

**GigaDevice Semiconductor Inc.**

**GD32VF103  
RISC-V 32-bit MCU**

**User Manual**

Revision 1.0

( Jun. 2019 )

# Table of Contents

<b>Table of Contents .....</b>	<b>2</b>
<b>List of Figures.....</b>	<b>14</b>
<b>List of Tables.....</b>	<b>20</b>
<b>1. System and memory architecture .....</b>	<b>22</b>
1.1. RISC-V CPU.....	22
1.2. System architecture.....	22
1.3. Memory map .....	24
1.3.1. On-chip SRAM memory.....	28
1.3.2. On-chip flash memory overview .....	28
1.4. Boot configuration .....	29
1.5. Device electronic signature.....	29
1.5.1. Memory density information .....	30
1.5.2. Unique device ID (96 bits) .....	30
<b>2. Flash memory controller (FMC) .....</b>	<b>32</b>
2.1. Overview .....	32
2.2. Characteristics.....	32
2.3. Function overview.....	32
2.3.1. Flash memory architecture.....	32
2.3.2. Read operations.....	33
2.3.3. Unlock the FMC_CTL0 registers .....	33
2.3.4. Page erase .....	33
2.3.5. Mass erase .....	34
2.3.6. Main flash programming .....	35
2.3.7. Option bytes Erase .....	37
2.3.8. Option bytes modify.....	38
2.3.9. Option bytes description .....	38
2.3.10. Page erase/program protection .....	39
2.3.11. Security protection .....	40
2.4. Register definition.....	41
2.4.1. Wait state register (FMC_WS) .....	41
2.4.2. Unlock key register 0(FMC_KEY0) .....	41
2.4.3. Option byte unlock key register (FMC_OBKEY) .....	42
2.4.4. Status register 0 (FMC_STAT0).....	42
2.4.5. Control register 0 (FMC_CTL0) .....	43
2.4.6. Address register 0 (FMC_ADDR0).....	44

2.4.7. Option byte status register (FMC_OBSTAT).....	45
2.4.8. Erase/Program Protection register (FMC_WP).....	45
2.4.9. Product ID register (FMC_PID) .....	46
<b>3. Power management unit (PMU) .....</b>	<b>47</b>
<b>3.1. Overview .....</b>	<b>47</b>
<b>3.2. Characteristics.....</b>	<b>47</b>
<b>3.3. Function overview.....</b>	<b>47</b>
3.3.1. Battery backup domain .....	48
3.3.2. V <sub>DD</sub> /V <sub>DDA</sub> power domain .....	49
3.3.3. 1.2V power domain.....	51
3.3.4. Power saving modes .....	51
<b>3.4. Register definition.....</b>	<b>54</b>
3.4.1. Control register (PMU_CTL) .....	54
3.4.2. Control and status register (PMU_CS).....	55
<b>4. Backup registers (BKP) .....</b>	<b>57</b>
<b>4.1. Overview .....</b>	<b>57</b>
<b>4.2. Characteristics.....</b>	<b>57</b>
<b>4.3. Function overview.....</b>	<b>57</b>
4.3.1. RTC clock calibration .....	57
4.3.2. Tamper detection .....	58
<b>4.4. Register definition.....</b>	<b>59</b>
4.4.1. Backup data register x (BKP_DATAx) (x= 0..41) .....	59
4.4.2. RTC signal output control register (BKP_OCTL) .....	59
4.4.3. Tamper pin control register (BKP_TPCTL) .....	60
4.4.4. Tamper control and status register (BKP_TPCS) .....	60
<b>5. Reset and clock unit (RCU) .....</b>	<b>62</b>
<b>5.1. Reset control unit (RCTL) .....</b>	<b>62</b>
5.1.1. Overview .....	62
5.1.2. Function overview.....	62
<b>5.2. Clock control unit (CCTL).....</b>	<b>63</b>
5.2.1. Overview .....	63
5.2.2. Characteristics.....	65
5.2.3. Function overview.....	65
<b>5.3. Register definition.....</b>	<b>69</b>
5.3.1. Control register (RCU_CTL).....	69
5.3.2. Clock configuration register 0 (RCU_CFG0) .....	71
5.3.3. Clock interrupt register (RCU_INT) .....	74
5.3.4. APB2 reset register (RCU_APB2RST).....	77

5.3.5.	APB1 reset register (RCU_APB1RST).....	79
5.3.6.	AHB enable register (RCU_AHBEN) .....	82
5.3.7.	APB2 enable register (RCU_APB2EN) .....	83
5.3.8.	APB1 enable register (RCU_APB1EN) .....	85
5.3.9.	Backup domain control register (RCU_BDCTL) .....	87
5.3.10.	Reset source/clock register (RCU_RSTSCK) .....	88
5.3.11.	AHB reset register (RCU_AHBRST).....	90
5.3.12.	Clock configuration register 1 (RCU_CFG1) .....	91
5.3.13.	Deep-sleep mode voltage register (RCU_DSV) .....	93
<b>6.</b>	<b>Interrupt/event controller (EXTI) .....</b>	<b>94</b>
6.1.	<b>Overview .....</b>	<b>94</b>
6.2.	<b>Characteristics.....</b>	<b>94</b>
6.3.	<b>Function overview.....</b>	<b>94</b>
6.4.	<b>External interrupt and event (EXTI) block diagram.....</b>	<b>97</b>
6.5.	<b>External Interrupt and Event function overview .....</b>	<b>97</b>
6.6.	<b>Register definition.....</b>	<b>99</b>
6.6.1.	Interrupt enable register (EXTI_INTEN) .....	99
6.6.2.	Event enable register (EXTI_EVENT).....	99
6.6.3.	Rising edge trigger enable register (EXTI_RTEN).....	100
6.6.4.	Falling edge trigger enable register (EXTI_FTEN) .....	100
6.6.5.	Software interrupt event register (EXTI_SWIEV).....	101
6.6.6.	Pending register (EXTI_PD).....	101
<b>7.</b>	<b>General-purpose and alternate-function I/Os (GPIO and AFIO) .....</b>	<b>102</b>
7.1.	<b>Overview .....</b>	<b>102</b>
7.2.	<b>Characteristics.....</b>	<b>102</b>
7.3.	<b>Function overview.....</b>	<b>102</b>
7.3.1.	GPIO pin configuration.....	103
7.3.2.	External interrupt/event lines .....	104
7.3.3.	Alternate functions (AF) .....	104
7.3.4.	Input configuration .....	104
7.3.5.	Output configuration .....	105
7.3.6.	Analog configuration.....	106
7.3.7.	Alternate function (AF) configuration .....	106
7.3.8.	IO pin function selection .....	107
7.3.9.	GPIO locking function .....	107
7.4.	<b>Remapping function I/O and debug configuration .....</b>	<b>108</b>
7.4.1.	Introduction .....	108
7.4.2.	Main features.....	108
7.4.3.	JTAG alternate function remapping .....	108

7.4.4.	TIMER AF remapping.....	109
7.4.5.	USART AF remapping.....	110
7.4.6.	I2C0 AF remapping.....	110
7.4.7.	SPI0 AF remapping .....	111
7.4.8.	SPI2/I2S2 AF remapping .....	111
7.4.9.	CAN0 AF remapping .....	111
7.4.10.	CAN1 AF remapping .....	111
7.4.11.	CLK pins AF remapping.....	112
<b>7.5.</b>	<b>Register definition.....</b>	<b>113</b>
7.5.1.	Port control register 0 (GPIOx_CTL0, x=A..E) .....	113
7.5.2.	Port control register 1 (GPIOx_CTL1, x=A..E) .....	115
7.5.3.	Port input status register (GPIOx_ISTAT, x=A..E).....	116
7.5.4.	Port output control register (GPIOx_OCTL, x=A..E).....	117
7.5.5.	Port bit operate register (GPIOx_BOP, x=A..E).....	117
7.5.6.	Port bit clear register (GPIOx_BC, x=A..E) .....	118
7.5.7.	Port configuration lock register (GPIOx_LOCK, x=A..E) .....	118
7.5.8.	Event control register (AFIO_EC).....	119
7.5.9.	AFIO port configuration register 0 (AFIO_PCF0).....	120
7.5.10.	EXTI sources selection register 0 (AFIO_EXTISSL0) .....	123
7.5.11.	EXTI sources selection register 1 (AFIO_EXTISSL1) .....	124
7.5.12.	EXTI sources selection register 2 (AFIO_EXTISSL2) .....	125
7.5.13.	EXTI sources selection register 3 (AFIO_EXTISSL3) .....	126
7.5.14.	AFIO port configuration register 1 (AFIO_PCF1) .....	127
<b>8.</b>	<b>CRC calculation unit (CRC) .....</b>	<b>129</b>
8.1.	<b>Overview .....</b>	<b>129</b>
8.2.	<b>Characteristics.....</b>	<b>129</b>
8.3.	<b>Function overview.....</b>	<b>130</b>
8.4.	<b>Register definition.....</b>	<b>131</b>
8.4.1.	Data register (CRC_DATA) .....	131
8.4.2.	Free data register (CRC_FDATA) .....	131
8.4.3.	Control register (CRC_CTL).....	132
<b>9.</b>	<b>Direct memory access controller (DMA) .....</b>	<b>133</b>
9.1.	<b>Overview .....</b>	<b>133</b>
9.2.	<b>Characteristics.....</b>	<b>133</b>
9.3.	<b>Block diagram .....</b>	<b>134</b>
9.4.	<b>Function overview.....</b>	<b>134</b>
9.4.1.	DMA operation.....	134
9.4.2.	Peripheral handshake .....	136
9.4.3.	Arbitration.....	136
9.4.4.	Address generation .....	137

9.4.5.	Circular mode .....	137
9.4.6.	Memory to memory mode.....	137
9.4.7.	Channel configuration .....	137
9.4.8.	Interrupt.....	138
9.4.9.	DMA request mapping .....	139
<b>9.5.</b>	<b>Register definition.....</b>	<b>142</b>
9.5.1.	Interrupt flag register (DMA_INTF).....	142
9.5.2.	Interrupt flag clear register (DMA_INTC) .....	143
9.5.3.	Channel x control register (DMA_CHxCTL).....	143
9.5.4.	Channel x counter register (DMA_CHxCNT).....	145
9.5.5.	Channel x peripheral base address register (DMA_CHxPADDR).....	146
9.5.6.	Channel x memory base address register (DMA_CHxMADDR) .....	146
<b>10.</b>	<b>Debug (DBG) .....</b>	<b>148</b>
10.1.	<b>Overview.....</b>	<b>148</b>
10.2.	<b>JTAG function overview.....</b>	<b>148</b>
10.2.1.	Pin assignment.....	148
10.2.2.	JTAG daisy chained structure .....	148
10.2.3.	Debug reset .....	149
10.3.	<b>Debug hold function overview .....</b>	<b>149</b>
10.3.1.	Debug support for power saving mode .....	149
10.3.2.	Debug support for TIMER, I2C, WWDGT, FWDGT and CAN .....	149
10.4.	<b>Register definition .....</b>	<b>150</b>
10.4.1.	ID code register (DBG_ID) .....	150
10.4.2.	Control register (DBG_CTL).....	150
<b>11.</b>	<b>Analog-to-digital converter (ADC).....</b>	<b>153</b>
11.1.	<b>Introduction .....</b>	<b>153</b>
11.2.	<b>Main features .....</b>	<b>153</b>
11.3.	<b>Pins and internal signals.....</b>	<b>154</b>
11.4.	<b>Functional description .....</b>	<b>155</b>
11.4.1.	Calibration (CLB) .....	155
11.4.2.	ADC clock .....	156
11.4.3.	ADCON switch .....	156
11.4.4.	Regular and inserted channel groups.....	156
11.4.5.	Conversion modes.....	156
11.4.6.	Inserted channel management .....	160
11.4.7.	Data alignment .....	161
11.4.8.	Programmable sample time .....	162
11.4.9.	External trigger .....	163
11.4.10.	DMA request.....	163
11.4.11.	Temperature sensor, and internal reference voltage V <sub>REFINT</sub> .....	163

---

11.4.12.	Programmable resolution (DRES) - fast conversion mode .....	164
11.4.13.	On-chip hardware oversampling .....	165
<b>11.5.</b>	<b>ADC sync mode.....</b>	<b>166</b>
<b>11.6.</b>	<b>Free mode.....</b>	<b>167</b>
11.6.1.	Regular parallel mode .....	168
11.6.2.	Inserted parallel mode .....	168
11.6.3.	Follow-up fast mode .....	169
11.6.4.	Follow-up slow mode .....	169
11.6.5.	Trigger rotation mode .....	170
11.6.6.	Combined regular parallel & inserted parallel mode .....	171
11.6.7.	Combined regular parallel & trigger rotation mode .....	171
11.6.8.	Combined inserted parallel & follow-up mode.....	172
<b>11.7.</b>	<b>ADC interrupts.....</b>	<b>173</b>
<b>11.8.</b>	<b>ADC registers .....</b>	<b>174</b>
11.8.1.	Status register (ADC_STAT) .....	174
11.8.2.	Control register 0 (ADC_CTL0) .....	175
11.8.3.	Control register 1 (ADC_CTL1) .....	177
11.8.4.	Sample time register 0 (ADC_SAMPT0).....	179
11.8.5.	Sample time register 1 (ADC_SAMPT1).....	180
11.8.6.	Inserted channel data offset register x (ADC_IOFFx) (x=0..3) .....	181
11.8.7.	Watchdog high threshold register (ADC_WDHT) .....	181
11.8.8.	Watchdog low threshold register (ADC_WDLT).....	182
11.8.9.	Regular sequence register 0 (ADC_RSQ0).....	182
11.8.10.	Regular sequence register 1 (ADC_RSQ1).....	183
11.8.11.	Regular sequence register 2 (ADC_RSQ2).....	183
11.8.12.	Inserted sequence register (ADC_ISQ) .....	184
11.8.13.	Inserted data register x (ADC_IDATAx) (x= 0..3) .....	185
11.8.14.	Regular data register (ADC_RDATA) .....	185
11.8.15.	Oversample control register (ADC_OVSAMPCTL) .....	186
<b>12.</b>	<b>Digital-to-analog converter (DAC).....</b>	<b>188</b>
<b>12.1.</b>	<b>Overview.....</b>	<b>188</b>
<b>12.2.</b>	<b>Characteristics .....</b>	<b>188</b>
<b>12.3.</b>	<b>Function overview .....</b>	<b>189</b>
12.3.1.	DAC enable .....	189
12.3.2.	DAC output buffer .....	190
12.3.3.	DAC data configuration.....	190
12.3.4.	DAC trigger .....	190
12.3.5.	DAC conversion .....	190
12.3.6.	DAC noise wave .....	191
12.3.7.	DAC output voltage .....	192
12.3.8.	DMA request.....	192

---

12.3.9. DAC concurrent conversion .....	192
<b>12.4. Register definition .....</b>	<b>193</b>
12.4.1. Control register (DAC_CTL).....	193
12.4.2. Software trigger register (DAC_SWT) .....	195
12.4.3. DAC0 12-bit right-aligned data holding register (DAC0_R12DH).....	196
12.4.4. DAC0 12-bit left-aligned data holding register (DAC0_L12DH).....	196
12.4.5. DAC0 8-bit right-aligned data holding register (DAC0_R8DH) .....	197
12.4.6. DAC1 12-bit right-aligned data holding register (DAC1_R12DH).....	197
12.4.7. DAC1 12-bit left-aligned data holding register (DAC1_L12DH).....	198
12.4.8. DAC1 8-bit right-aligned data holding register (DAC1_R8DH) .....	198
12.4.9. DAC concurrent mode 12-bit right-aligned data holding register (DACC_R12DH) .....	199
12.4.10. DAC concurrent mode 12-bit left-aligned data holding register (DACC_L12DH) .....	199
12.4.11. DAC concurrent mode 8-bit right-aligned data holding register (DACC_R8DH) .....	200
12.4.12. DAC0 data output register (DAC0_DO) .....	200
12.4.13. DAC1 data output register (DAC1_DO) .....	201
<b>13. Watchdog timer (WDGT).....</b>	<b>202</b>
<b>13.1. Free watchdog timer (FWDGT) .....</b>	<b>202</b>
13.1.1. Overview .....	202
13.1.2. Characteristics.....	202
13.1.3. Function overview.....	202
13.1.4. Register definition .....	205
<b>13.2. Window watchdog timer (WWDT) .....</b>	<b>208</b>
13.2.1. Overview .....	208
13.2.2. Characteristics.....	208
13.2.3. Function overview.....	208
13.2.4. Register definition .....	211
<b>14. Real-time Clock (RTC).....</b>	<b>213</b>
<b>14.1. Overview.....</b>	<b>213</b>
<b>14.2. Characteristics .....</b>	<b>213</b>
<b>14.3. Function overview .....</b>	<b>213</b>
14.3.1. RTC reset.....	214
14.3.2. RTC reading .....	214
14.3.3. RTC configuration.....	215
14.3.4. RTC flag assertion .....	215
<b>14.4. Register definition .....</b>	<b>217</b>
14.4.1. RTC interrupt enable register(RTC_INTEN) .....	217
14.4.2. RTC control register(RTC_CTL) .....	217
14.4.3. RTC prescaler high register (RTC_PSCH) .....	218
14.4.4. RTC prescaler low register (RTC_PSCL) .....	219
14.4.5. RTC divider high register (RTC_DIVH) .....	219

14.4.6.	RTC divider low register (RTC_DIVL).....	219
14.4.7.	RTC counter high register (RTC_CNTH) .....	220
14.4.8.	RTC counter low register (RTC_CNTL).....	220
14.4.9.	RTC alarm high register (RTC_ALRMH).....	221
14.4.10.	RTC alarm low register (RTC_ALRML).....	221
<b>15.</b>	<b>Timer(TIMERx) .....</b>	<b>222</b>
<b>15.1.</b>	<b>Advanced timer (TIMERx, x=0) .....</b>	<b>223</b>
15.1.1.	Overview .....	223
15.1.2.	Characteristics.....	223
15.1.3.	Block diagram.....	224
15.1.4.	Function overview.....	224
15.1.5.	TIMERx registers(x=0) .....	254
<b>15.2.</b>	<b>General level0 timer (TIMERx, x=1, 2, 3, 4).....</b>	<b>279</b>
15.2.1.	Overview .....	279
15.2.2.	Characteristics.....	279
15.2.3.	Block diagram.....	279
15.2.4.	Function overview.....	280
15.2.5.	TIMERx registers(x=1,2,3,4) .....	297
<b>15.3.</b>	<b>Basic timer (TIMERx, x=5, 6) .....</b>	<b>318</b>
15.3.1.	Overview .....	318
15.3.2.	Characteristics.....	318
15.3.3.	Block diagram.....	318
15.3.4.	Function overview.....	318
15.3.5.	TIMERx registers(x=5,6) .....	323
<b>16.</b>	<b>Universal synchronous/asynchronous receiver /transmitter (USART) .....</b>	<b>328</b>
<b>16.1.</b>	<b>Overview.....</b>	<b>328</b>
<b>16.2.</b>	<b>Characteristics .....</b>	<b>328</b>
<b>16.3.</b>	<b>Function overview .....</b>	<b>329</b>
16.3.1.	USART frame format.....	330
16.3.2.	Baud rate generation.....	331
16.3.3.	USART transmitter.....	331
16.3.4.	USART receiver .....	333
16.3.5.	Use DMA for data buffer access.....	334
16.3.6.	Hardware flow control .....	335
16.3.7.	Multi-processor communication.....	336
16.3.8.	LIN mode.....	337
16.3.9.	Synchronous mode.....	338
16.3.10.	IrDA SIR ENDEC mode .....	339
16.3.11.	Half-duplex communication mode.....	341
16.3.12.	Smartcard (ISO7816-3) mode .....	341
16.3.13.	USART interrupts.....	342

