOIE	Bus-off interrupt enable 0: Bus-off interrupt disable 1: Bus-off interrupt enable
9	PERRIE	Passive error interrupt enable 0: Passive error interrupt disable 1: Passive error interrupt enable
8	WERRIE	Warning error interrupt enable 0: Warning error interrupt disable 1: Warning error interrupt enable
7	Reserved	Must be kept at reset value
6	RFOIE1	Receive FIFO1 overfull interrupt enable 0: Receive FIFO1 overfull interrupt disable 1: Receive FIFO1 overfull interrupt enable
5	RFFIE1	Receive FIFO1 full interrupt enable 0: Receive FIFO1 full interrupt disable 1: Receive FIFO1 full interrupt enable
4	RFNEIE1	Receive FIFO1 not empty interrupt enable 0: Receive FIFO1 not empty interrupt disable 1: Receive FIFO1 not empty interrupt enable
3	RFOIE0	Receive FIFO0 overfull interrupt enable 0: Receive FIFO0 overfull interrupt disable 1: Receive FIFO0 overfull interrupt enable
2	RFFIE0	Receive FIFO0 full interrupt enable 0: Receive FIFO0 full interrupt disable 1: Receive FIFO0 full interrupt enable
1	RFNEIE0	Receive FIFO0 not empty interrupt enable 0: Receive FIFO0 not empty interrupt disable

1: Receive FIFO0 not empty interrupt enable

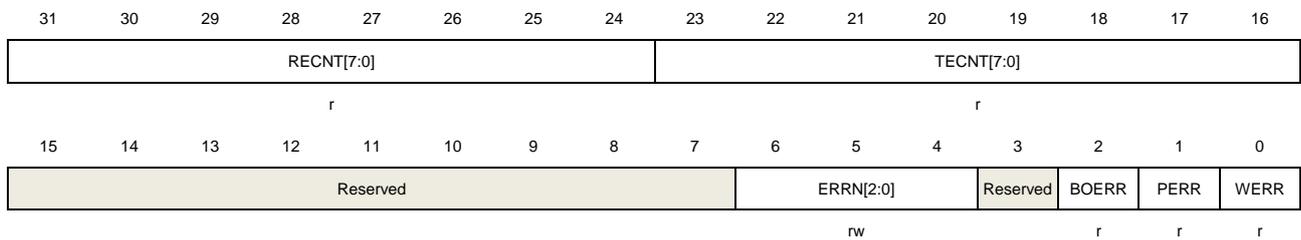
0            TMEIE            Transmit mailbox empty interrupt enable  
 0: Transmit mailbox empty interrupt disable  
 1: Transmit mailbox empty interrupt enable

## 25.4.7. Error register (CAN\_ERR)

Address offset: 0x18

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)



Bits	Fields	Descriptions
31:24	RECNT[7:0]	Receive Error Count defined by the CAN standard
23:16	TECNT[7:0]	Transmit Error Count defined by the CAN standard
15:7	Reserved	Must be kept at reset value
6:4	ERRN[2:0]	Error number These bits indicate the error status of bit transformation. They are updated by the hardware. While the bit transformation is successful, they are equal to 0. Software can set these bits to 0b111. 000: No Error 001: Stuff Error 010: Form Error 011: Acknowledgment Error 100: Bit recessive Error 101: Bit dominant Error 110: CRC Error 111: Set by software
3	Reserved	Must be kept at reset value
2	BOERR	Bus-off error Whenever the CAN enters bus-off state, the bit will be set by the hardware. The bus-off state is entered on TECNT overflow, greater than 255.
1	PERR	Passive error

Whenever the TECNT or RECNT is greater than 127, the bit will be set by the hardware.