---

<b>16.4. Register definition .....</b>	<b>344</b>
16.4.1. Status register (USART_STAT) .....	344
16.4.2. Data register (USART_DATA).....	346
16.4.3. Baud rate register (USART_BAUD).....	346
16.4.4. Control register 0 (USART_CTL0) .....	347
16.4.5. Control register 1 (USART_CTL1) .....	349
16.4.6. Control register 2 (USART_CTL2) .....	350
16.4.7. Guard time and prescaler register (USART_GP).....	352
<b>17. Inter-integrated circuit interface (I2C) .....</b>	<b>354</b>
<b>17.1. Overview.....</b>	<b>354</b>
<b>17.2. Characteristics .....</b>	<b>354</b>
<b>17.3. Function overview .....</b>	<b>354</b>
17.3.1. SDA and SCL lines .....	355
17.3.2. Data validation.....	356
17.3.3. START and STOP condition.....	356
17.3.4. Clock synchronization .....	356
17.3.5. Arbitration.....	357
17.3.6. I2C communication flow.....	357
17.3.7. Programming model.....	358
17.3.8. SCL line stretching.....	367
17.3.9. Use DMA for data transfer.....	368
17.3.10. Packet error checking .....	368
17.3.11. SMBus support .....	368
17.3.12. Status, errors and interrupts .....	370
<b>17.4. Register definition .....</b>	<b>371</b>
17.4.1. Control register 0 (I2C_CTL0).....	371
17.4.2. Control register 1 (I2C_CTL1).....	373
17.4.3. Slave address register 0 (I2C_SADDR0).....	374
17.4.4. Slave address register 1 (I2C_SADDR1).....	374
17.4.5. Transfer buffer register (I2C_DATA).....	375
17.4.6. Transfer status register 0 (I2C_STAT0).....	375
17.4.7. Transfer status register 1 (I2C_STAT1).....	377
17.4.8. Clock configure register (I2C_CKCFG) .....	378
17.4.9. Rise time register (I2C_RT).....	379
<b>18. Serial peripheral interface/Inter-IC sound (SPI/I2S).....</b>	<b>380</b>
<b>18.1. Overview.....</b>	<b>380</b>
<b>18.2. Characteristics .....</b>	<b>380</b>
18.2.1. SPI characteristics .....	380
18.2.2. I2S characteristics .....	380
<b>18.3. SPI block diagram .....</b>	<b>381</b>

---

<b>18.4. SPI signal description .....</b>	<b>381</b>
18.4.1. Normal configuration .....	381
<b>18.5. SPI function overview .....</b>	<b>382</b>
18.5.1. SPI clock timing and data format.....	382
18.5.2. NSS function.....	382
18.5.3. SPI operating modes.....	383
18.5.4. DMA function .....	389
18.5.5. CRC function .....	389
<b>18.6. SPI interrupts .....</b>	<b>389</b>
18.6.1. Status flags .....	389
18.6.2. Error flags .....	390
<b>18.7. I2S block diagram .....</b>	<b>391</b>
<b>18.8. I2S signal description .....</b>	<b>391</b>
<b>18.9. I2S function overview .....</b>	<b>392</b>
18.9.1. I2S audio standards .....	392
18.9.2. I2S clock.....	400
18.9.3. Operation .....	401
18.9.4. DMA function .....	403
<b>18.10. I2S interrupts.....</b>	<b>404</b>
18.10.1. Status flags.....	404
18.10.2. Error flags .....	404
<b>18.11. Register definition .....</b>	<b>406</b>
18.11.1. Control register 0 (SPI_CTL0) .....	406
18.11.2. Control register 1 (SPI_CTL1) .....	408
18.11.3. Status register (SPI_STAT) .....	409
18.11.4. Data register (SPI_DATA).....	410
18.11.5. CRC polynomial register (SPI_CRCPOLY) .....	411
18.11.6. RX CRC register (SPI_RCRC) .....	411
18.11.7. TX CRC register (SPI_TCRC) .....	412
18.11.8. I2S control register (SPI_I2SCTL) .....	413
18.11.9. I2S clock prescaler register (SPI_I2SPSC) .....	414
<b>19. External memory controller (EXMC) .....</b>	<b>416</b>
<b>19.1. Overview .....</b>	<b>416</b>
<b>19.2. Characteristics.....</b>	<b>416</b>
<b>19.3. Function overview .....</b>	<b>416</b>
19.3.1. Block diagram.....	416
19.3.2. Basic regulation of EXMC access .....	417
19.3.3. External device address mapping .....	418
19.3.4. NOR/PSRAM controller .....	419

<b>19.4. Register definition .....</b>	<b>424</b>
19.4.1. NOR/PSRAM controller registers .....	424
<b>20. Controller area network (CAN) .....</b>	<b>427</b>
<b>20.1. Overview .....</b>	<b>427</b>
<b>20.2. Characteristics.....</b>	<b>427</b>
<b>20.3. Function overview .....</b>	<b>428</b>
20.3.1. Working mode .....	428
20.3.2. Communication modes .....	429
20.3.3. Data transmission .....	430
20.3.4. Data reception .....	432
20.3.5. Filtering function.....	433
20.3.6. Time-triggered communication .....	436
20.3.7. Communication parameters.....	437
20.3.8. Error flags .....	438
20.3.9. CAN interrupts.....	439
<b>20.4. Register definition .....</b>	<b>441</b>
20.4.1. Control register (CAN_CTL).....	441
20.4.2. Status register (CAN_STAT) .....	442
20.4.3. Transmit status register (CAN_TSTAT) .....	444
20.4.4. Receive message FIFO0 register (CAN_RFIFO0) .....	447
20.4.5. Receive message FIFO1 register (CAN_RFIFO1) .....	447
20.4.6. Interrupt enable register (CAN_INTEN) .....	448
20.4.7. Error register (CAN_ERR) .....	450
20.4.8. Bit timing register (CAN_BT).....	451
20.4.9. Transmit mailbox identifier register (CAN_TMI $x$ ) ( $x=0..2$ ) .....	452
20.4.10. Transmit mailbox property register (CAN_TMP $x$ ) ( $x=0..2$ ) .....	453
20.4.11. Transmit mailbox data0 register (CAN_TMDATA0 $x$ ) ( $x=0..2$ ) .....	453
20.4.12. Transmit mailbox data1 register (CAN_TMDATA1 $x$ ) ( $x=0..2$ ) .....	454
20.4.13. Receive FIFO mailbox identifier register (CAN_RFIFOMI $x$ ) ( $x=0,1$ ) .....	454
20.4.14. Receive FIFO mailbox property register (CAN_RFIFOMP $x$ ) ( $x=0,1$ ) .....	455
20.4.15. Receive FIFO mailbox data0 register (CAN_RFIFOMDATA0 $x$ ) ( $x=0,1$ ) .....	456
20.4.16. Receive FIFO mailbox data1 register (CAN_RFIFOMDATA1 $x$ ) ( $x=0,1$ ) .....	456
20.4.17. Filter control register (CAN_FCTL) .....	457
20.4.18. Filter mode configuration register (CAN_FMCFG) .....	457
20.4.19. Filter scale configuration register (CAN_FSCFG) .....	458
20.4.20. Filter associated FIFO register (CAN_FAFIFO) .....	458
20.4.21. Filter working register (CAN_FW) .....	459
20.4.22. Filter x data y register (CAN_FxDATAy) ( $x=0..27, y=0,1$ ) .....	459
<b>21. Universal serial bus full-speed interface (USBFS) .....</b>	<b>461</b>
<b>21.1. Overview.....</b>	<b>461</b>
<b>21.2. Characteristics.....</b>	<b>461</b>

<b>21.3.</b>	<b>Block diagram.....</b>	<b>462</b>
<b>21.4.</b>	<b>Signal description .....</b>	<b>462</b>
<b>21.5.</b>	<b>Function overview .....</b>	<b>462</b>
21.5.1.	USBFS clocks and working modes.....	462
21.5.2.	USB host function.....	464
21.5.3.	USB device function .....	466
21.5.4.	OTG function overview .....	467
21.5.5.	Data FIFO .....	468
21.5.6.	Operation guide.....	471
<b>21.6.</b>	<b>Interrupts.....</b>	<b>475</b>
<b>21.7.</b>	<b>Register definition .....</b>	<b>477</b>
21.7.1.	Global control and status registers.....	477
21.7.2.	Host control and status registers .....	498
21.7.3.	Device control and status registers.....	510
21.7.4.	Power and clock control register (USBFS_PWRCLKCTL).....	534
<b>22.</b>	<b>Revision history.....</b>	<b>535</b>

# List of Figures

Figure 1-1. GD32VF103 system architecture .....	24
Figure 2-1. Process of page erase operation .....	34
Figure 2-2. Process of mass erase operation .....	35
Figure 2-3. Process of word program operation .....	37
Figure 3-1. Power supply overview .....	48
Figure 3-2. Waveform of the POR/PDR .....	50
Figure 3-3. Waveform of the LVD threshold .....	50
Figure 5-1. The system reset circuit .....	63
Figure 5-2. Clock tree .....	64
Figure 5-3. HXTAL clock source .....	65
Figure 6-1. Block diagram of EXTI .....	97
Figure 7.1. Basic structure of a standard I/O port bit .....	103
Figure 7.2. Input configuration .....	104
Figure 7.3. Output configuration .....	105
Figure 7.4. Analog configuration .....	106
Figure 7.5. Alternate function configuration .....	106
Figure 8-1. Block diagram of CRC calculation unit .....	129
Figure 9-1. Block diagram of DMA .....	134
Figure 9-2. Handshake mechanism .....	136
Figure 9-3. DMA interrupt logic .....	138
Figure 9-4. DMA0 request mapping .....	139
Figure 9-5. DMA1 request mapping .....	140
Figure 11-1. ADC module block diagram .....	155
Figure 11-2. Single conversion mode .....	156
Figure 11-3. Continuous conversion mode .....	157
Figure 11-4. Scan conversion mode, continuous disable .....	158
Figure 11-5. Scan conversion mode, continuous enable .....	159
Figure 11-6. Discontinuous conversion mode .....	160
Figure 11-7. Auto-insertion, CNT = 1 .....	161
Figure 11-8. Triggered insertion .....	161
Figure 11-9. 12-bit Data alignment .....	162
Figure 11-10. 6-bit Data alignment .....	162
Figure 11-11. 20-bit to 16-bit result truncation .....	165
Figure 11-12. Numerical example with 5-bits shift and rounding .....	166
Figure 11-13. ADC sync block diagram .....	167
Figure 11-14. Regular parallel mode on 16 channels .....	168
Figure 11-15. Inserted parallel mode on 4 channels .....	169
Figure 11-16. Follow-up fast mode on 1 channel in continuous conversion mode .....	169
Figure 11-17. Follow-up slow mode on 1 channel .....	170
Figure 11-18. Trigger rotation: inserted channel group .....	171

Figure 11-19. Trigger rotation: inserted channels in discontinuous mode .....	171
Figure 11-20. Regular parallel & trigger rotation mode .....	172
Figure 11-21. Trigger occurs during inserted conversion .....	172
Figure 11-22 Follow-up single channel with inserted sequence CH1, CH2 .....	173
Figure 12-1. DAC block diagram.....	189
Figure 12-2. DAC LFSR algorithm .....	191
Figure 12-3. DAC triangle noise wave .....	191
Figure 13.1. Free watchdog block diagram .....	203
Figure 13.2. Window watchdog timer block diagram .....	209
Figure 13.3. Window watchdog timing diagram .....	210
Figure 14.1. Block diagram of RTC .....	214
Figure 14.2. RTC second and alarm waveform example (RTC_PSC = 3, RTC_ALRM = 2).....	215
Figure 14.3. RTC second and overflow waveform example (RTC_PSC= 3).....	216
Figure 15-1. Advanced timer block diagram .....	224
Figure 15-2. Normal mode, internal clock divided by 1 .....	225
Figure 15-3. Counter timing diagram with prescaler division change from 1 to 2 .....	226
Figure 15-4. Timing chart of up counting mode, PSC=0/1 .....	227
Figure 15-5. Timing chart of up counting mode, change TIMERx_CAR ongoing .....	228
Figure 15-6. Timing chart of down counting mode, PSC=0/1 .....	229
Figure 15-7. Timing chart of down counting mode, change TIMERx_CAR ongoing .....	230
Figure 15-8. Timing chart of center-aligned counting mode .....	231
Figure 15-9. Repetition counter timing chart of center-aligned counting mode.....	232
Figure 15-10. Repetition counter timing chart of up counting mode.....	232
Figure 15-11. Repetition counter timing chart of down counting mode.....	233
Figure 15-12. Input capture logic.....	234
Figure 15-13. Output compare logic (with complementary output, x=0,1,2) .....	235
Figure 15-14. Output compare logic (CH3_O) .....	235
Figure 15-15. Output-compare in three modes .....	237
Figure 15-16. Timing chart of EAPWM.....	238
Figure 15-17. Timing chart of CAPWM .....	238
Figure 15-18. Complementary output with dead time insertion.....	241
Figure 15-19. Output behavior of the channel in response to a break (the break high active) .....	242
Figure 15-20. Example of counter operation in encoder interface mode .....	243
Figure 15-21. Example of encoder interface mode with CI0FE0 polarity inverted .....	243
Figure 15-22. Hall sensor is used to BLDC motor .....	244
Figure 15-23. Hall sensor timing between two timers .....	245
Figure 15-24. Restart mode .....	246
Figure 15-25. Pause mode .....	246
Figure 15-26. Event mode .....	247
Figure 15-27. Single pulse mode TIMERx_CHxCV=0x04, TIMERx_CAR=0x60.....	248
Figure 15-28. TIMER0 Master/Slave mode timer example.....	248
Figure 15-29. Triggering TIMER0 with enable signal of TIMER2.....	249
Figure 15-30. Triggering TIMER0 with update signal of TIMER2 .....	250
Figure 15-31. Pause TIMER0 with enable signal of TIMER2 .....	251

---

Figure 15-32. Pause TIMER0 with O0CPREF signal of Timer2 .....	251
Figure 15-33. Triggering TIMER0 and TIMER2 with TIMER2's CI0 input .....	252
Figure 15-34. General Level 0 timer block diagram .....	280
Figure 15-35. Normal mode, internal clock divided by 1 .....	281
Figure 15-36. Counter timing diagram with prescaler division change from 1 to 2 .....	282
Figure 15-37. Timing chart of up counting mode, PSC=0/1 .....	283
Figure 15-38. Timing chart of up counting mode, change TIMERx_CAR ongoing .....	284
Figure 15-39. Timing chart of down counting mode, PSC=0/1 .....	285
Figure 15-40. Timing chart of down counting mode, change TIMERx_CAR .....	286
Figure 15-41. Timing chart of center-aligned counting mode .....	287
Figure 15-42. Input capture logic .....	288
Figure 15-43. Output-compare in three modes .....	290
Figure 15-44. EAPWM timechart .....	291
Figure 15-45. CAPWM timechart .....	291
Figure 15-46. Example of counter operation in encoder interface mode .....	293
Figure 15-47. Example of encoder interface mode with CI0FE0 polarity inverted .....	293
Figure 15-48. Restart mode .....	294
Figure 15-49. Pause mode .....	295
Figure 15-50. Event mode .....	295
Figure 15-51. Single pulse mode TIMERx_CHxCV = 0x04 TIMERx_CAR=0x60 .....	296
Figure 15-52. Basic timer block diagram .....	318
Figure 15-53. Normal mode, internal clock divided by 1 .....	319
Figure 15-54. Counter timing diagram with prescaler division change from 1 to 2 .....	320
Figure 15-55. Timing chart of up counting mode, PSC=0/1 .....	321
Figure 15-56. Timing chart of up counting mode, change TIMERx_CAR ongoing .....	322
Figure 16-1. USART module block diagram .....	330
Figure 16-2. USART character frame (8 bits data and 1 stop bit) .....	330
Figure 16-3. USART transmit procedure .....	332
Figure 16-4. Receiving a frame bit by oversampling method .....	333
Figure 16-5. Configuration steps when using DMA for USART transmission .....	334
Figure 16-6. Configuration steps when using DMA for USART reception .....	335
Figure 16-7. Hardware flow control between two USARTs .....	336
Figure 16-8. Hardware flow control .....	336
Figure 16-9. Break frame occurs during idle state .....	338
Figure 16-10. Break frame occurs during a frame .....	338
Figure 16-11. Example of USART in synchronous mode .....	339
Figure 16-12. 8-bit format USART synchronous waveform (CLEN=1) .....	339
Figure 16-13. IrDA SIR ENDEC module .....	340
Figure 16-14. IrDA data modulation .....	340
Figure 16-15. ISO7816-3 frame format .....	341
Figure 16-16. USART interrupt mapping diagram .....	343
Figure 17-1. I2C module block diagram .....	355
Figure 17-2. Data validation .....	356
Figure 17-3. START and STOP condition .....	356

---

Figure 17-4. Clock synchronization .....	357
Figure 17-5. SDA Line arbitration .....	357
Figure 17-6. I2C communication flow with 7-bit address.....	358
Figure 17-7. I2C communication flow with 10-bit address (Master Transmit).....	358
Figure 17-8. I2C communication flow with 10-bit address (Master Receive) .....	358
Figure 17-9. Programming model for slave transmitting(10-bit address mode).....	360
Figure 17-10. Programming model for slave receiving(10-bit address mode).....	361
Figure 17-11. Programming model for master transmitting(10-bit address mode).....	363
Figure 17-12. Programming model for master receiving using Solution A(10-bit address mode) ..	365
Figure 17-13. Programming model for master receiving using solution B(10-bit address mode) ..	367
Figure 18-1. Block diagram of SPI.....	381
Figure 18-2. SPI timing diagram in normal mode .....	382
Figure 18-3. A typical full-duplex connection .....	384
Figure 18-4. A typical simplex connection (Master: Receive, Slave: Transmit).....	384
Figure 18-5. A typical simplex connection (Master: Transmit only, Slave: Receive).....	384
Figure 18-6. A typical bidirectional connection .....	385
Figure 18-7. Timing diagram of TI master mode with discontinuous transfer.....	386
Figure 18-8. Timing diagram of TI master mode with continuous transfer .....	387
Figure 18-9. Timing diagram of TI slave mode .....	387
Figure 18-10. Timing diagram of NSS pulse with continuous transmission .....	388
Figure 18-11. Block diagram of I2S.....	391
Figure 18-12. I2S Phillips standard timing diagram (DTLEN=00, CHLEN=0, CKPL=0) .....	392
Figure 18-13. I2S Phillips standard timing diagram (DTLEN=00, CHLEN=0, CKPL=1) .....	392
Figure 18-14. I2S Phillips standard timing diagram (DTLEN=10, CHLEN=1, CKPL=0).....	393
Figure 18-15. I2S Phillips standard timing diagram (DTLEN=10, CHLEN=1, CKPL=1) .....	393
Figure 18-16. I2S Phillips standard timing diagram (DTLEN=01, CHLEN=1, CKPL=0) .....	393
Figure 18-17. I2S Phillips standard timing diagram (DTLEN=01, CHLEN=1, CKPL=1) .....	393
Figure 18-18. I2S Phillips standard timing diagram (DTLEN=00, CHLEN=1, CKPL=0) .....	394
Figure 18-19. I2S Phillips standard timing diagram (DTLEN=00, CHLEN=1, CKPL=1) .....	394
Figure 18-20. MSB justified standard timing diagram (DTLEN=00, CHLEN=0, CKPL=0) .....	394
Figure 18-21. MSB justified standard timing diagram (DTLEN=00, CHLEN=0, CKPL=1) .....	394
Figure 18-22. MSB justified standard timing diagram (DTLEN=10, CHLEN=1, CKPL=0) .....	394
Figure 18-23. MSB justified standard timing diagram (DTLEN=10, CHLEN=1, CKPL=1) .....	395
Figure 18-24. MSB justified standard timing diagram (DTLEN=01, CHLEN=1, CKPL=0) .....	395
Figure 18-25. MSB justified standard timing diagram (DTLEN=01, CHLEN=1, CKPL=1) .....	395
Figure 18-26. MSB justified standard timing diagram (DTLEN=00, CHLEN=1, CKPL=0) .....	395
Figure 18-27. MSB justified standard timing diagram (DTLEN=00, CHLEN=1, CKPL=1) .....	395
Figure 18-28. LSB justified standard timing diagram (DTLEN=01, CHLEN=1, CKPL=0) .....	396
Figure 18-29. LSB justified standard timing diagram (DTLEN=01, CHLEN=1, CKPL=1) .....	396
Figure 18-30. LSB justified standard timing diagram (DTLEN=00, CHLEN=1, CKPL=0) .....	396
Figure 18-31. LSB justified standard timing diagram (DTLEN=00, CHLEN=1, CKPL=1) .....	396
Figure 18-32. PCM standard short frame synchronization mode timing diagram (DTLEN=00, CHLEN=0, CKPL=0).....	397
Figure 18-33. PCM standard short frame synchronization mode timing diagram (DTLEN=00,	

---

CHLEN=0, CKPL=1).....	397
Figure 18-34. PCM standard short frame synchronization mode timing diagram (DTLEN=10, CHLEN=1, CKPL=0).....	397
Figure 18-35. PCM standard short frame synchronization mode timing diagram (DTLEN=10, CHLEN=1, CKPL=1).....	397
Figure 18-36. PCM standard short frame synchronization mode timing diagram (DTLEN=01, CHLEN=1, CKPL=0).....	397
Figure 18-37. PCM standard short frame synchronization mode timing diagram (DTLEN=01, CHLEN=1, CKPL=1).....	397
Figure 18-38. PCM standard short frame synchronization mode timing diagram (DTLEN=00, CHLEN=1, CKPL=0).....	398
Figure 18-39. PCM standard short frame synchronization mode timing diagram (DTLEN=00, CHLEN=1, CKPL=1).....	398
Figure 18-40. PCM standard long frame synchronization mode timing diagram (DTLEN=00, CHLEN=0, CKPL=0).....	398
Figure 18-41. PCM standard long frame synchronization mode timing diagram (DTLEN=00, CHLEN=0, CKPL=1).....	398
Figure 18-42. PCM standard long frame synchronization mode timing diagram (DTLEN=10, CHLEN=1, CKPL=0).....	398
Figure 18-43. PCM standard long frame synchronization mode timing diagram (DTLEN=10, CHLEN=1, CKPL=1).....	399
Figure 18-44. PCM standard long frame synchronization mode timing diagram (DTLEN=01, CHLEN=1, CKPL=0).....	399
Figure 18-45. PCM standard long frame synchronization mode timing diagram (DTLEN=01, CHLEN=1, CKPL=1).....	399
Figure 18-46. PCM standard long frame synchronization mode timing diagram (DTLEN=00, CHLEN=1, CKPL=0).....	399
Figure 18-47. PCM standard long frame synchronization mode timing diagram (DTLEN=00, CHLEN=1, CKPL=1).....	399
Figure 18-48. Block diagram of I2S clock generator .....	400
Figure 19-1. The EXMC block diagram .....	417
Figure 19-2. EXMC memory banks .....	418
Figure 19-3. Region of bank0 address mapping.....	418
Figure 19-4. Multiplex mode read access .....	421
Figure 19-5. Multiplex mode write access .....	422
Figure 19-6. Read access timing diagram under async-wait signal assertion.....	423
Figure 19-7. Write access timing diagram under async-wait signal assertion .....	423
Figure 20-1. CAN module block diagram .....	428
Figure 20-2. Transmission register .....	430
Figure 20-3. State of transmission mailbox.....	431
Figure 20-4. Reception register .....	432
Figure 20-5. 32-bit filter .....	434
Figure 20-6. 16-bit filter .....	434
Figure 20-7. 32-bit mask mode filter .....	434

---

Figure 20-8. 16-bit mask mode filter .....	434
Figure 20-9. 32-bit list mode filter.....	434
Figure 20-10. 16-bit list mode filter .....	434
Figure 20-11. The bit time .....	438
Figure 21-1. USBFS block diagram .....	462
Figure 21-2. Connection with host or device mode .....	463
Figure 21-3. Connection with OTG mode.....	464
Figure 21-4. State transition diagram of host port .....	464
Figure 21-5. HOST mode FIFO space in SRAM .....	469
Figure 21-6. Host mode FIFO access register map.....	469
Figure 21-7. Device mode FIFO space in SRAM .....	470
Figure 21-8. Device mode FIFO access register map .....	471

# List of Tables

Table 1-1. The interconnection relationship of the AHB interconnect matrix .....	23
Table 1-2. Memory map of GD32VF103 devices .....	25
Table 1-3. Boot modes .....	29
Table 2-1. Base address and size for flash memory .....	32
Table 2-2. Option byte.....	38
Table 3-1. Power saving mode summary .....	52
Table 5-1. Clock output 0 source select .....	68
Table 5-2. 1.2V domain voltage selected in deep-sleep mode .....	68
Table 6-1. Interrupt vector table .....	95
Table 6-2. EXTI source .....	98
Table 7.1. GPIO configuration table .....	102
Table 7.2. Debug interface signals .....	108
Table 7.3. Debug port mapping.....	108
Table 7.4. TIMER0 alternate function remapping .....	109
Table 7.5. TIMER1 alternate function remapping .....	109
Table 7.6. TIMER2 alternate function remapping .....	109
Table 7.7. TIMER3 alternate function remapping .....	109
Table 7.8. TIMER4 alternate function remapping .....	110
Table 7.9. USART0 alternate function remapping.....	110
Table 7.10. USART1 alternate function remapping.....	110
Table 7.11. USART2 alternate function remapping.....	110
Table 7.12. I2C0 alternate function remapping.....	110
Table 7.13. SPI0 alternate function remapping.....	111
Table 7.14. SPI2/I2S2 alternate function remapping.....	111
Table 7.15. CAN0 alternate function remapping .....	111
Table 7.16. CAN1 alternate function remapping .....	111
Table 7.17. OSC32 pins configuration.....	112
Table 7.18. OSC pins configuration.....	112
Table 9-1. DMA transfer operation .....	135
Table 9-2. Interrupt events.....	138
Table 9-3. DMA0 requests for each channel.....	140
Table 9-4. DMA1 requests for each channel.....	141
Table 11-1. ADC internal signals .....	154
Table 11-2. ADC pins definition .....	154
Table 11-3. External trigger for regular channels for ADC0 and ADC1 .....	163
Table 11-4. External trigger for inserted channels for ADC0 and ADC1 .....	163
Table 11-5. $t_{CONV}$ timings depending on resolution .....	164
Table 11-6. Maximum output results vs N and M (Grayed values indicates truncation) .....	166
Table 12-1. DAC pins .....	189
Table 12-2. External triggers of DAC.....	190

Table 13.1. Min/max FWDGT timeout period at 40 kHz (IRC40K) .....	203
Table 13.2. Min/max timeout value at 54 MHz ( $f_{PCLK1}$ ) .....	210
Table 15-1. Timers (TIMERx) are devided into three sorts.....	222
Table 15-2. Complementary outputs controlled by parameters .....	240
Table 15-3. Counting direction versus encoder signals .....	243
Table 15-4. Slave mode example table .....	245
Table 15-5. Counting direction versus encoder signals .....	292
Table 15-6. Slave controller examples .....	294
Table 16-1. Description of USART important pins.....	329
Table 16-2. Stop bits configuration .....	330
Table 16-3. USART interrupt requests .....	342
Table 17-1. Definition of I2C-bus terminology (refer to the I2C specification of philips semiconductors) .....	355
Table17-2. Event status flags .....	370
Table17-3. I2C error flags .....	370
Table 18-1. SPI signal description.....	381
Table 18-2. SPI operating modes.....	383
Table 18-3. SPI interrupt requests.....	390
Table 18-4. I2S bitrate calculation formulas .....	400
Table 18-5. Audio sampling frequency calculation formulas .....	400
Table 18-6. Direction of I2S interface signals for each operation mode .....	401
Table 18-7. I2S interrupt .....	405
Table 19-1. NOR Flash interface signals description .....	419
Table 19-2. PSRAM muxed signal description .....	419
Table 19-3. EXMC bank 0 supports all transactions .....	420
Table 19-4. NOR / PSRAM controller timing parameters.....	421
Table 19-5. EXMC_timing models.....	421
Table 19-6. Multiplex mode related registers configuration .....	422
Table 20-1. 32-bit filter number .....	435
Table 20-2. Filtering index.....	435
Table 21-1. USBFS signal description .....	462
Table 21-2. USBFS global interrupt.....	475
Table 22-1. Revision history .....	535

## 1. System and memory architecture

The devices of GD32VF103 series are 32-bit general-purpose microcontrollers based on the 32bit RISC-V processor. The RISC-V processor includes three AHB buses known as I-Code bus, D-Code bus and System buse. All memory accesses of the RISC-V 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. RISC-V CPU

RISC-V CPU target for embedded applications that require low energy consumption and small area, which is compliant to RISC-V architecture with several efficient micro-architecture features, including simple dynamic branch prediction, instruction pre-fetch buffers and local memories. It supports 32 general purpose registers (GPRs) and fast multiplier for performance/area tradeoff:

- RISC-V compliant little-endian RV32IMAC (32GPRs) ;
- Configurable 2-stage pipeline optimized for low gate-count and high frequency;
- Machine (M) and User (U) Privilege levels support;
- Single-cycle hardware multiplier and Multi-cycles hardware divider support;
- Misaligned load/store hardware support;
- Atomic instructions hardware support;
- Non-maskable interrupt (NMI) support;
- Dynamic Branch Prediction and instruction pre-fetch buffers to speed up control code;
- State-of-the-art micro-architecture design to tradeoff area and performance requirements;
- WFI (Wait for Interrupt) support;
- WFE (Wait for Event) support;
- Interrupt priority levels configurable and programmable;
- Enhancement of vectored interrupt handling for real-time performance;
- Support interrupt preemption with priority ;
- Support interrupt tail chaining;
- Standard 4-wire JTAG debug port
- Support interactive debug functionalities
- Support 4 triggers for hardware breakpoint

### 1.2. System architecture

A 32-bit multilayer bus is implemented in the GD32VF103 devices, which makes the parallel access paths between multiple masters and slaves in the system possible. The multilayer bus consists of an AHB interconnect matrix, one AHB bus and two APB buses. The

interconnection relationship of the AHB interconnect matrix is shown below. In [\*\*Table 1-1. The interconnection relationship of the AHB interconnect matrix\*\*](#), “1” indicates the corresponding master is able to access the corresponding slave through the AHB interconnect matrix, the blank indicates the corresponding master cannot access the corresponding slave through the AHB interconnect matrix.

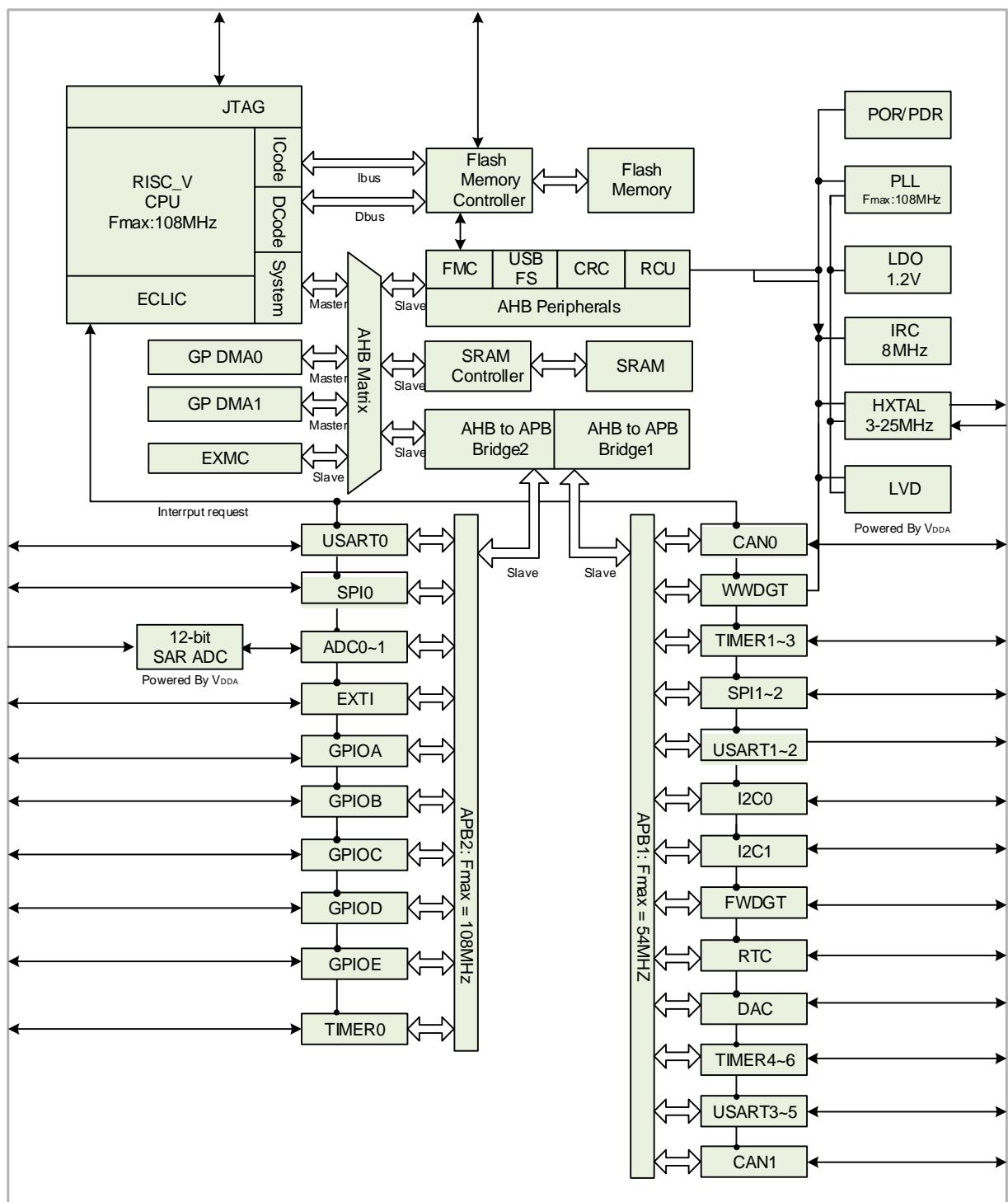
**Table 1-1. The interconnection relationship of the AHB interconnect matrix**

	IBUS	DBUS	SBUS	DMA0	DMA1
FMC-I	1				
FMC-D		1		1	1
SRAM	1	1	1	1	1
EXMC	1	1	1	1	1
AHB			1	1	1
APB1			1	1	1
APB2			1	1	1

As is shown above, there are several masters connected with the AHB interconnect matrix, including IBUS, DBUS, SBUS, DMA0 and DMA1. IBUS is the instruction bus of the RISC-V core, which is used for fetching instruction/vector from the Code region (0x0000 0000 ~ 0x1FFF FFFF). DBUS is the data bus of the RISC-V core, which is used for loading/storing data and also for debugging access of the Code region. Similarly, SBUS is the system bus of the RISC-V core, which is used for fetching instruction/vector, loading/storing data and debugging access of the system regions. The System regions include the internal SRAM region and the Peripheral region. DMA0 and DMA1 are the buses of DMA0 and DMA1 respectively.

There are also several slaves connected with the AHB interconnect matrix, including FMC-I, FMC-D, SRAM, EXMC, AHB, APB1 and APB2. FMC-I is the instruction bus of the flash memory controller, FMC-D is the data bus of the flash memory controller. SRAM is on-chip static random access memories. EXMC is the external memory controller. AHB is the AHB bus connected with all AHB slaves, APB1 and APB2 connected with all APB slaves and all APB peripherals. APB1 is limited to 54 MHz, APB2 operates at full speed (up to 108MHz depending on the device).

As shown in the following figure, these are interconnected using the multilayer AHB bus architecture.

**Figure 1-1. GD32VF103 system architecture**


### 1.3. Memory map

The RISC-V processor is structured using a Harvard architecture which uses 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 .The maximum address range of the RISC-V is 4-Gbyte due to its 32-bit bus address width. Additionally, a pre-defined memory map is provided by the RISC-V processor to reduce the software complexity of repeated implementation for different device vendors. In the map, some regions are used by the RISC-V system peripherals which can not be modified. However, the other regions are available to the vendors. [\*\*Table 1-2. Memory map of GD32VF103 devices\*\*](#) shows the memory map of the GD32VF103 series devices, including Code, SRAM, peripheral, and other pre-defined regions. Almost each peripheral is allocated 1KB of space. This allows simplifying the address decoding for each peripheral.