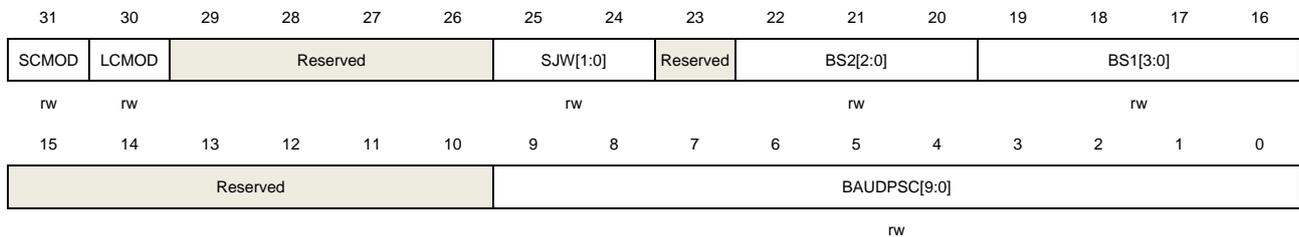
0	WERR	Warning error Whenever the TECNT or RECNT is greater than or equal to 96, the bit will be set by the hardware.
---	------	---

## 25.4.8. Bit timing register (CAN\_BT)

Address offset: 0x1C

Reset value: 0x0123 0000

This register has to be accessed by word(32-bit)



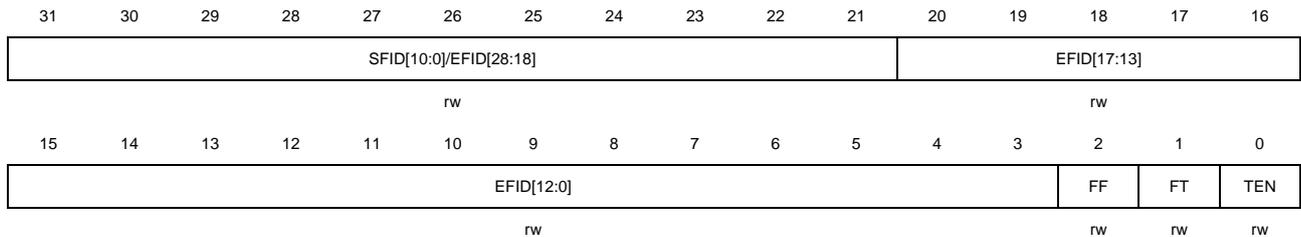
Bits	Fields	Descriptions
31	SCMOD	Silent communication mode 0: Silent communication disable 1: Silent communication enable
30	LCMOD	Loopback communication mode 0: Loopback communication disable 1: Loopback communication enable
29:26	Reserved	Must be kept at reset value
25:24	SJW[1:0]	Resynchronization jump width Resynchronization jump width time quantum= SJW[1:0]+1
23	Reserved	Must be kept at reset value
22:20	BS2[2:0]	Bit segment 2 Bit segment 2 time quantum=BS2[2:0]+1
19:16	BS1[3:0]	Bit segment 1 Bit segment 1 time quantum=BS1[3:0]+1
15:10	Reserved	Must be kept at reset value
9:0	BAUDPSC[9:0]	Baud rate prescaler The CAN baud rate prescaler

### 25.4.9. Transmit mailbox identifier register (CAN\_TMIx) (x=0..2)

Address offset: 0x180, 0x190, 0x1A0

Reset value: 0xFFFF XXXX (bit0=0)

This register has to be accessed by word(32-bit)



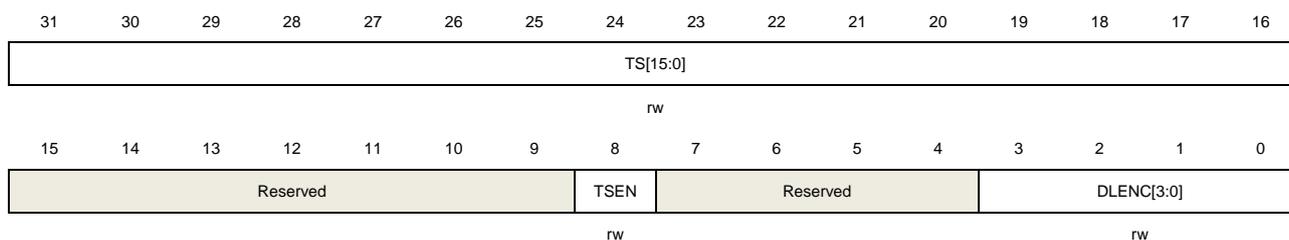
Bits	Fields	Descriptions
31:21	SFID[10:0]/EFID[28:18]	The frame identifier SFID[10:0]: Standard format frame identifier EFID[28:18]: Extended format frame identifier
20:16	EFID[17:13]	The frame identifier EFID[17:13]: Extended format frame identifier
15:3	EFID[12:0]	The frame identifier EFID[12:0]: Extended format frame identifier
2	FF	Frame format 0: Standard format frame 1: Extended format frame
1	FT	Frame type 0: Data frame 1: Remote frame
0	TEN	Transmit enable This bit is set by the software when one frame will be transmitted and reset by the hardware when the transmit mailbox is empty. 0: Transmit disable 1: Transmit enable