**Table 1-2. Memory map of GD32VF103 devices**

Pre-defined Regions	Bus	Address	Peripherals	
External device	AHB	0xA000 0000 - 0xA000 0FFF	EXMC - SWREG	
External RAM		0x9000 0000 - 0x9FFF FFFF	Reserved	
		0x7000 0000 - 0x8FFF FFFF	Reserved	
		0x6000 0000 - 0x6FFF FFFF	EXMC - NOR/PSRAM/SRAM M	
Peripheral	AHB	0x5000 0000 - 0x5003 FFFF	USBFS	
		0x4008 0000 - 0x4FFF FFFF	Reserved	
		0x4004 0000 - 0x4007 FFFF	Reserved	
		0x4002 BC00 - 0x4003 FFFF	Reserved	
		0x4002 B000 - 0x4002 BBFF	Reserved	
		0x4002 A000 - 0x4002 AFFF	Reserved	
		0x4002 8000 - 0x4002 9FFF	Reserved	
		0x4002 6800 - 0x4002 7FFF	Reserved	
		0x4002 6400 - 0x4002 67FF	Reserved	
		0x4002 6000 - 0x4002 63FF	Reserved	
		0x4002 5000 - 0x4002 5FFF	Reserved	
		0x4002 4000 - 0x4002 4FFF	Reserved	
		0x4002 3C00 - 0x4002 3FFF	Reserved	
		0x4002 3800 - 0x4002 3BFF	Reserved	
		0x4002 3400 - 0x4002 37FF	Reserved	
		0x4002 3000 - 0x4002 33FF	CRC	
		0x4002 2C00 - 0x4002 2FFF	Reserved	
		0x4002 2800 - 0x4002 2BFF	Reserved	
		0x4002 2400 - 0x4002 27FF	Reserved	
		0x4002 2000 - 0x4002 23FF	FMC	
		0x4002 1C00 - 0x4002 1FFF	Reserved	
		0x4002 1800 - 0x4002 1BFF	Reserved	

<b>Pre-defined Regions</b>	<b>Bus</b>	<b>Address</b>	<b>Peripherals</b>
APB2		0x4002 1400 - 0x4002 17FF	Reserved
		0x4002 1000 - 0x4002 13FF	RCU
		0x4002 0C00 - 0x4002 0FFF	Reserved
		0x4002 0800 - 0x4002 0BFF	Reserved
		0x4002 0400 - 0x4002 07FF	DMA1
		0x4002 0000 - 0x4002 03FF	DMA0
		0x4001 8400 - 0x4001 FFFF	Reserved
		0x4001 8000 - 0x4001 83FF	Reserved
		0x4001 7C00 - 0x4001 7FFF	Reserved
		0x4001 7800 - 0x4001 7BFF	Reserved
		0x4001 7400 - 0x4001 77FF	Reserved
		0x4001 7000 - 0x4001 73FF	Reserved
		0x4001 6C00 - 0x4001 6FFF	Reserved
		0x4001 6800 - 0x4001 6BFF	Reserved
		0x4001 5C00 - 0x4001 67FF	Reserved
		0x4001 5800 - 0x4001 5BFF	Reserved
		0x4001 5400 - 0x4001 57FF	Reserved
		0x4001 5000 - 0x4001 53FF	Reserved
		0x4001 4C00 - 0x4001 4FFF	Reserved
		0x4001 4800 - 0x4001 4BFF	Reserved
		0x4001 4400 - 0x4001 47FF	Reserved
		0x4001 4000 - 0x4001 43FF	Reserved
		0x4001 3C00 - 0x4001 3FFF	Reserved
	APB1	0x4001 3800 - 0x4001 3BFF	USART0
		0x4001 3400 - 0x4001 37FF	Reserved
		0x4001 3000 - 0x4001 33FF	SPI0
		0x4001 2C00 - 0x4001 2FFF	TIMER0
		0x4001 2800 - 0x4001 2BFF	ADC1
		0x4001 2400 - 0x4001 27FF	ADC0
		0x4001 2000 - 0x4001 23FF	Reserved
		0x4001 1C00 - 0x4001 1FFF	Reserved
		0x4001 1800 - 0x4001 1BFF	GPIOE
		0x4001 1400 - 0x4001 17FF	GPIOD
		0x4001 1000 - 0x4001 13FF	GPIOC
		0x4001 0C00 - 0x4001 0FFF	GPIOB
		0x4001 0800 - 0x4001 0BFF	GPIOA
		0x4001 0400 - 0x4001 07FF	EXTI
		0x4001 0000 - 0x4001 03FF	AFIO

<b>Pre-defined Regions</b>	<b>Bus</b>	<b>Address</b>	<b>Peripherals</b>
		0x4000 C400 - 0x4000 C7FF	Reserved
		0x4000 C000 - 0x4000 C3FF	Reserved
		0x4000 8000 - 0x4000 BFFF	Reserved
		0x4000 7C00 - 0x4000 7FFF	Reserved
		0x4000 7800 - 0x4000 7BFF	Reserved
		0x4000 7400 - 0x4000 77FF	DAC
		0x4000 7000 - 0x4000 73FF	PMU
		0x4000 6C00 - 0x4000 6FFF	BKP
		0x4000 6800 - 0x4000 6BFF	CAN1
		0x4000 6400 - 0x4000 67FF	CAN0
		0x4000 6000 - 0x4000 63FF	Shared USB/CAN SRAM 512 bytes
		0x4000 5C00 - 0x4000 5FFF	USB device FS registers
		0x4000 5800 - 0x4000 5BFF	I2C1
		0x4000 5400 - 0x4000 57FF	I2C0
		0x4000 5000 - 0x4000 53FF	UART4
		0x4000 4C00 - 0x4000 4FFF	UART3
		0x4000 4800 - 0x4000 4BFF	USART2
		0x4000 4400 - 0x4000 47FF	USART1
		0x4000 4000 - 0x4000 43FF	Reserved
		0x4000 3C00 - 0x4000 3FFF	SPI2/I2S2
		0x4000 3800 - 0x4000 3BFF	SPI1/I2S1
		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	Reserved
		0x4000 1C00 - 0x4000 1FFF	Reserved
		0x4000 1800 - 0x4000 1BFF	Reserved
		0x4000 1400 - 0x4000 17FF	TIMER6
		0x4000 1000 - 0x4000 13FF	TIMER5
		0x4000 0C00 - 0x4000 0FFF	TIMER4
		0x4000 0800 - 0x4000 0BFF	TIMER3
		0x4000 0400 - 0x4000 07FF	TIMER2
		0x4000 0000 - 0x4000 03FF	TIMER1
SRAM	AHB	0x2007 0000 - 0x3FFF FFFF	Reserved
		0x2006 0000 - 0x2006 FFFF	Reserved
		0x2003 0000 - 0x2005 FFFF	Reserved

Pre-defined Regions	Bus	Address	Peripherals
		0x2002 0000 - 0x2002 FFFF	Reserved
		0x2001 C000 - 0x2001 FFFF	Reserved
		0x2001 8000 - 0x2001 BFFF	Reserved
		0x2000 5000 - 0x2001 7FFF	SRAM
		0x2000 0000 - 0x2000 4FFF	
Code	AHB	0x1FFF F810 - 0x1FFF FFFF	Reserved
		0x1FFF F800 - 0x1FFF F80F	Option Bytes
		0x1FFF B000 - 0x1FFF F7FF	Boot loader
		0x1FFF 7A10 - 0x1FFF AFFF	Reserved
		0x1FFF 7800 - 0x1FFF 7A0F	Reserved
		0x1FFF 0000 - 0x1FFF 77FF	Reserved
		0x1FFE C010 - 0x1FFE FFFF	Reserved
		0x1FFE C000 - 0x1FFE C00F	Reserved
		0x1001 0000 - 0x1FFE BFFF	Reserved
		0x1000 0000 - 0x1000 FFFF	Reserved
		0x083C 0000 - 0x0FFF FFFF	Reserved
		0x0802 0000 - 0x083B FFFF	Reserved
		0x0800 0000 - 0x0801 FFFF	Main Flash
		0x0030 0000 - 0x07FF FFFF	Reserved
		0x0010 0000 - 0x002F FFFF	Aliased to Main Flash or Boot loader
		0x0002 0000 - 0x000F FFFF	
		0x0000 0000 - 0x0001 FFFF	

### 1.3.1. On-chip SRAM memory

The GD32VF103 series of devices contain up to 32 KB of on-chip SRAM which address starts at 0x2000 0000. It supports byte, half-word (16 bits), and word (32 bits) accesses.

### 1.3.2. On-chip flash memory overview

The devices provide high density on-chip flash memory, which is organized as follows:

- Up to 128KB of main flash memory.
- Up to 18KB of information blocks for the boot loader.
- Option bytes to configure the device.

Refer to [Flash memory controller \(FMC\)](#) Chapter for more details.

## 1.4. Boot configuration

The GD32VF103 devices provide three kinds of boot sources which can be selected by the BOOT0 and BOOT1 pins. The details are shown in the following table. The value on the two pins is latched on the 4th rising edge of CK\_SYS after a reset. User can select the required boot source by set the BOOT0 and BOOT1 pins after a power-on reset or a system reset. Once the two pins have been sampled, they are free and can be used for other purposes.

**Table 1-3. Boot modes**

Selected boot source	Boot mode selection pins	
	Boot1	Boot0
Main Flash Memory	x	0
Boot loader	0	1
On-chip SRAM	1	1

**Note:** When the boot source is hoped to be set as “Main Flash Memory”, the Boot0 pin has to be connected with GND definitely and can not be floating.

After power-on sequence or a system reset, the RISC-V processor fetches the top-of-stack value from address 0x0000 0000 and the base address of boot code from 0x0000 0004 in sequence. Then, starts code execution from the base address of boot code.

The embedded boot loader is located in the System memory, which is used to reprogram the Flash memory. In GD32VF103 devices, the boot loader can be activated through the USART0 (PA9 and PA10), USART1 (PD5 and PD6), USBFS in device mode (PA9, PA11 and PA12) interface.

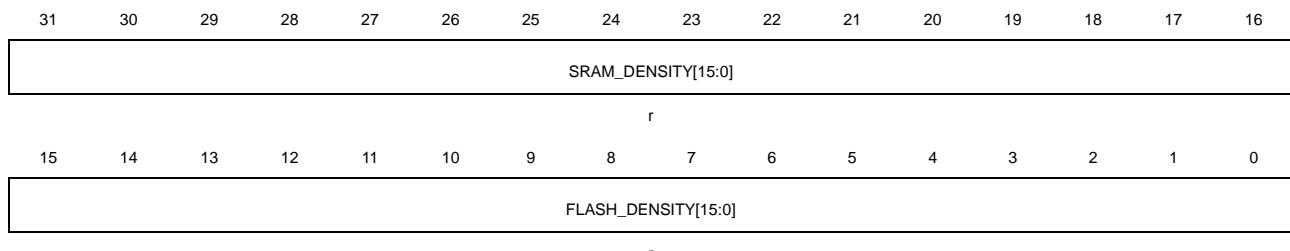
## 1.5. Device electronic signature

The device electronic signature contains memory size 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.

### 1.5.1. Memory density information

Base address: 0xFFFF F7E0

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

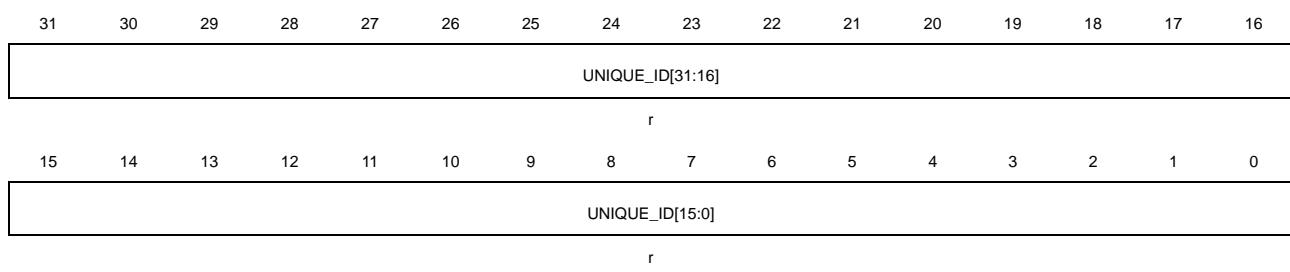


Bits	Fields	Descriptions
31:16	SRAM_DENSITY [15:0]	SRAM density The value indicates the on-chip SRAM density of the device in Kbytes. Example: 0x0008 indicates 8 Kbytes.
15:0	FLASH_DENSITY [15:0]	Flash memory density The value indicates the Flash memory density of the device in Kbytes. Example: 0x0020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

### 1.5.2. Unique device ID (96 bits)

Base address: 0xFFFF F7E8

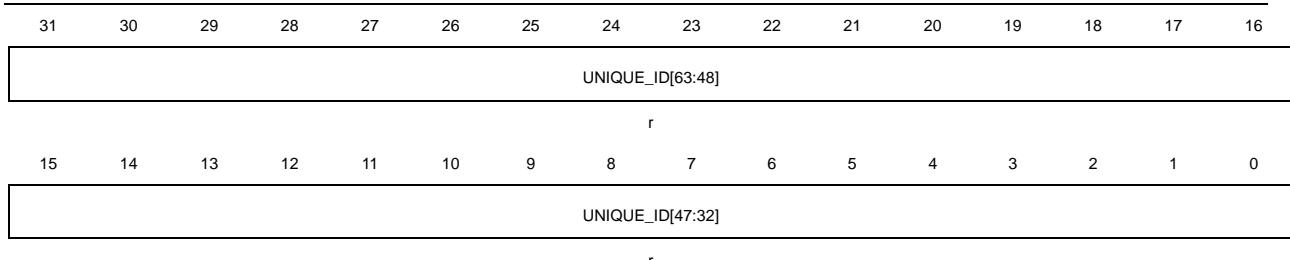
The value is factory programmed and can never be altered by user.



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

Base address: 0xFFFF F7EC

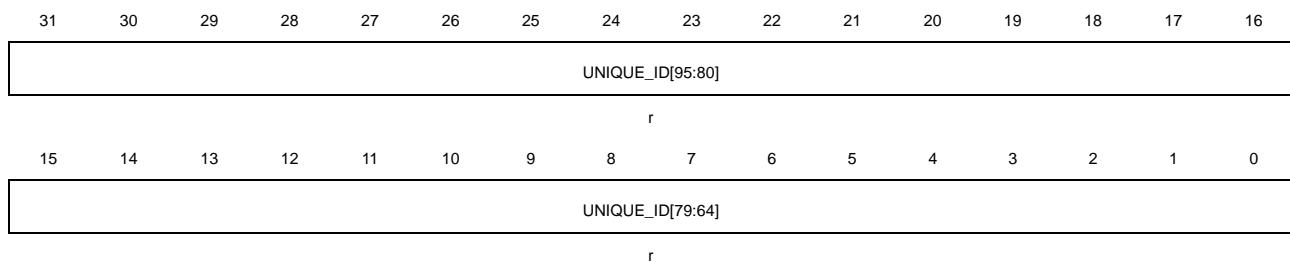
The value is factory programmed and can never be altered by user.



<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:0	UNIQUE_ID[63:32]	Unique device ID

Base address: 0x1FFF F7F0

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



<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
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 while CPU executes instructions stored in the flash. It also provides page erase, mass erase, and word/half-word program operations for flash memory.

### 2.2. Characteristics

- Up to 128KB of on-chip flash memory for instruction and data.
- No waiting time when CPU executes instructions.
- The flash page size is 1KB for all series.
- Word/half-word programming, page erase and mass erase operation.
- 16B option bytes block for user application requirements.
- Option bytes are uploaded to the option byte control registers on every system reset.
- Flash security 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 128 KB main flash organized into 128 pages with 1 KB capacity per page and a 18 KB Information Block for the Boot Loader. The main flash memory contains a total of up to 128 pages which can be erased individually. The [Table 2-1. Base address and size for flash memory](#) shows the details of flash organization.

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

Block	Name	Address Range	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 127	0x0801 FC00 - 0x0801 FFFF	1KB
Information Block	Boot loader area	0x1FFF B000- 0x1FFF F7FF	18KB
Option bytes Block	Option bytes	0x1FFF F800 - 0x1FFF F80F	16B

**Note:** The Information Block stores the boot loader. 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\_CTL0 registers

After reset, the FMC\_CTL0 registers are not accessible in write mode, and the LK bit in FMC\_CTL0 register is 1. An unlocking sequence consists of two write operations to the FMC\_KEY0 register to open the access to the FMC\_CTL0 register. The two write operations are writing 0x45670123 and 0xCDEF89AB to the FMC\_KEY0 register. After the two write operations, the LK bit in FMC\_CTL0 register is reset to 0 by hardware. The software can lock the FMC\_CTL0 again by setting the LK bit in FMC\_CTL0 register to 1. Any wrong operations to the FMC\_KEY0 will set the LK bit to 1, and lock FMC\_CTL0 register, and lead to a bus error.

The OBPG bit and OBER bit in FMC\_CTL0 are still protected even the FMC\_CTL0 is unlocked. The unlocking sequence is two write operations, which are writing 0x45670123 and 0xCDEF89AB to FMC\_OBKEY register. And then the hardware sets the OBWEN bit in FMC\_CTL0 register to 1. The software can reset OBWEN bit to 0 to protect the OBPG bit and OBER bit in FMC\_CTL0 register again.

### 2.3.4. Page erase

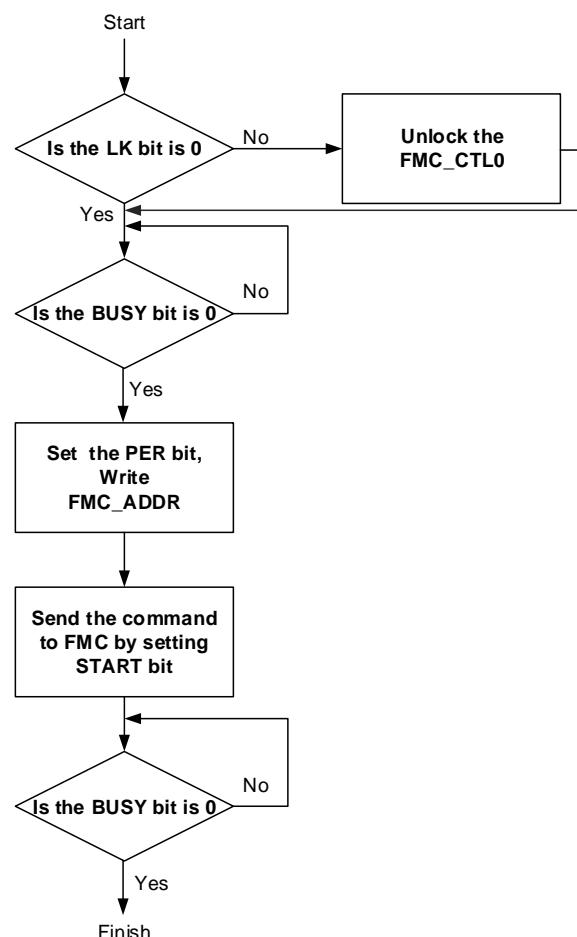
The FMC provides a page erase function which is used to initialize the contents of a main flash memory page to a high state. Each page can be erased independently without affecting the contents of other pages. The following steps show the access sequence of the registers for a page erase operation.

- Unlock the FMC\_CTL0 registers if necessary.
- Check the BUSY bit in FMC\_STATx registers to confirm that no flash memory operation is in progress (BUSY equals to 0). Otherwise, wait until the operation has finished.
- Set the PER bit in FMC\_CTL0 registers.
- Write the page absolute address (0x08XX XXXX) into the FMC\_ADDR0 registers.
- Send the page erase command to the FMC by setting the START bit in FMC\_CTL0 registers.
- Wait until all the operations have finished by checking the value of the BUSY bit in FMC\_STAT0 registers.
- Read and verify the page if required using a DBUS access.

When the operation is executed successfully, the ENDF in FMC\_STAT0 registers is set, and an interrupt will be triggered by FMC if the ENDIE bit in the FMC\_CTL0 registers 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 to fetch codes or to access data. The FMC will not provide any notification when this occurs. Additionally, the page erase operation will be ignored on erase/program protected pages. In this condition, a flash operation error interrupt will be triggered by the FMC if the ERRIE bit in the FMC\_CTL0 registers is set. The software can check the WPERR bit in the FMC\_STAT0 registers to detect this condition in the interrupt handler. [Figure 2-1. Process of page erase operation](#) 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 to initialize the main flash block contents. This erase can affect by setting MER bit to 1 in the FMC\_CTL0 register. The following steps show the mass erase register access sequence.

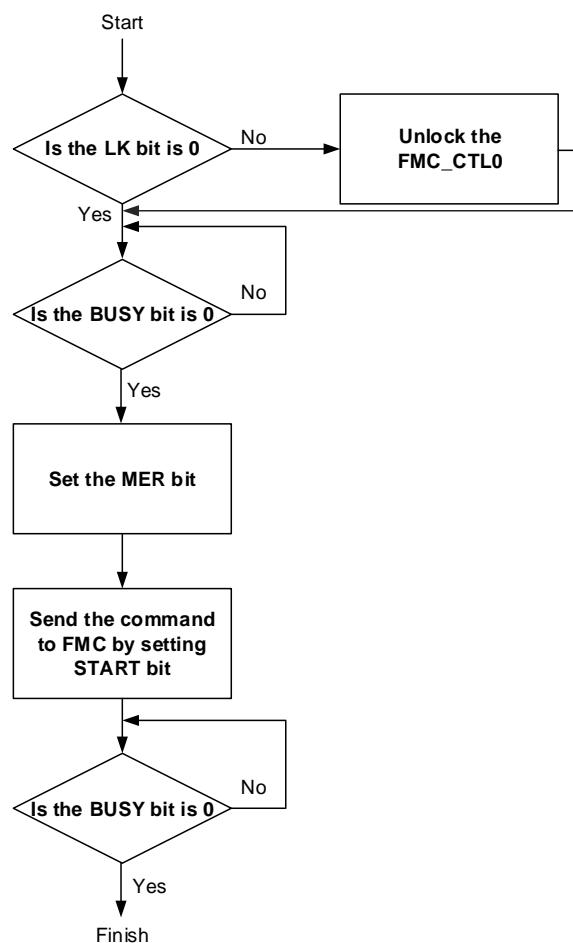
- Unlock the FMC\_CTL0 registers if necessary.
- Check the BUSY bit in FMC\_STAT0 registers to confirm that no flash memory operation is in progress (BUSY equals to 0). Otherwise, wait until the operation has finished.
- Set MER bit in FMC\_CTL0 register

- Send the mass erase command to the FMC by setting the START bit in FMC\_CTL0 registers.
- Wait until all the operations have been finished by checking the value of the BUSY bit in FMC\_STAT0 registers.
- Read and verify the flash memory if required using a DBUS access.

When the operation is executed successfully, the ENDF in FMC\_STAT0 registers is set, and an interrupt will be triggered by FMC if the ENDIE bit in the FMC\_CTL0 registers is set. Since all flash data will be modified 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 that accesses the FMC registers directly.

The following figure indicates the mass erase operation flow.

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



### 2.3.6. Main flash programming

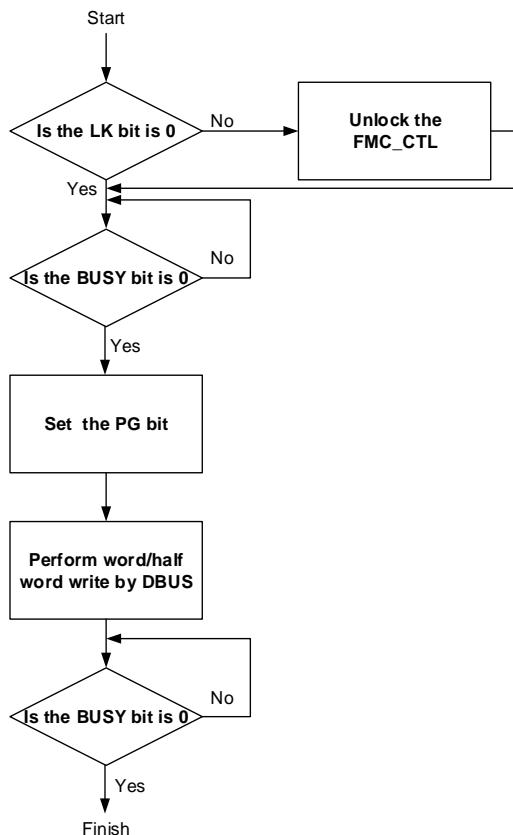
The FMC provides a 32-bit word/16-bit half word programming function which is used to modify the main flash memory contents. The following steps show the register access

sequence of the word programming operation.

- Unlock the FMC\_CTL0 registers if necessary.
- Check the BUSY bit in FMC\_STAT0 registers to confirm that no flash memory operation is in progress (BUSY equals to 0). Otherwise, wait until the operation has finished.
- Set the PG bit in FMC\_CTL0 registers.
- Write a 32-bit word/16-bit half word to desired absolute address (0x08XX XXXX) by DBUS.
- Wait until all the operations have been finished by checking the value of the BUSY bit in FMC\_STAT0 registers.
- Read and verify the Flash memory if required using a DBUS access.

When the operation is executed successfully, the ENDF in FMC\_STAT0 registers is set, and an interrupt will be triggered by FMC if the ENDIE bit in the FMC\_CTL0 registers is set. Note that the word/half word programming operation checks the address if it has been erased. If the address has not been erased, PGERR bit in the FMC\_STAT0 registers will be set when programming the address except 0x0. Note that the PG bit must be set before the word/half word programming operation. Additionally, the program operation will be ignored on erase/program protected pages and WPERR bit in FMC\_STAT0 is set. In these conditions, a flash operation error interrupt will be triggered by the FMC if the ERRIE bit in the FMC\_CTL0 registers is set. The software can check the PGERR bit or WPERR bit in the FMC\_STAT0 registers to detect which condition occurred in the interrupt handler. [\*\*Figure 2-3. Process of word program operation\*\*](#) displays the word programming operation flow.

Figure 2-3. Process of word program operation



**Note:** Reading the flash should be avoided when a program/erase operation is ongoing in the same bank. And flash memory accesses failed if the CPU enters the power saving modes.

### 2.3.7. Option bytes Erase

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

- Unlock the FMC\_CTL0 register if necessary.
- Check the BUSY bit in FMC\_STAT0 register to confirm that no Flash memory operation is in progress (BUSY equal to 0). Otherwise, wait until the operation has finished.
- Unlock the option bytes operation bits in FMC\_CTL0 register if necessary.
- Wait until OBWEN bit is set in FMC\_CTL0 register.
- Set OBER bit in FMC\_CTL0 register.
- Send the option bytes erase command to the FMC by setting the START bit in FMC\_CTL0 register.
- Wait until all the operations have been finished by checking the value of the BUSY bit in FMC\_STAT0 register.
- Read and verify the Flash memory if required using a DBUS access.

When the operation is executed successful, the ENDF in FMC\_STAT0 register is set, and an interrupt will be triggered by FMC if the ENDIE bit in the FMC\_CTL0 register is set.

### 2.3.8. Option bytes modify

The FMC provides an erase and then program function which is used to modify the option bytes block in flash. There are 8 pair option bytes. The MSB is the complement of the LSB in each pair. And when the option bytes are modified, the MSB is generated by FMC automatically, not the value of input data. The following steps show the erase sequence.

- Unlock the FMC\_CTL0 register if necessary.
- Check the BUSY bit in FMC\_STAT0 register to confirm that no Flash memory operation is in progress (BUSY equals to 0). Otherwise, wait until the operation has finished.
- Unlock the option bytes operation bits in FMC\_CTL0 register if necessary.
- Wait until OBWEN bit is set in FMC\_CTL0 register.
- Set the OBPG bit in FMC\_CTL0 register.
- A 32-bit word/16-bit half word write at desired address by DBUS.
- Wait until all the operations have been finished by checking the value of the BUSY bit in FMC\_STAT0 register.
- Read and verify the Flash memory if required using a DBUS access.

When the operation is executed successfully, the ENDF in FMC\_STAT0 register is set, and an interrupt will be triggered by FMC if the ENDIE bit in the FMC\_CTL0 register is set. Note that the word/half word programming operation checks the address if it has been erased. If the address has not been erased, PGERR bit in the FMC\_STAT0 register will set when program the address except programming 0x0.

The modified option bytes only take effect after a system reset is generated.

### 2.3.9. Option bytes description

The option bytes block is reloaded to FMC\_OBSTAT and FMC\_WP registers after each system reset, and the option bytes take effect. The complement option bytes are the opposite of option bytes. When option bytes reload, if the complement option byte and option byte do not match, the OBERR bit in FMC\_OBSTAT register is set, and the option byte is set to 0xFF.

The OBERR bit is not set if both the option byte and its complement byte are 0xFF. [Table 2-2. Option byte](#) shows the detail of option bytes.

**Table 2-2. Option byte**

Address	Name	Description
0x1fff f800	SPC	option byte Security Protection value 0xA5 : no security protection any value except 0xA5 : under security protection
0x1fff f801	SPC_N	SPC complement value
0x1fff f802	USER	[7:4]: reserved [3]: BB

Address	Name	Description
		0: boot from bank1 or bank0 if bank1 is void, when configured boot from main memory 1: boot from bank0, when configured boot from main memory [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_HW 0: hardware free watchdog 1: software free watchdog
0x1fff f803	USER_N	USER complement value
0x1fff f804	DATA[7:0]	user defined data bit 7 to 0
0x1fff f805	DATA_N[7:0]	DATA complement value bit 7 to 0
0x1fff f806	DATA[15:8]	user defined data bit 15 to 8
0x1fff f807	DATA_N[15:8]	DATA complement value bit 15 to 8
0x1fff f808	WP[7:0]	Page Erase/Program Protection bit 7 to 0 0: protection active 1: unprotected
0x1fff f809	WP_N[7:0]	WP complement value bit 7 to 0
0x1fff f80a	WP[15:8]	Page Erase/Program Protection bit 15 to 8
0x1fff f80b	WP_N[15:8]	WP complement value bit 15 to 8
0x1fff f80c	WP[23:16]	Page Erase/Program Protection bit 23 to 16
0x1fff f80d	WP_N[23:16]	WP complement value bit 23 to 16
0x1fff f80e	WP[31:24]	Page Erase/Program Protection bit 31 to 24 WP[30:24]: Each bit is related to 4KB flash protection, that means 4 pages for GD32VF103. Bit 0 configures the first 4KB flash protection, and so on.
0x1fff f80f	WP_N[31:24]	WP complement value bit 31 to 24

### 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\_STAT0 registers 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 draw the attention of the CPU. The page protection function can be individually enabled by configuring the WP [31:0] bit field to 0 in the option bytes. If a page erase operation is executed on the option bytes block, all

---

the Flash Memory page protection functions will be disabled. When WP in the option bytes is modified, a system reset followed is necessary.

### 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.

No protection: when setting SPC byte and its complement value to 0x5AA5, no protection performed. The main flash and option bytes block are accessible by all operations.

Under protection: when setting SPC byte and its complement value to any value except 0x5AA5, the security protection is performed. Note that a power reset should be followed instead of a system reset if the SPC modification is performed while the debug module is still connected to JTAG device. Under the security protection, the main flash can only be accessed by user code and the first 4KB flash is under erase/program protection. In debug mode, boot from SRAM or boot from boot loader mode, all operations to main flash is forbidden. If a read operation to main flash in debug, boot from SRAM or boot from boot loader mode, a bus error will be generated. If a program/erase operation to main flash in debug mode, boot from SRAM or boot from boot loader mode, the WPERR bit in FMC\_STAT0 registers will be set. Option bytes block are accessible by all operations, which can be used to disable the security protection. If program back to no protection level by setting SPC byte and its complement value to 0x5AA5, a mass erase for main flash will be performed.

## 2.4. Register definition

FMC base address: 0x4002 2000

### 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)

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												WSCNT[2:0]			

rw

Bits	Fields	Descriptions
31:3	Reserved	Must be kept at reset value.
2:0	WSCNT[2:0]	Wait state counter These bits is set and reset by software. The WSCNT valid when WSEN bit in FMC_WSEN 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 0(FMC\_KEY0)

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
KEY[31:16]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[15:0]															

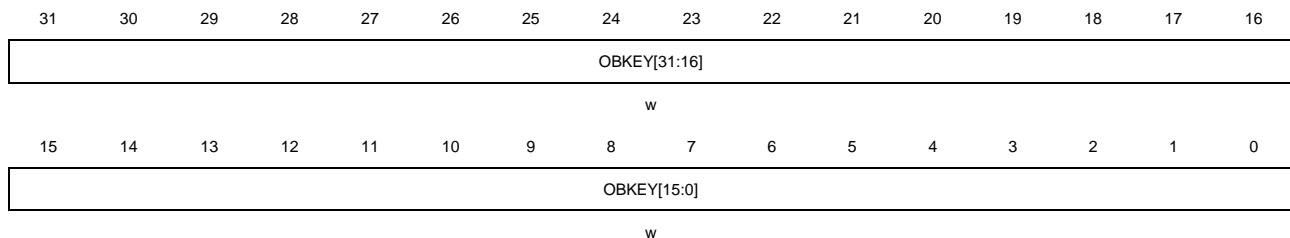
<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:0	KEY[31:0]	FMC_CTL0 unlock key These bits are only be written by software. Write KEY[31:0] with keys to unlock FMC_CTL0 register

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

Address offset: 0x08

Reset value: 0x0000 0000

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



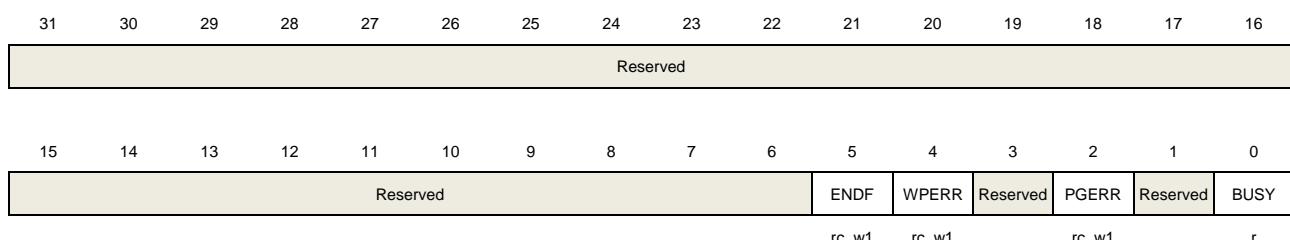
<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:0	OBKEY[31:0]	FMC_CTL0 option bytes operation unlock key These bits are only be written by software. Write OBKEY[31:0] with keys to unlock option bytes command in FMC_CTL0 register.

### 2.4.4. Status register 0 (FMC\_STAT0)

Address offset: 0x0C

Reset value: 0x0000 0000

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



<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
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 erase/program 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 program 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 is generated, this bit is cleared.

#### 2.4.5. Control register 0 (FMC\_CTL0)

Address offset: 0x10

Reset value: 0x0000 0080

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	ENDIE	Reserved	ERRIE	OBWEN	Reserved	LK	START	OBER	OBPG	Reserved	MER	PER	PG	rw	rw	rw

Bits	Fields	Descriptions
31:13	Reserved	Must be kept at reset value.
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 byte 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_CTL0 lock bit This bit is cleared by hardware when right sequence written to FMC_KEY0 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 clear by software 0: no effect 1: option byte erase command
4	OBPG	Option bytes program command bit This bit is set or clear by software 0: no effect 1: option bytes program command
3	Reserved	Must be kept at reset value.
2	MER	Main flash mass erase for bank0 command bit This bit is set or cleared by software 0: no effect 1: main flash mass erase command for bank0
1	PER	Main flash page erase for bank0 command bit This bit is set or clear by software 0: no effect 1: main flash page erase command for bank0
0	PG	Main flash program for bank0 command bit This bit is set or clear by software 0: no effect 1: main flash program command for bank0

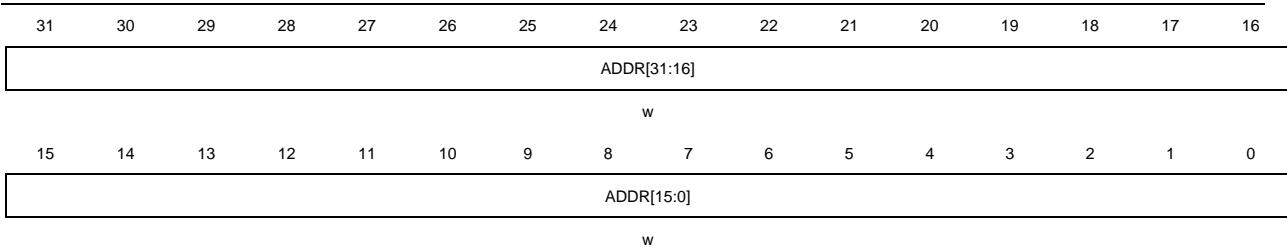
**Note:** This register should be reset after the corresponding flash operation completed.

#### 2.4.6. Address register 0 (FMC\_ADDR0)

Address offset: 0x14

Reset value: 0x0000 0000

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



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

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

Address offset: 0x1C

Reset value: 0x0XXXX XXXX.

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



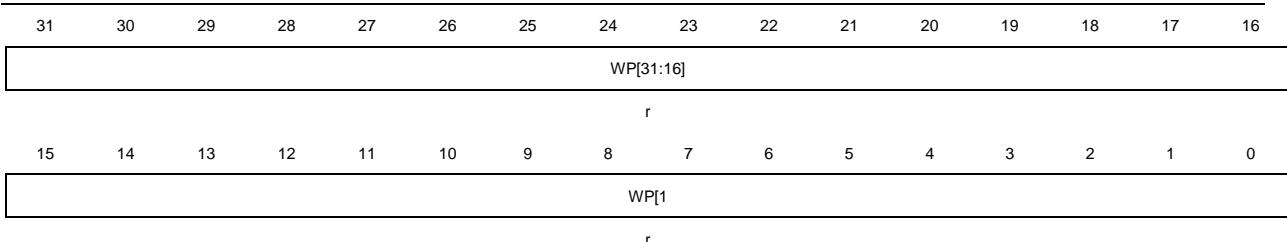
Bits	Fields	Descriptions
31:26	Reserved	Must be kept at reset value.
25:10	DATA[15:0]	Store DATA of option bytes block after system reset.
9:2	USER[7:0]	Store USER of option bytes block after system reset.
1	SPC	Option bytes security protection code 0: no protection 1: protection
0	OBERR	Option bytes read error bit. This bit is set by hardware when the option bytes and its complement byte do not match, then the option bytes is set to 0xFF.