### 25.4.10. Transmit mailbox property register (CAN\_TMPx) (x=0..2)

Address offset: 0x184, 0x194, 0x1A4

Reset value: 0xFFFF XXXX

This register has to be accessed by word(32-bit)



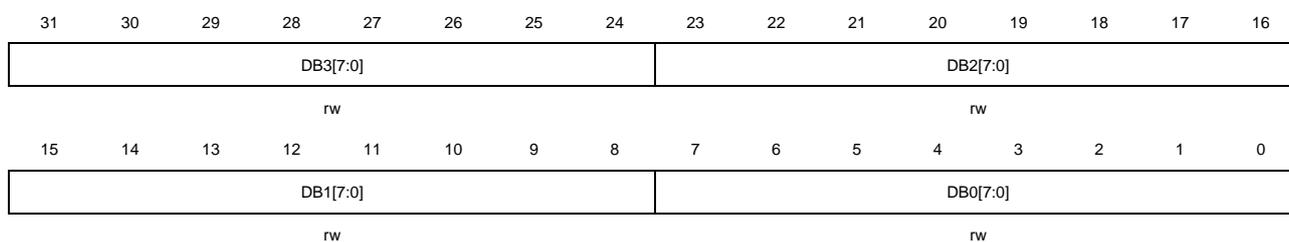
Bits	Fields	Descriptions
31:16	TS[15:0]	Time stamp The time stamp of frame in transmit mailbox.
15:9	Reserved	Must be kept at reset value
8	TSEN	Time stamp enable 0: Time stamp disable 1: Time stamp enable. The TS[15:0] will be transmitted in the DB6 and DB7 in DL This bit is available while the TTC bit in CAN_CTL is set.
7:4	Reserved	Must be kept at reset value
3:0	DLENC[3:0]	Data length code DLENC[3:0] is the number of bytes in a frame.

#### 25.4.11. Transmit mailbox data0 register (CAN\_TMDATA0x) (x=0..2)

Address offset: 0x188, 0x198, 0x1A8

Reset value: 0XXXXX XXXX

This register has to be accessed by word(32-bit)



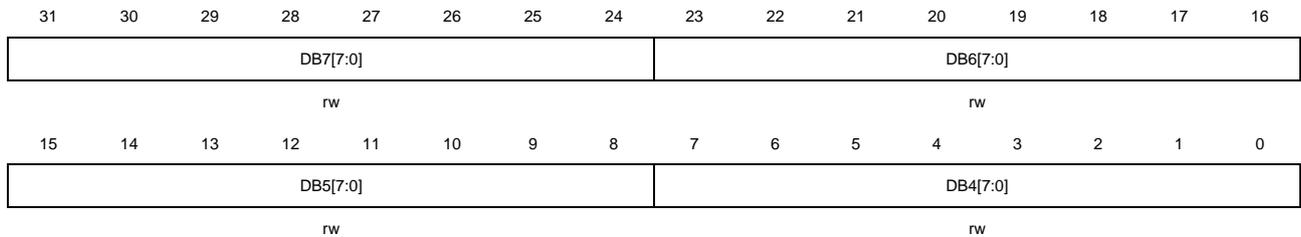
Bits	Fields	Descriptions
31:24	DB3[7:0]	Data byte 3
23:16	DB2[7:0]	Data byte 2
15:8	DB1[7:0]	Data byte 1
7:0	DB0[7:0]	Data byte 0

### 25.4.12. Transmit mailbox data1 register (CAN\_TMDATA1x) (x=0..2)

Address offset: 0x18C, 0x19C, 0x1AC

Reset value: 0XXXXX XXXX

This register has to be accessed by word(32-bit)