#### 2.4.8. Erase/Program Protection register (FMC\_WP)

Address offset: 0x20

Reset value: 0xXXXX XXXX

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



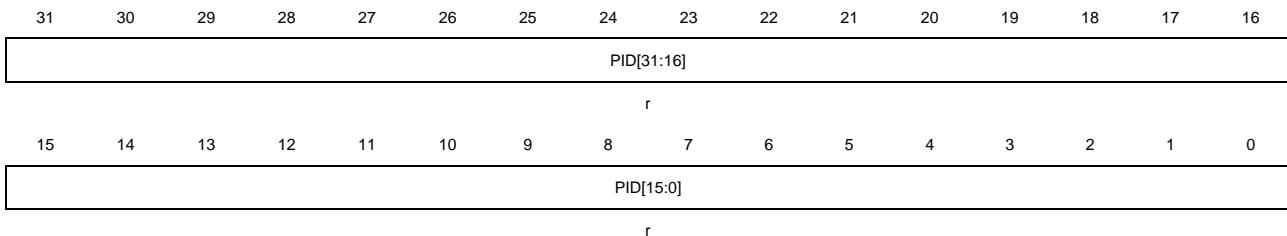
<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:0	WP[31:0]	Store WP of option bytes block after system reset

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

Address offset: 0x100

Reset value: 0xFFFF XXXX

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



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

### 3. Power management unit (PMU)

#### 3.1. Overview

The power consumption is regarded as one of the most important issues for the devices of GD32VF103 series. According to 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 GD32VF103 devices, there are three power domains, including  $V_{DD}/V_{DDA}$  domain, 1.2V domain, and Backup domain, as is shown in [Figure 3-1. Power supply overview](#). 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 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.

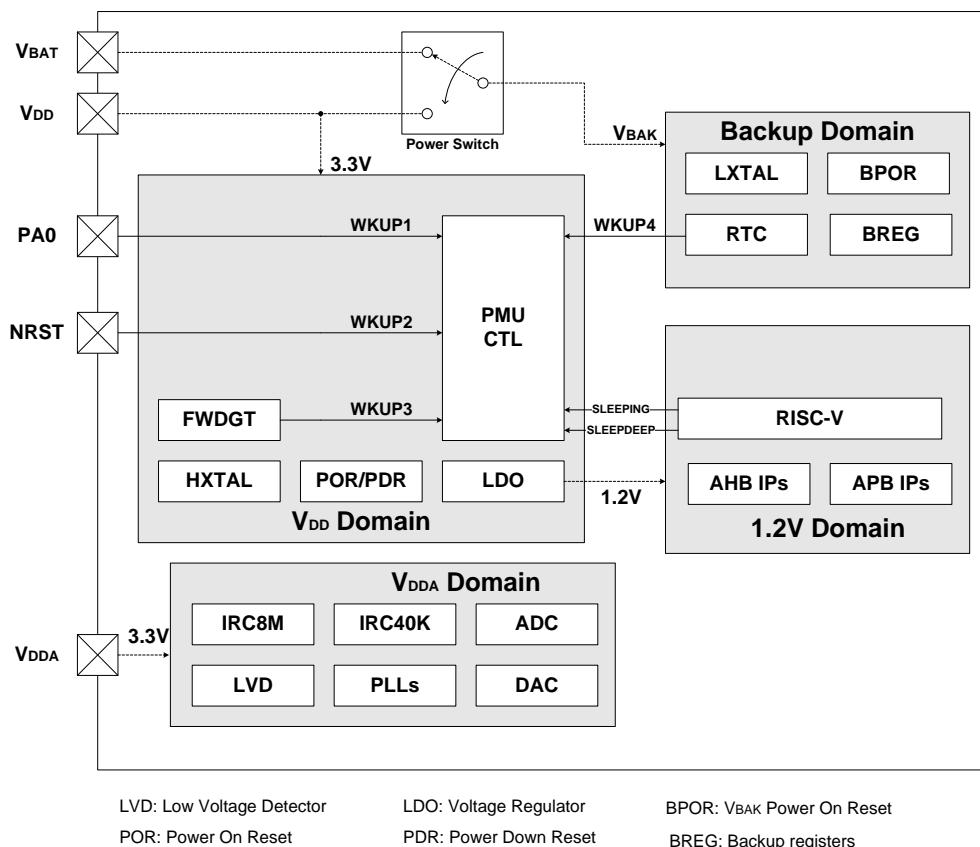
#### 3.2. Characteristics

- Three power domains:  $V_{BAK}$ ,  $V_{DD}/V_{DDA}$  and 1.2V power domains.
- Three power saving modes: Sleep, Deep-sleep and Standby modes.
- Internal Voltage regulator (LDO) supplies around 1.2V voltage source for 1.2V domain.
- 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.
- LDO output voltage select for power saving.

#### 3.3. Function overview

[Figure 3-1. Power supply overview](#) provides details on the internal configuration of the PMU and the relevant power domains.

Figure 3-1. Power supply overview



### 3.3.1. Battery backup domain

The Backup domain is powered by the V<sub>DD</sub> or the battery power source (V<sub>BAT</sub>) selected by the internal power switch, and the V<sub>BAK</sub> pin which drives Backup Domain, supplies power for RTC unit, LXTAL oscillator, BPOR and BREG, and three pads, including PC13 to PC15. In order to ensure the content of the Backup domain registers and the RTC supply, when V<sub>DD</sub> supply is shut down, V<sub>BAT</sub> 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<sub>DD</sub>/V<sub>DDA</sub> domain. If no external battery is used in the application, it is recommended to connect V<sub>BAT</sub> pin externally to V<sub>DD</sub> 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<sub>BAK</sub> 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 the Real Time Clock (RTC) circuit can be derived from the Internal 40KHz RC oscillator (IRC40K) or the Low Speed Crystal oscillator (LXTAL), or HXTAL clock divided by 128. When V<sub>DD</sub> is shut down, only LXTAL is valid for RTC. Before entering the power saving mode by executing the WFI/WFE instruction, the RISC-V can setup the RTC register with an expected wakeup time and enable the wakeup function to achieve the RTC timer wakeup

event. After entering the power saving mode for a certain amount of time, the RTC will wake up the device when the time match event occurs. The details of the RTC configuration and operation will be described in the [Real-time Clock \(RTC\)](#).

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 [Real-time Clock \(RTC\)](#)..
- 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 [Real-time Clock \(RTC\)](#)..
- PC14 and PC15 can be used as LXTAL Crystal oscillator pins only.

**Note:** Since PC13, PC14, 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).

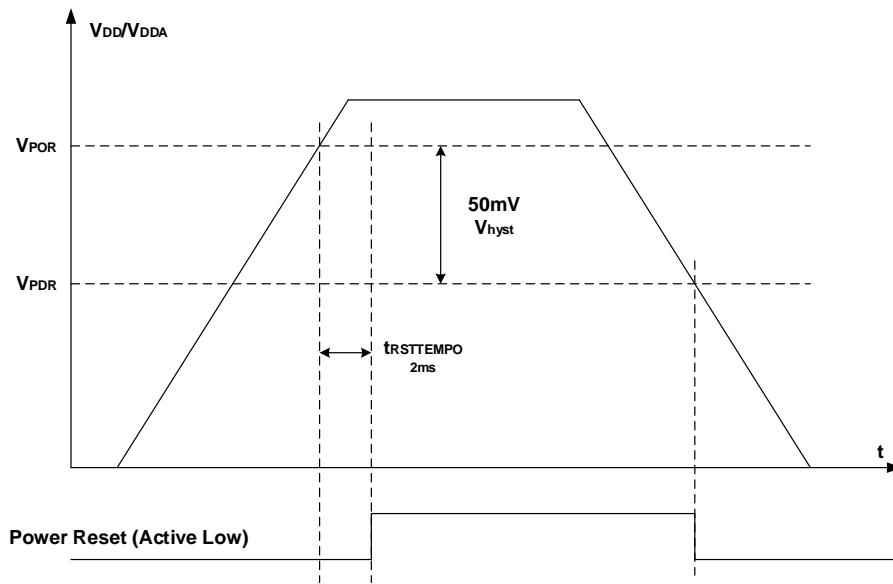
### 3.3.2. $V_{DD}/V_{DDA}$ power domain

$V_{DD}/V_{DDA}$  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), IRC40K (Internal 40KHz RC oscillator), PLLs (Phase Locking Loop), LVD (Low Voltage Detector), etc.

#### $V_{DD}$ domain

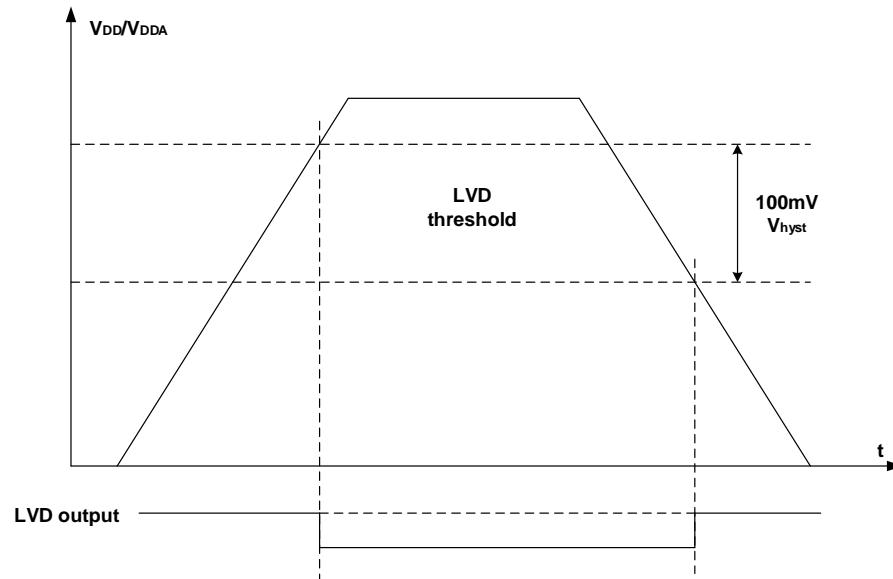
The LDO, which is implemented to supply power for the 1.2V domain, is always enabled after reset. It can be configured to operate in three different status, including in the Sleep mode (full power on), in the Deep-sleep mode (on or low power), and in the Standby mode (power off).

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 3-2. Waveform of the POR/PDR](#) shows the relationship between the supply voltage and the power reset signal.  $V_{POR}$ , which typical value is 2.40V, indicates the threshold of power on reset, while  $V_{PDR}$ , which typical value is 2.35V, means the threshold of power down reset. The hysteresis voltage ( $V_{hyst}$ ) is around 50mV.

**Figure 3-2. Waveform of the POR/PDR**


### **V<sub>DDA</sub> 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 3-3. Waveform of the 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 3-3. Waveform of the 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.

To ensure a high accuracy on low voltage ADC and DAC, the separate external reference voltage on  $V_{REF}$  should be connected to ADC/DAC pins. According to the different packages,  $V_{REF+}$  pin must be connected to  $V_{DDA}$  pin,  $V_{REF-}$  pin must be connected to  $V_{SSA}$  pin. The  $V_{REF+}$  pin is only available on no less than 100-pin packages, or else the  $V_{REF+}$  pin is not available and internally connected to  $V_{DDA}$ . The  $V_{REF-}$  pin is only available on no less than 100-pin packages, or else the  $V_{REF-}$  pin is not available and internally connected to  $V_{SSA}$ .

### 3.3.3. 1.2V power domain

The main functions that include RISC-V logic, AHB/APB peripherals, the APB interfaces for the Backup domain and the  $V_{DD}/V_{DDA}$  domain, etc, are located in this power domain. Once the 1.2V is powered up, the POR will generate a reset sequence on the 1.2V power domain. If need 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. Power saving modes

After a system reset or a power reset, the GD32VF103 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 RISC-V. In Sleep mode, only clock of RISC-V is off. To enter the Sleep mode, it is only necessary to clear the CSR\_SLEEPVALUE bit in the RISC-V 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.

### Deep-sleep mode

The Deep-sleep mode is based on the SLEEPDEEP mode of the RISC-V. In Deep-sleep mode, all clocks in the 1.2V domain are off, and all of IRC8M, 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 CSR\_SLEEPVALUE bit in the RISC-V 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/time stamp/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 RISC-V, too. In Standby mode, the whole 1.2V domain is power off, the LDO is shut down, and all of IRC8M, HXTAL and PLLs are disabled. Before entering the Standby mode, it is necessary to set the CSR\_SLEEPVALUE bit in the RISC-V System Control Register, and set the STBMOD bit in the PMU\_CTL register, and clear WUF bit in the PMU\_CS register. Then, the device enters the Standby mode after a WFI or WFE instruction is executed, and the STBF status flag in the PMU\_CS register indicates that the MCU has been in Standby mode. There are four 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 WKUP pin. The Standby mode achieves the lowest power consumption, but spends longest time to wake up. Besides, the contents of SRAM and registers in 1.2V power domain are lost in Standby mode. When exiting from the Standby mode, a power-on reset occurs and the RISC-V will execute instruction code from the 0x00000000 address.

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

Mode	Sleep	Deep-sleep	Standby
Description	Only CPU clock is off	All clocks in the 1.2V domain are off Disable IRC8M, HXTAL and PLL	The 1.2V domain is power off Disable IRC8M, HXTAL and PLL
LDO Status	On	On or in low power mode	Off
Configuration	CSR_SLEEPVALUE = 0	CSR_SLEEPVALUE = 1 STBMOD = 0	CSR_SLEEPVALUE = 1 STBMOD = 1, WURST=1
Entry	WFI or WFE	WFI or WFE	WFI or WFE

<b>Mode</b>	<b>Sleep</b>	<b>Deep-sleep</b>	<b>Standby</b>
Wakeup	Any interrupt for WFI Any event (or interrupt) for WFE(WFI)	Any interrupt from EXTI lines for WFI Any event(or interrupt) from EXTI for WFE(WFI)	NRST pin WKUP pin FWDGT reset RTC
Wakeup Latency	None	IRC8M wakeup time, LDO wakeup time added if LDO is in low power mode	Power on sequence

## 3.4. Register definition

PMU base address: 0x4000 7000

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

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)

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						BKPWEN	LVDT[2:0]		LVDEN	STBRST	WURST	STBMOD	LDOLP		
						rw	rw		rw	rc_w1	rc_w1	rw	rw	rw	

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
3	STBRST	Standby Flag Reset 0: No effect 1: Reset the standby flag This bit is always read as 0.

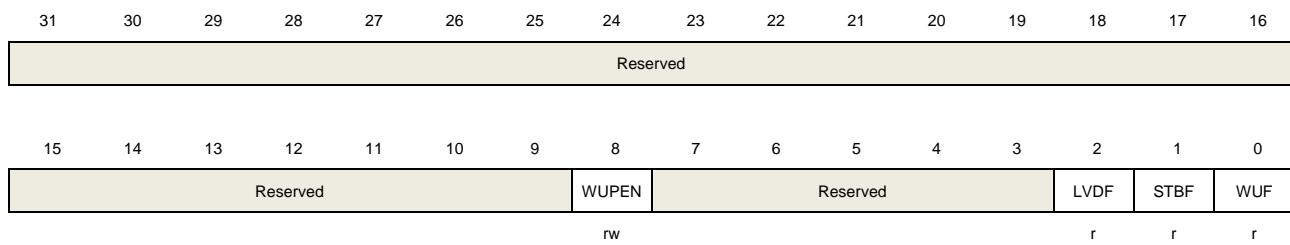
2	WURST	Wakeup Flag Reset 0: No effect 1: Reset the wakeup flag This bit is always read as 0.
1	STBMOD	Standby Mode 0: Enter the Deep-sleep mode when the RISC-V enters SLEEPDEEP mode 1: Enter the Standby mode when the RISC-V enters SLEEPDEEP mode
0	LDOLP	LDO Low Power Mode 0: The LDO operates normally during the Deep-sleep mode 1: The LDO is in low power mode during the Deep-sleep mode

### 3.4.2. Control and 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
31:9	Reserved	Must be kept at reset value.
8	WUPEN	WKUP Pin Enable 0: Disable WKUP pin function 1: Enable WKUP pin function If WUPEN is set before entering the power saving mode, a rising edge on the WKUP pin wakes up the system from the power saving mode. As the WKUP pin is active high, the WKUP pin is internally configured to input pull down mode. And set this bit will trigger a wakeup event when the input changes to high.
7:3	Reserved	Must be kept at reset value.
2	LVDF	Low Voltage Detector Status Flag 0: Low Voltage event has not occurred ( $V_{DD}$ is higher than the specified LVD threshold) 1: Low Voltage event occurred ( $V_{DD}$ is equal to or lower than the specified LVD threshold)

**Note:** The LVD function is stopped in Standby mode.

---

1	STBF	Standby Flag 0: The device has not entered the Standby mode 1: The device has been in the Standby mode This bit is cleared only by a POR/PDR or by setting the STBRST bit in the PMU_CTL register.
0	WUF	Wakeup Flag 0: No wakeup event has been received 1: Wakeup event occurred from the WKUP pin or the RTC wakeup event including RTC Tamper event, RTC alarm event, or RTC Time Stamp event. This bit is cleared only by a POR/PDR or by setting the WURST bit in the PMU_CTL register.

## 4. Backup registers (BKP)

### 4.1. Overview

The Backup registers are located in the Backup domain that remains powered-on by  $V_{BAT}$  even if  $V_{DD}$  power is shut down, they are forty-two 16-bit (84 bytes) registers for data protection of user application data, and the wake-up action from Standby mode or system reset do not affect these registers.

In addition, the BKP registers can be used to implement the tamper detection and RTC calibration function.

After reset, any writing access to the registers in Backup domain is disabled, that is, the Backup registers and RTC cannot be written to access. In order to enable access to the Backup registers and RTC, the Power and Backup interface clocks should be enabled firstly by setting the PMUEN and BKPIEN bits in the RCU\_APB1EN register, and writing access to the registers in Backup domain should be enabled by setting the BKPWEN bit in the PMU\_CTL register.

### 4.2. Characteristics

- 84 bytes Backup registers which can keep data under power saving mode. If tamper event is detected, Backup registers will be reset.
- The active level of Tamper source (PC13) can be configured.
- RTC Clock Calibration register provides RTC alarm and second output selection, and the calibration value configuration.
- Tamper control and status register (BKP\_TPCS) can control tamper detection with interrupt or event capability.

### 4.3. Function overview

#### 4.3.1. RTC clock calibration

In order to improve the RTC clock accuracy, the MCU provides the RTC output for calibration function. The clock with the frequency  $f_{RTCCLK}/64$  can be output on the PC13. It is enabled by setting the COEN bit in the BKP\_OCTL register.

The calibration value is set by RCCV[6:0] in the BKP\_OCTL register, and the calibration function can slow down the RTC clock by steps of  $1000000/2^{20}$  ppm.

#### 4.3.2. Tamper detection

In order to protect the important user data, the MCU provides the tamper detection function, and it can be independently enabled on TAMPER pin by setting corresponding TPEN bit in the BKP\_TPCTL register. To prevent the tamper event from losing, the edge detection is logically ANDed with the TPEN bit, used for tamper detection signal. So the tamper detection configuration should be set before enable TAMPER pin. When the tamper event is detected, the corresponding TEF bit in the BKP\_TPCS register will be set. Tamper event can generate an interrupt if tamper interrupt is enabled. Any tamper event will reset all Backup data registers.

**Note:** When TPAL=0/1, if the TAMPER pin is already high/low before it is enabled(by setting TPEN bit), an extra tamper event is detected, while there was no rising/falling edge on the TAMPER pin after TPEN bit was set.

## 4.4. Register definition

BKP base address: 0x4000 6C00

### 4.4.1. Backup data register x (BKP\_DATAx) (x= 0..41)

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

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
DATA [15:0]															
rw															

Bits	Fields	Descriptions
15:0	DATA[15:0]	Backup data These bits are used for general purpose data storage. The contents of the BKP_DATAx register will remain even if the wake-up action from Standby mode or system reset or power reset occurs.

### 4.4.2. RTC signal output control register (BKP\_OCTL)

Address offset: 0x2C

Reset value: 0x0000(bit [6:0], bit 8, bit 9 reset by a Backup domain reset, bit 7 reset by a POR/PDR)

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					ROSEL	ASOEN	COEN	RCCV[6:0]							
rw															

Bits	Fields	Descriptions
15:10	Reserved	Must be kept at reset value.
9	ROSEL	RTC output selection 0: RTC alarm pulse is selected as the RTC output 1: RTC second pulse is selected as the RTC output
8	ASOEN	RTC alarm or second signal output enable 0: Disable RTC alarm or second output 1: Enable RTC alarm or second output When enable, the TAMPER pin will output the RTC output.
7	COEN	RTC clock calibration output enable 0: Disable RTC clock calibration output

1: Enable RTC clock calibration output

When enable, the TAMPER pin will output a clock with the frequency  $f_{RTCCLK}/64$ .

ASOEN has the priority over COEN. When ASOEN is set, the TAMPER pin will output the RTC alarm or second signal whether COEN is set or not.

6:0	RCCV[6:0]	RTC clock calibration value The value indicates how many clock pulses are ignored or added every $2^{20}$ RTC clock pulses.
-----	-----------	--

#### 4.4.3. Tamper pin control register (BKP\_TPCTL)

Address offset: 0x30

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														TPAL	TPEN
rw														rw	rw

Bits	Fields	Descriptions
15:2	Reserved	Must be kept at reset value.
1	TPAL	TAMPER pin active level 0: The TAMPER pin is active high 1: The TAMPER pin is active low
0	TPEN	TAMPER detection enable 0: The TAMPER pin is free for GPIO functions 1: The TAMPER pin is dedicated for the Backup Reset function. The active level on the TAMPER pin resets all data of the BKP_DATAx registers.

#### 4.4.4. Tamper control and status register (BKP\_TPCS)

Address offset: 0x34

Reset value: 0x0000(bit 2 reset by a system reset or the wake-up from Standby mode)

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							TIF	TEF	Reserved					TPIE	TIR	TER
r							r	r						rw	w	w

Bits	Fields	Descriptions
15:10	Reserved	Must be kept at reset value.

9	TIF	Tamper interrupt flag 0: No tamper interrupt occurred 1: A tamper interrupt occurred This bit is reset by writing 1 to the TIR bit or the TPIE bit being 0.
8	TEF	Tamper event flag 0: No tamper event occurred 1: A tamper event occurred This bit is reset by writing 1 to the TER bit.
7:3	Reserved	Must be kept at reset value.
2	TPIE	Tamper interrupt enable 0: Disable the tamper interrupt 1: Enable the tamper interrupt
1	TIR	Tamper interrupt reset 0: No effect 1: Reset the TIF bit This bit is always read as 0.
0	TER	Tamper event reset 0: No effect 1: Reset the TEF bit This bit is always read as 0.

## 5. Reset and clock unit (RCU)

### 5.1. Reset control unit (RCTL)

#### 5.1.1. Overview

GD32VF103 Reset Control includes the control of three kinds of reset: power reset, system reset and backup domain reset. The power reset, known as a cold reset, resets the full system except the Backup domain. The system reset resets the processor core and peripheral IP components except for the JTAG controller and the Backup domain. The 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.

#### 5.1.2. 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 whose active signal is low, it will be de-asserted when the internal LDO voltage regulator is ready to provide 1.2V power. The RESET service routine vector is fixed at address 0x0000 0000 in the memory map.

##### System reset

A system reset is generated by the following events:

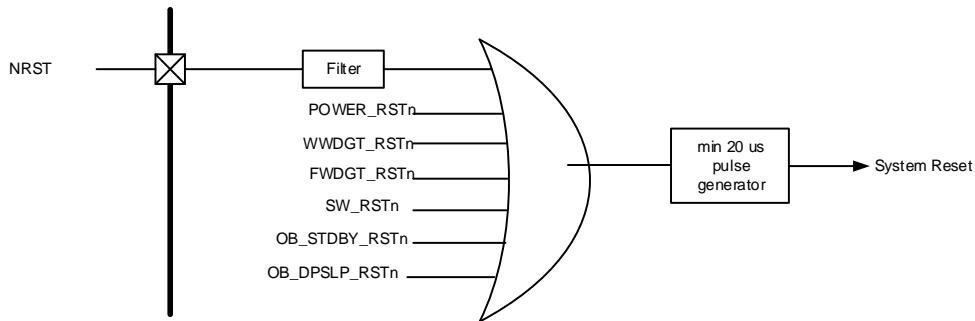
- A power reset (POWER\_RSTn).
- An external pin reset (NRST).
- A window watchdog timer reset (WWDGT\_RSTn).
- A free watchdog timer reset (FWDGT\_RSTn).
- The SYSRESETREQ bit in RISC-V Application Interrupt and Reset Control Register is set (SW\_RSTn).
- Reset generated when entering Standby mode when resetting nRST\_STDBY bit in User Option Bytes (OB\_STDBY\_RSTn).
- Reset generated when entering Deep-sleep mode when resetting nRST\_DPSLP bit in User Option Bytes (OB\_DPSLP\_RSTn).

A system reset resets the processor core and peripheral IP components except for the JTAG 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 5-1. The system reset circuit**



### **Backup domain reset**

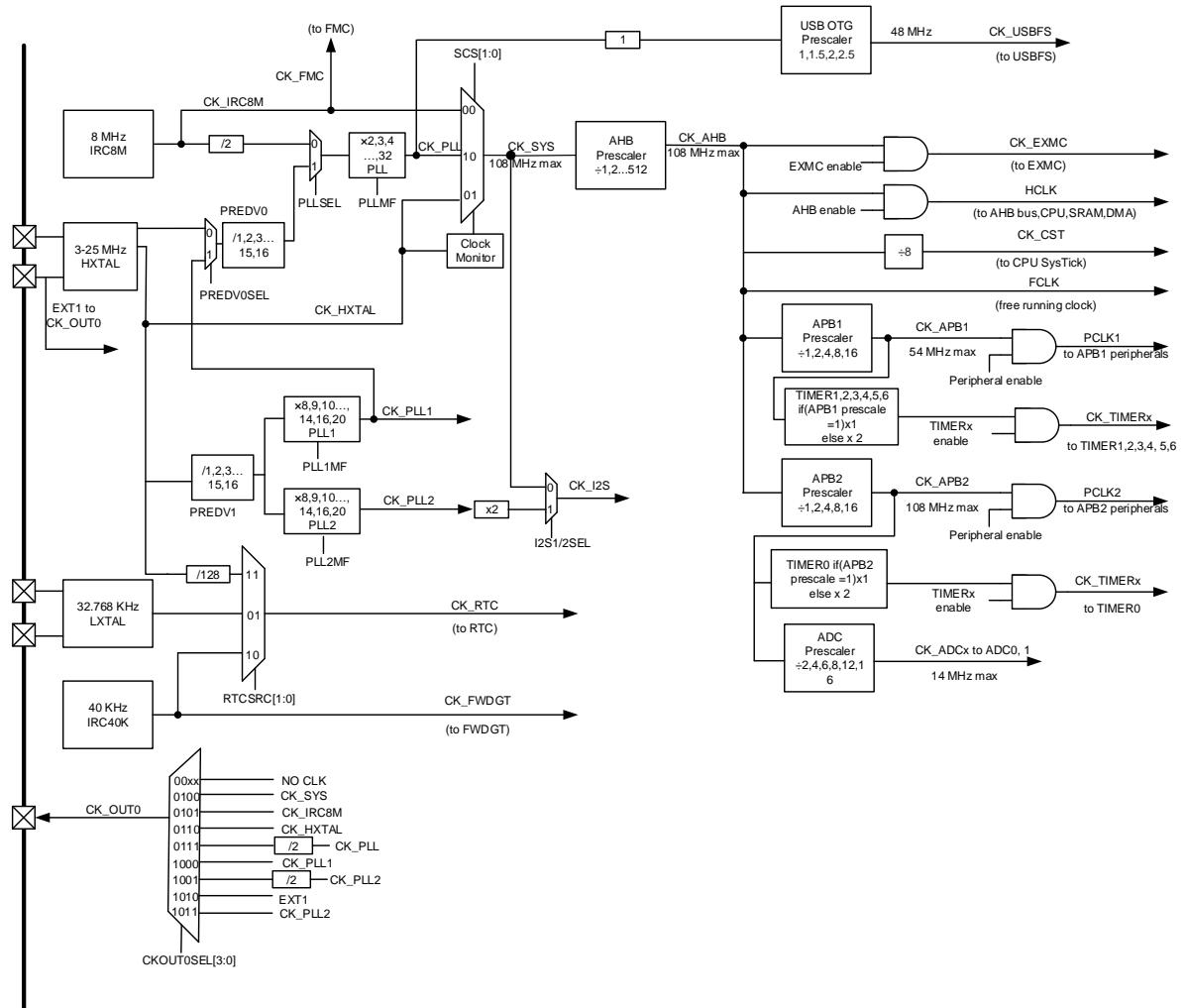
A backup domain reset is generated by setting the BKPRST bit in the Backup domain control register or Backup domain power on reset ( $V_{DD}$  or  $V_{BAT}$  power on, if both supplies have previously been powered off).

## **5.2. Clock control unit (CCTL)**

### **5.2.1. Overview**

The Clock Control unit provides a range of frequencies and clock functions. These include an Internal 8M RC oscillator (IRC8M), a High Speed crystal oscillator (HXTAL), a Low Speed Internal 40K RC oscillator (IRC40K), a Low Speed crystal oscillator (LXTAL), three Phase Lock Loop (PLL), a HXTAL clock monitor, clock prescalers, clock multiplexers and clock gating circuitry.

The clocks of the AHB, APB and RISC-V 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 108 MHz. The Free Watchdog Timer has independent clock source (IRC40K), and Real Time Clock (RTC) uses the IRC40K, LXTAL or HXTAL/128 as its clock source.

**Figure 5-2. Clock tree**


The frequency of AHB, APB2 and the APB1 domains can be configured by each prescaler. The maximum frequency of the AHB, APB2 and APB1 domains is 108 MHz/108 MHz/54 MHz. The RISCV 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 AHB clock (HCLK), configurable in the SysTick Control and Status Register.

The ADCs are clocked by the clock of APB2 divided by 2, 4, 6, 8, 12, 16.

The TIMERS are clocked by the clock divided from CK\_APB2 and CK\_APB1. The frequency of TIMERS clock is equal to CK\_APBx(APB prescaler is 1), twice the CK\_APBx(APB

prescaler is not 1).

The USBFS is clocked by the clock of CK\_PLL as the clock source of 48MHz.

The I2S is clocked by the clock of CK\_SYS or PLL2\*2 which defined by I2SxSEL bit in RCU\_CFG1 register.

The RTC is clocked by LXTAL clock or IRC40K clock or HXTAL clock divided by 128 (defined which select by RTCSRC bit in Backup Domain Control Register (RCU\_BDCTL)). After the RTC select HXTAL clock divided by 128, the clock disappeared when the 1.2V core domain power off. After the RTC select IRC40K, the clock disappeared when V<sub>DD</sub> power off. After the RTC select LXTAL, the clock disappeared when V<sub>DD</sub> and V<sub>BAT</sub> power off.

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

The FMC is clocked by IRC8M clock, which is forced on when IRC8M started.

### 5.2.2. Characteristics

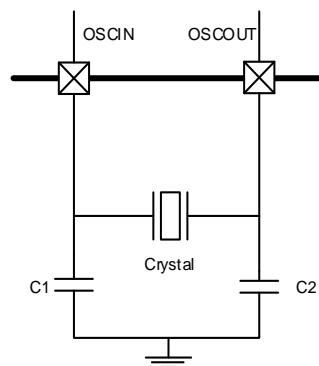
- 3 to 25 MHz High Speed crystal oscillator (HXTAL) .
- Internal 8 MHz RC oscillator (IRC8M).
- 32,768 Hz Low Speed crystal oscillator (LXTAL).
- Internal 40KHz RC oscillator (IRC40K).
- PLL clock source can be HXTAL or IRC8M.
- HXTAL clock monitor.

### 5.2.3. Function overview

#### High speed crystal oscillator (HXTAL)

The high speed external crystal oscillator (HXTAL), which has a frequency from 3 to 25 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 5-3. HXTAL clock source**



The HXTAL crystal oscillator can be switched on or off using the HXTALEN bit in the Control

Register RCU\_CTL. The HXTALSTB flag in Control Register RCU\_CTL 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 RCU\_CTL. The CK\_HXTAL is equal to the external clock which drives the OSCIN pin.

### Internal 8M RC oscillators (IRC8M)

The internal 8M 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 RCU\_CTL. The IRC8MSTB flag in the Control Register RCU\_CTL is used to indicate if the internal 8M 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 Clock Interrupt Register, RCU\_INT, is set when the IRC8M becomes stable. The IRC8M clock can also be used as the system clock source or 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)

There are three internal Phase Locked Loop, including PLL, PLL1 and PLL2.

The PLL can be switched on or off by using the PLLEN bit in the RCU\_CTL Register. The PLLSTB flag in the RCU\_CTL Register will indicate if the PLL clock is stable. An interrupt can be generated if the related interrupt enable bit, PLLSTBIE, in the RCU\_INT Register, is set as the PLL becomes stable.

The PLL1 can be switched on or off by using the PLL1EN bit in the RCU\_CTL Register. The PLL1STB flag in the RCU\_CTL Register will indicate if the PLL1 clock is stable. An interrupt can be generated if the related interrupt enable bit, PLL1STBIE, in the RCU\_INT Register, is set as the PLL1 becomes stable.

The PLL2 can be switched on or off by using the PLL2EN bit in the RCU\_CTL Register. The PLL2STB flag in the RCU\_CTL Register will indicate if the PLL2 clock is stable. An interrupt

can be generated if the related interrupt enable bit, PLL2STBIE, in the RCU\_INT Register, is set as the PLL2 becomes stable.

The three PLLs are closed by hardware when entering the Deepsleep/Standy mode or HXTAL monitor fail when HXTAL used as the source clock of the PLLs.

### Low speed crystal oscillator (LXTAL)

The low speed external 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 40K RC oscillator (IRC40K)

The internal 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 Clock Interrupt Register (RCU\_INT) is set when the IRC40K becomes stable.

The IRC40K can be trimmed by TIMER4\_CH3, user can get the clocks frequency, and adjust the RTC and FWDGT counter. Please refer to TIMER4CH3\_IREMAP in AFIO\_PCF0 register.

### System clock (CK\_SYS) selection

After the system reset, the default CK\_SYS source will be IRC8M and can be switched to HXTAL or CK\_PLL by changing the System Clock Switch bits, SCS, in the Clock 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 directly or indirectly (by PLL) used as the CK\_SYS, 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 (RCU\_CTL). 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 interrupt Flag, CKMIF, in the Clock 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 RISC-V. If the HXTAL is selected as the clock source of CK\_SYS, PLL and CK\_RTC, the HXTAL failure will force the CK\_SYS source to IRC8M, the PLL will be disabled automatically. If the HXTAL is selected as the clock source of PLL, the HXTAL failure will force the PLL closed automatically. If the HXTAL is selected as the clock source of RTC, the HXTAL failure will reset the RTC clock selection.

### Clock output capability

The clock output capability is ranging from 0.09375 MHz to 108 MHz. There are several clock signals can be selected via the CK\_OUT0 Clock Source Selection bits, CKOUT0SEL, in the Clock 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 5-1. Clock output 0 source select**

Clock Source 0 Selection bits	Clock Source
00xx	NO CLK
0100	CK_SYS
0101	CK_IRC8M
0110	CK_HXTAL
0111	CK_PLL/2
1000	CK_PLL1
1001	CK_PLL2/2
1010	EXT1
1011	CK_PLL2

### Voltage control

The 1.2V domain voltage in Deep-sleep mode can be controlled by DSLPVS[1:0] bit in the Deep-sleep mode voltage register (RCU\_DSV).

**Table 5-2. 1.2V domain voltage selected in deep-sleep mode**

DSLPVS[1:0]	Deep-sleep mode voltage(V)
00	1.2
01	1.1
10	1.0
11	0.9

## 5.3. Register definition

RCU base address: 0x4002 1000

### 5.3.1. Control register (RCU\_CTL)

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		PLL2STB	PLL2EN	PLL1STB	PLL1EN	PLLSTB	PLL EN		Reserved		CKMEN	HXTALB PS	HXTALST B	HXTALE N	
	r	rw	r	rw	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]		Reserved	IRC8MST B	IRC8MEN		
				r					rw			r	rw		

Bits	Fields	Descriptions
31:30	Reserved	Must be kept at reset value.
29	PLL2STB	PLL2 Clock Stabilization Flag  Set by hardware to indicate if the PLL2 output clock is stable and ready for use. 0: PLL2 is not stable 1: PLL2 is stable
28	PLL2EN	PLL2 enable  Set and reset by software. Reset by hardware when entering Deep-sleep or Standby mode. 0: PLL2 is switched off 1: PLL2 is switched on
27	PLL1STB	PLL1 Clock Stabilization Flag  Set by hardware to indicate if the PLL1 output clock is stable and ready for use. 0: PLL1 is not stable 1: PLL1 is stable
26	PLL1EN	PLL1 enable  Set and reset by software. Reset by hardware when entering Deep-sleep or Standby mode. 0: PLL1 is switched off 1: PLL1 is switched on
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	<p>PLL enable</p> <p>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.</p> <p>0: PLL is switched off 1: PLL is switched on</p>
23:20	Reserved	Must be kept at reset value.
19	CKMEN	<p>HXTAL Clock Monitor Enable</p> <p>0: Disable the High speed 3 ~ 25 MHz crystal oscillator (HXTAL) clock monitor 1: Enable the High speed 3 ~ 25 MHz crystal oscillator (HXTAL) clock monitor</p> <p>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.</p> <p><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.</p>
18	HXTALBPS	<p>High speed crystal oscillator (HXTAL) clock bypass mode enable</p> <p>The HXTALBPS bit can be written only if the HXTALEN is 0.</p> <p>0: Disable the HXTAL Bypass mode 1: Enable the HXTAL Bypass mode in which the HXTAL output clock is equal to the input clock.</p>
17	HXTALSTB	<p>High speed crystal oscillator (HXTAL) clock stabilization flag</p> <p>Set by hardware to indicate if the HXTAL oscillator is stable and ready for use.</p> <p>0: HXTAL oscillator is not stable 1: HXTAL oscillator is stable</p>
16	HXTALEN	<p>High Speed crystal oscillator (HXTAL) Enable</p> <p>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 when PLL clock is selected to the system clock. Reset by hardware when entering Deep-sleep or Standby mode.</p> <p>0: High speed 3 ~ 25 MHz crystal oscillator disabled 1: High speed 3 ~ 25 MHz crystal oscillator enabled</p>
15:8	IRC8MCALIB[7:0]	<p>Internal 8MHz RC Oscillator calibration value register</p> <p>These bits are load automatically at power on.</p>
7:3	IRC8MADJ[4:0]	<p>Internal 8MHz RC Oscillator clock trim adjust value</p> <p>These bits are set by software. The trimming value is these bits (IRC8MADJ) added to the IRC8MCALIB[7:0] bits. The trimming value should trim the IRC8M to 8 MHz</p>

$\pm 1\%$ .