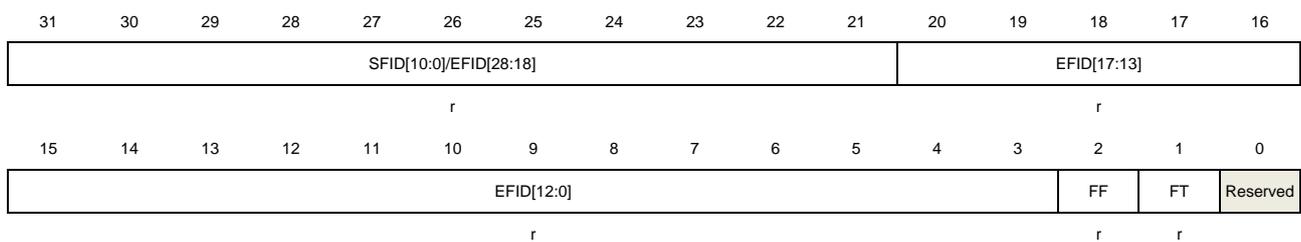
Bits	Fields	Descriptions
31:24	DB7[7:0]	Data byte 7
23:16	DB6[7:0]	Data byte 6
15:8	DB5[7:0]	Data byte 5
7:0	DB4[7:0]	Data byte 4

### 25.4.13. Receive FIFO mailbox identifier register (CAN\_RFIFOMIx) (x=0,1)

Address offset: 0x1B0, 0x1C0

Reset value: 0XXXXX XXXX

This register has to be accessed by word(32-bit)



Bits	Fields	Descriptions
31:21	SFID[10:0]/EFID[28:18]	The frame identifier SFID[10:0]: Standard format frame identifier EFID[28:18]: Extended format frame identifier
20:16	EFID[17:13]	The frame identifier EFID[17:13]: Extended format frame identifier
15:3	EFID[12:0]	The frame identifier

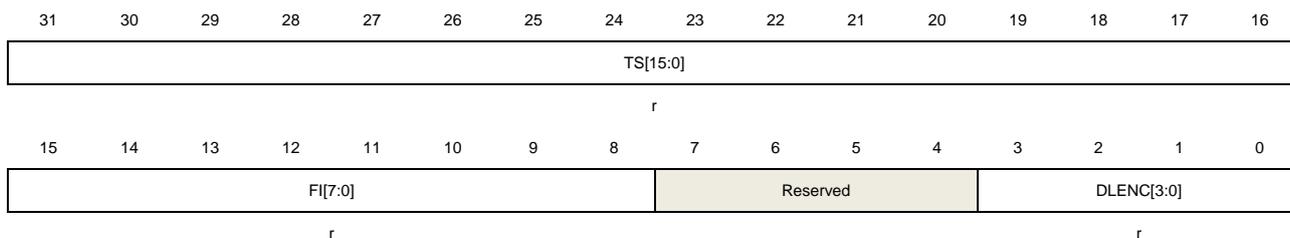
		EFID[12:0]: Extended format frame identifier
2	FF	Frame format 0: Standard format frame 1: Extended format frame
1	FT	Frame type 0: Data frame 1: Remote frame
0	Reserved	Must be kept at reset value

## 25.4.14. Receive FIFO mailbox property register (CAN\_RFIFOMPx) (x=0,1)

Address offset: 0x1B4, 0x1C4

Reset value: 0xFFFF XXXX

This register has to be accessed by word(32-bit)



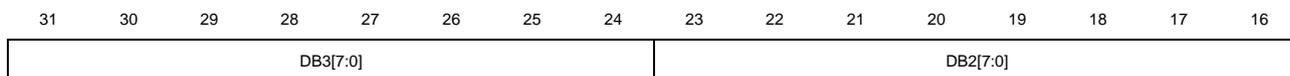
Bits	Fields	Descriptions
31:16	TS[15:0]	Time stamp The time stamp of frame in transmit mailbox.
15:8	FI[7:0]	Filtering index The index of the filter by which the frame is passed.
7:4	Reserved	Must be kept at reset value
3:0	DLENC[3:0]	Data length code DLENC[3:0] is the number of bytes in a frame.

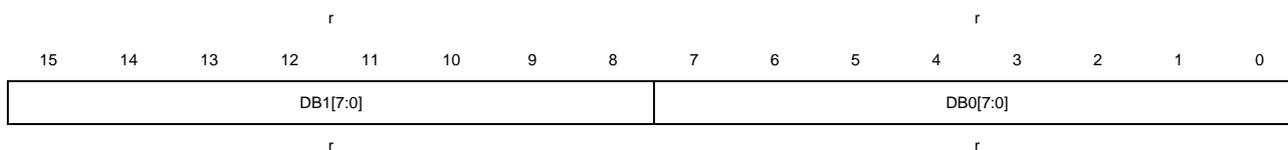
## 25.4.15. Receive FIFO mailbox data0 register (CAN\_RFIFOMDATA0x) (x=0,1)

Address offset: 0x1B8, 0x1C8

Reset value: 0xFFFF XXXX

This register has to be accessed by word(32-bit)





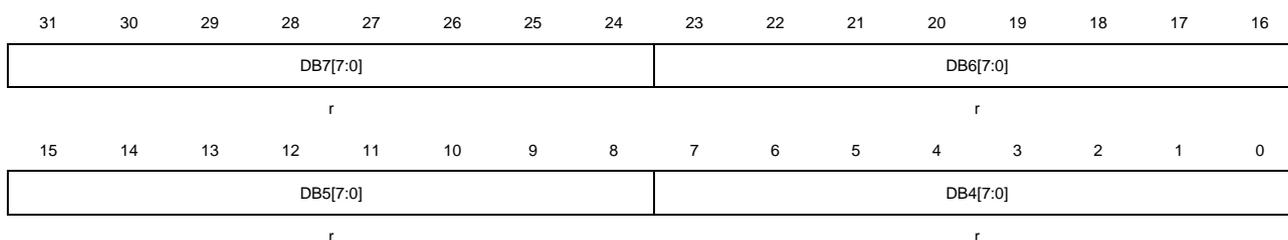
Bits	Fields	Descriptions
31:24	DB3[7:0]	Data byte 3
23:16	DB2[7:0]	Data byte 2
15:8	DB1[7:0]	Data byte 1
7:0	DB0[7:0]	Data byte 0

#### 25.4.16. Receive FIFO mailbox data1 register (CAN\_RFIFOMDATA1x) (x=0,1)

Address offset: 0x1BC, 0x1CC

Reset value: 0XXXXX XXXX

This register has to be accessed by word(32-bit)



Bits	Fields	Descriptions
31:24	DB7[7:0]	Data byte 7
23:16	DB6[7:0]	Data byte 6
15:8	DB5[7:0]	Data byte 5
7:0	DB4[7:0]	Data byte 4

#### 25.4.17. Filter control register (CAN\_FCTL)

Address offset: 0x200

Reset value: 0x2A1C 0E01

This register has to be accessed by word(32-bit)



15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		HBC1F[5:0]						Reserved							FLD
rw															rw

Bits	Fields	Descriptions
31:14	Reserved	Must be kept at reset value
13:8	HBC1F[5:0]	Header bank of CAN1 filter These bits are set and cleared by software to define the first bank for CAN1 filter. Bank0 ~ Bank HBC1F-1 used to CAN0. Bank HBC1F ~ Bank27 used to CAN1. When set 0, not bank used to CAN0. When set 28, not bank used to CAN1.
7:1	Reserved	Must be kept at reset value
0	FLD	Filter lock disable 0: Filter lock enable 1: Filter lock disable

#### 25.4.18. Filter mode configuration register (CAN\_FMCFG)

Address offset: 0x204

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit). This register can be modified only when FLD bit in CAN\_FCTL register is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				FMOD27	FMOD26	FMOD25	FMOD24	FMOD23	FMOD22	FMOD21	FMOD20	FMOD19	FMOD18	FMOD17	FMOD16
				rw											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FMOD15	FMOD14	FMOD13	FMOD12	FMOD11	FMOD10	FMOD9	FMOD8	FMOD7	FMOD6	FMOD5	FMOD4	FMOD3	FMOD2	FMOD1	FMOD0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:28	Reserved	Must be kept at reset value
27:0	FMODx	Filter mode 0: Filter x with Mask mode 1: Filter x with List mode

#### 25.4.19. Filter scale configuration register (CAN\_FSCFG)

Address offset: 0x20C

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit). This register can be modified only when FLD bit in CAN\_FCTL register is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				FS27	FS26	FS25	FS24	FS23	FS22	FS21	FS20	FS19	FS18	FS17	FS16
				rw											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FS15	FS14	FS13	FS12	FS11	FS10	FS9	FS8	FS7	FS6	FS5	FS4	FS3	FS2	FS1	FS0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:28	Reserved	Must be kept at reset value
27:0	FSx	Filter scale 0: Filter x with 16-bit scale 1: Filter x with 32-bit scale

### 25.4.20. Filter associated FIFO register (CAN\_FAFIFO)

Address offset: 0x214

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit). This register can be modified only when FLD bit in CAN\_FCTL register is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				FAF27	FAF26	FAF25	FAF24	FAF23	FAF22	FAF21	FAF20	FAF19	FAF18	FAF17	FAF16
				rw											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FAF15	FAF14	FAF13	FAF12	FAF11	FAF10	FAF9	FAF8	FAF7	FAF6	FAF5	FAF4	FAF3	FAF2	FAF1	FAF0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:28	Reserved	Must be kept at reset value
27:0	FAFx	Filter associated FIFO 0: Filter x associated with FIFO0 1: Filter x associated with FIFO1

### 25.4.21. Filter working register (CAN\_FW)

Address offset: 0x21C

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				FW27	FW26	FW25	FW24	FW23	FW22	FW21	FW20	FW19	FW18	FW17	FW16
				rw											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FW15	FW14	FW13	FW12	FW11	FW10	FW9	FW8	FW7	FW6	FW5	FW4	FW3	FW2	FW1	FW0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:28	Reserved	Must be kept at reset value
27:0	FWx	Filter working 0: Filter x working disable 1: Filter x working enable

#### 25.4.22. Filter x data y register (CAN\_FxDATAy) (x=0..27, y=0,1)

Address offset: 0x240+8\*x+4\*y, (x=0..27, y=0,1)

Reset value: 0XXXXX XXXX

This register has to be accessed by word(32-bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FD31	FD30	FD29	FD28	FD27	FD26	FD25	FD24	FD23	FD22	FD21	FD20	FD19	FD18	FD17	FD16
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FD15	FD14	FD13	FD12	FD11	FD10	FD9	FD8	FD7	FD6	FD5	FD4	FD3	FD2	FD1	FD0
rw															

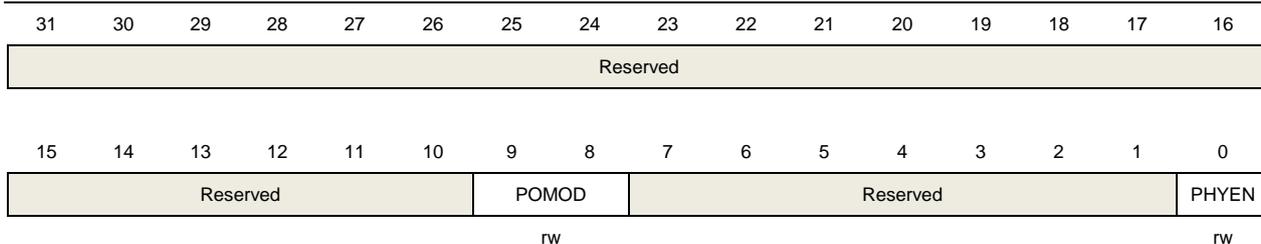
Bits	Fields	Descriptions
31:0	FDx	Filter data Mask mode 0: Mask match disable 1: Mask match enable  List mode 0: List identifier bit is 0 1: List identifier bit is 1

#### 25.4.23. PHY control register (CAN\_PHYCTL)

Address offset: 0x3FC

Reset value: 0x0000 0300

This register has to be accessed by word(32-bit).



Bits	Fields	Descriptions
31:10	Reserved	Must be kept at reset value
9:8	POMOD	CAN PHY output driver control register This bit set and cleared by software. 00: Low Slope Mode 01: Middle Slope Mode 10: High Slope Mode 11: High Speed Mode(Default)
7:1	Reserved	Must be kept at reset value
0	PHYEN	PHY enable bit This bit set and cleared by software. This bit can use CAN PHY for CAN0 or not. 0: No CAN PHY mode 1: CAN PHY enable for CAN0

## 26. Revision history

Table 26-1. Revision history

Revision No.	Description	Date
1.0	Initial Release	Mar.18, 2014
2.0	Add GD32F170/190 Products	Jan.15, 2016
3.0	Adapt To New Name Convention	Jun.24, 2016
3.0.1	Proofreading	Mar.30, 2017
3.1.0	Proofreading	Jan.22, 2018