2	Reserved	Must be kept at reset value.
1	IRC8MSTB	<p>IRC8M Internal 8MHz RC Oscillator stabilization Flag</p> <p>Set by hardware to indicate if the IRC8M oscillator is stable and ready for use.</p> <p>0: IRC8M oscillator is not stable</p> <p>1: IRC8M oscillator is stable</p>
0	IRC8MEN	<p>Internal 8MHz RC oscillator Enable</p> <p>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 CKMEN is set.</p> <p>0: Internal 8 MHz RC oscillator disabled</p> <p>1: Internal 8 MHz RC oscillator enabled</p>

### 5.3.2. Clock configuration register 0 (RCU\_CFG0)

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
Reserved	PLLMF[4]	ADCPSC[2]		CKOUT0SEL[3:0]		USBFSPSC[1:0]		PLLMF[3:0]		PREDV0_LSB		PLLSEL			
	rw	rw		rw		rw		rw		rw		rw		rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADCPSC[1:0]		APB2PSC[2:0]		APB1PSC[2:0]		AHPBSC[3:0]		SCSS[1:0]		SCS[1:0]					
	rw		rw		rw		rw		rw		r		rw		

Bits	Fields	Descriptions
31:30	Reserved	Must be kept at reset value.
29	PLLMF[4]	<p>Bit 4 of PLLMF</p> <p>see bits 21:18 of RCU_CFG0</p>
28	ADCPSC[2]	<p>Bit 2 of ADCPSC</p> <p>see bits 15:14 of RCU_CFG0</p>
27:24	CKOUT0SEL[3:0]	<p>CKOUT0 Clock Source Selection</p> <p>Set and reset by software.</p> <p>00xx: No clock selected</p> <p>0100: System clock selected</p> <p>0101: High Speed 8M Internal Oscillator clock selected</p> <p>0110: External High Speed oscillator clock selected</p> <p>0111: (CK_PLL / 2) clock selected</p>

		1000: CK_PLL1 clock selected 1001: CK_PLL2 clock divided by 2 selected 1010: EXT1 selected 1011: CK_PLL2 clock selected
23:22	USBFSPSC[1:0]	<p>USBFS clock prescaler selection</p> <p>Set and reset by software to control the USBFS clock prescaler value. The USBFS clock must be 48MHz. These bits can't be reset if the USBFS clock is enabled.</p> <p>00: CK_USBFS = CK_PLL / 1.5 01: CK_USBFS = CK_PLL 10: CK_USBFS = CK_PLL / 2.5 11: CK_USBFS = CK_PLL / 2</p>
21:18	PLLMF[3:0]	<p>The PLL clock multiplication factor</p> <p>Bit 29 of RCU_CFG0 and these bits are written by software to define the PLL multiplication factor</p> <p><b>Caution:</b> The PLL output frequency must not exceed 108 MHz</p> <p>00000: (PLL source clock x 2) 00001: (PLL source clock x 3) 00010: (PLL source clock x 4) 00011: (PLL source clock x 5) 00100: (PLL source clock x 6) 00101: (PLL source clock x 7) 00110: (PLL source clock x 8) 00111: (PLL source clock x 9) 01000: (PLL source clock x 10) 01001: (PLL source clock x 11) 01010: (PLL source clock x 12) 01011: (PLL source clock x 13) 01100: (PLL source clock x 14) 01101: (PLL source clock x 6.5) 01110: (PLL source clock x 16) 01111: (PLL source clock x 16) 10000: (PLL source clock x 17) 10001: (PLL source clock x 18) 10010: (PLL source clock x 19) 10011: (PLL source clock x 20) 10100: (PLL source clock x 21) 10101: (PLL source clock x 22) 10110: (PLL source clock x 23) 10111: (PLL source clock x 24) 11000: (PLL source clock x 25) 11001: (PLL source clock x 26) 11010: (PLL source clock x 27) 11011: (PLL source clock x 28)</p>

---

		11100: (PLL source clock x 29) 11101: (PLL source clock x 30) 11110: (PLL source clock x 31) 11111: (PLL source clock x 32)
17	PREDV0_LSB	<p>The LSB of PREDV0 division factor</p> <p>This bit is the same bit as PREDV0 division factor bit [0] from RCU_CFG1. Changing the PREDV0 division factor bit [0] from RCU_CFG1, this bit is also changed. When the PREDV0 division factor bits [3:1] are not set, this bit controls PREDV0 input clock divided by 2 or not.</p>
16	PLLSEL	<p>PLL Clock Source Selection</p> <p>Set and reset by software to control the PLL clock source.</p> <p>0: (IRC8M / 2) clock selected as source clock of PLL 1: HXTAL selected as source clock of PLL</p>
15:14	ADCPSC[1:0]	<p>ADC clock prescaler selection</p> <p>These bits and bit 28 of RCU_CFG0 are written by software to define the ADC prescaler factor. Set and cleared by software.</p> <p>000: (CK_APB2 / 2) selected 001: (CK_APB2 / 4) selected 010: (CK_APB2 / 6) selected 011: (CK_APB2 / 8) selected 100: (CK_APB2 / 2) selected 101: (CK_APB2 / 12) selected 110: (CK_APB2 / 8) selected 111: (CK_APB2 / 16) 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 100: (CK_AHB / 2) selected 101: (CK_AHB / 4) selected 110: (CK_AHB / 8) selected 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>Caution: The CK_APB1 output frequency must not exceed 60 MHz.</p> <p>0xx: CK_AHB selected 100: (CK_AHB / 2) selected 101: (CK_AHB / 4) selected 110: (CK_AHB / 8) selected 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>

0xxx: CK\_SYS selected  
 1000: (CK\_SYS / 2) selected  
 1001: (CK\_SYS / 4) selected  
 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 HXTAL failure is detected by HXTAL clock monitor when HXTAL is selected directly or indirectly as the clock source of CK_SYS  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

### 5.3.3. Clock interrupt register (RCU\_INT)

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	PLL2	PLL1	PLL	HXTAL	IRC8M	LXTAL	IRC40K
									w	w	w	w	w	w	w	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved	PLL2 STBIE	PLL1 STBIE	PLL STBIE	HXTAL STBIE	IRC8M STBIE	LXTAL STBIE	IRC40K STBIE	CKMIF	PLL2 STBIF	PLL1 STBIF	PLL STBIF	HXTAL STBIF	IRC8M STBIF	LXTAL STBIF	IRC40K STBIF	
	rw	rw	rw	rw	rw	rw	rw	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	PLL2STBIC	<p>PLL2 stabilization Interrupt Clear</p> <p>Write 1 by software to reset the PLL2STBIF flag.</p> <p>0: Not reset PLL2STBIF flag</p> <p>1: Reset PLL2STBIF flag</p>
21	PLL1STBIC	<p>PLL1 stabilization Interrupt Clear</p> <p>Write 1 by software to reset the PLL1STBIF flag.</p> <p>0: Not reset PLL1STBIF flag</p> <p>1: Reset PLL1STBIF flag</p>
20	PLLSTBIC	<p>PLL stabilization Interrupt Clear</p> <p>Write 1 by software to reset the PLLSTBIF flag.</p> <p>0: Not reset PLLSTBIF flag</p> <p>1: Reset PLLSTBIF flag</p>
19	HXTALSTBIC	<p>HXTAL Stabilization Interrupt Clear</p> <p>Write 1 by software to reset the HXTALSTBIF flag.</p> <p>0: Not reset HXTALSTBIF flag</p> <p>1: Reset HXTALSTBIF flag</p>
18	IRC8MSTBIC	<p>IRC8M Stabilization Interrupt Clear</p> <p>Write 1 by software to reset the IRC8MSTBIF flag.</p> <p>0: Not reset IRC8MSTBIF flag</p> <p>1: Reset IRC8MSTBIF flag</p>
17	LXTALSTBIC	<p>LXTAL Stabilization Interrupt Clear</p> <p>Write 1 by software to reset the LXTALSTBIF flag.</p> <p>0: Not reset LXTALSTBIF flag</p> <p>1: Reset LXTALSTBIF flag</p>
16	IRC40KSTBIC	<p>IRC40K Stabilization Interrupt Clear</p> <p>Write 1 by software to reset the IRC40KSTBIF flag.</p> <p>0: Not reset IRC40KSTBIF flag</p> <p>1: Reset IRC40KSTBIF flag</p>
15	Reserved	Must be kept at reset value
14	PLL2STBIE	<p>PLL2 Stabilization Interrupt Enable</p> <p>Set and reset by software to enable/disable the PLL2 stabilization interrupt.</p> <p>0: Disable the PLL2 stabilization interrupt</p> <p>1: Enable the PLL2 stabilization interrupt</p>

13	PLL1STBIE	PLL1 Stabilization Interrupt Enable Set and reset by software to enable/disable the PLL1 stabilization interrupt. 0: Disable the PLL1 stabilization interrupt 1: Enable the PLL1 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	IRC8M Stabilization Interrupt Enable Set and reset by software to enable/disable the IRC8M stabilization interrupt 0: Disable the IRC8M stabilization interrupt 1: Enable the IRC8M stabilization interrupt
9	LXTALSTBIE	LXTAL Stabilization Interrupt Enable LXTAL stabilization interrupt enable/disable control 0: Disable the LXTAL stabilization interrupt 1: Enable the LXTAL stabilization interrupt
8	IRC40KSTBIE	IRC40K Stabilization interrupt enable IRC40K stabilization interrupt enable/disable control 0: Disable the IRC40K stabilization interrupt 1: Enable the IRC40K stabilization interrupt
7	CKMIF	HXTAL Clock Stuck Interrupt Flag Set by hardware when the HXTAL clock is stuck. Reset when setting the CKMIC bit by software. 0: Clock operating normally 1: HXTAL clock stuck
6	PLL2STBIF	PLL2 stabilization interrupt flag Set by hardware when the PLL2 is stable and the PLL2STBIE bit is set. Reset when setting the PLL2STBIC bit by software. 0: No PLL2 stabilization interrupt generated 1: PLL2 stabilization interrupt generated
5	PLL1STBIF	PLL1 stabilization interrupt flag Set by hardware when the PLL1 is stable and the PLL1STBIE bit is set. Reset when setting the PLL1STBIC bit by software. 0: No PLL1 stabilization interrupt generated 1: PLL1 stabilization interrupt generated

---

4	PLLSTBIF	PLL stabilization interrupt flag Set by hardware when the PLL is stable and the PLLSTBIE bit is set. Reset when setting the PLLSTBIC bit by software. 0: No PLL stabilization interrupt generated 1: PLL stabilization interrupt generated
3	HXTALSTBIF	HXTAL stabilization interrupt flag Set by hardware when the High speed 3 ~ 25 MHz crystal oscillator clock is stable and the HXTALSTBIE bit is set. Reset when setting the HXTALSTBIC bit by software. 0: No HXTAL stabilization interrupt generated 1: HXTAL stabilization interrupt generated
2	IRC8MSTBIF	IRC8M stabilization interrupt flag Set by hardware when the Internal 8 MHz RC oscillator clock is stable and the IRC8MSTBIE bit is set. Reset when setting the IRC8MSTBIC bit by software. 0: No IRC8M stabilization interrupt generated 1: IRC8M stabilization interrupt generated
1	LXTALSTBIF	LXTAL stabilization interrupt flag Set by hardware when the Low speed 32,768 Hz crystal oscillator clock is stable and the LXTALSTBIE bit is set. Reset when setting the LXTALSTBIC bit by software. 0: No LXTAL stabilization interrupt generated 1: LXTAL stabilization interrupt generated
0	IRC40KSTBIF	IRC40K stabilization interrupt flag Set by hardware when the Internal 40kHz RC oscillator clock is stable and the IRC40KSTBIE bit is set. Reset when setting the IRC40KSTBIC bit by software. 0: No IRC40K stabilization clock ready interrupt generated 1: IRC40K stabilization interrupt generated

### 5.3.4. 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															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Reserved	USART0 RST	Reserved	SPI0RST	TIMER0R ST	ADC1RS T	ADC0RS T	Reserved	PERST	PDRST	PCRST	PBRST	PARST	Reserved	AFRST
	rw		rw	rw	rw	rw		rw	rw	rw	rw	rw		rw

Bits	Fields	Descriptions
31:15	Reserved	Must be kept at reset value
14	USART0RST	<p>USART0 Reset</p> <p>This bit is set and reset by software.</p> <p>0: No reset</p> <p>1: Reset the USART0</p>
13	Reserved	Must be kept at reset value
12	SPI0RST	<p>SPI0 reset</p> <p>This bit is set and reset by software.</p> <p>0: No reset</p> <p>1: Reset the SPI0</p>
11	TIMER0RST	<p>Timer 0 reset</p> <p>This bit is set and reset by software.</p> <p>0: No reset</p> <p>1: Reset the TIMER0</p>
10	ADC1RST	<p>ADC1 reset</p> <p>This bit is set and reset by software.</p> <p>0: No reset</p> <p>1: Reset the ADC1</p>
9	ADC0RST	<p>ADC0 reset</p> <p>This bit is set and reset by software.</p> <p>0: No reset</p> <p>1: Reset the ADC0</p>
8:7	Reserved	Must be kept at reset value
6	PERST	<p>GPIO port E reset</p> <p>This bit is set and reset by software.</p> <p>0: No reset</p> <p>1: Reset the GPIO port E</p>
5	PDRST	<p>GPIO port D reset</p> <p>This bit is set and reset by software.</p> <p>0: No reset</p> <p>1: Reset the GPIO port D</p>
4	PCRST	<p>GPIO port C reset</p> <p>This bit is set and reset by software.</p> <p>0: No reset</p>

1: Reset the GPIO port C

3	PBRST	GPIO port B reset This bit is set and reset by software. 0: No reset 1: Reset the GPIO port B
2	PARST	GPIO port A reset This bit is set and reset by software. 0: No reset 1: Reset the GPIO port A
1	Reserved	Must be kept at reset value
0	AFRST	Alternate function I/O reset This bit is set and reset by software. 0: No reset 1: Reset Alternate Function I/O

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

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	DACRST	PMURST	BKPIRST	CAN1RS T	CAN0RS T	Reserved	I2C1RST	I2C0RST	UART4R ST	UART3R ST	USART2 RST	USART1 RST	Reserved		
	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI2RST	SPI1RST	Reserved	WWWDGT RST	Reserved				TIMER6R ST	TIMER5R ST	TIMER4R ST	TIMER3R ST	TIMER2R ST	TIMER1R ST		
rw	rw		rw					rw	rw	rw	rw	rw	rw	rw	

Bits	Fields	Descriptions
31:30	Reserved	Must be kept at reset value
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	BKPIRST	<p>Backup interface reset</p> <p>This bit is set and reset by software.</p> <p>0: No reset</p> <p>1: Reset backup interface</p>
26	CAN1RST	<p>CAN1 reset</p> <p>This bit is set and reset by software.</p> <p>0: No reset</p> <p>1: Reset the CAN1</p>
25	CAN0RST	<p>CAN0 reset</p> <p>This bit is set and reset by software.</p> <p>0: No reset</p> <p>1: Reset the CAN0</p>
24:23	Reserved	Must be kept at reset value
22	I2C1RST	<p>I2C1 reset</p> <p>This bit is set and reset by software.</p> <p>0: No reset</p> <p>1: Reset the I2C1</p>
21	I2C0RST	<p>I2C0 reset</p> <p>This bit is set and reset by software.</p> <p>0: No reset</p> <p>1: Reset the I2C0</p>
20	UART4RST	<p>UART4 reset</p> <p>This bit is set and reset by software.</p> <p>0: No reset</p> <p>1: Reset the UART4</p>
19	UART3RST	<p>UART3 reset</p> <p>This bit is set and reset by software.</p> <p>0: No reset</p> <p>1: Reset the UART3</p>
18	USART2RST	<p>USART2 reset</p> <p>This bit is set and reset by software.</p> <p>0: No reset</p> <p>1: Reset the USART2</p>
17	USART1RST	<p>USART1 reset</p> <p>This bit is set and reset by software.</p> <p>0: No reset</p> <p>1: Reset the USART1</p>

16	Reserved	Must be kept at reset value
15	SPI2RST	<p>SPI2 reset</p> <p>This bit is set and reset by software.</p> <p>0: No reset</p> <p>1: Reset the SPI2</p>
14	SPI1RST	<p>SPI1 reset</p> <p>This bit is set and reset by software.</p> <p>0: No reset</p> <p>1: Reset the SPI1</p>
13:12	Reserved	Must be kept at reset value
11	WWDGTRST	<p>WWDGT reset</p> <p>This bit is set and reset by software.</p> <p>0: No reset</p> <p>1: Reset the WWDGT</p>
10:6	Reserved	Must be kept at reset value
5	TIMER6RST	<p>TIMER6 reset</p> <p>This bit is set and reset by software.</p> <p>0: No reset</p> <p>1: Reset the TIMER6</p>
4	TIMER5RST	<p>TIMER5 reset</p> <p>This bit is set and reset by software.</p> <p>0: No reset</p> <p>1: Reset the TIMER5</p>
3	TIMER4RST	<p>TIMER4 reset</p> <p>This bit is set and reset by software.</p> <p>0: No reset</p> <p>1: Reset the TIMER4</p>
2	TIMER3RST	<p>TIMER3 reset</p> <p>This bit is set and reset by software.</p> <p>0: No reset</p> <p>1: Reset the TIMER3</p>
1	TIMER2RST	<p>TIMER2 reset</p> <p>This bit is set and reset by software.</p> <p>0: No reset</p> <p>1: Reset the TIMER2</p>
0	TIMER1RST	<p>TIMER1 reset</p> <p>This bit is set and reset by software.</p> <p>0: No reset</p>

1: Reset the TIMER1

### 5.3.6. 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)

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	USBFSEN N	Reserved		EXMCEN	Reserved	CRCEN	Reserved	FMCSPEN N	Reserved	SRAMSPEN	DMA1EN	DMA0EN			

rw                    rw

Bits	Fields	Descriptions
31:13	Reserved	Must be kept at reset value
12	USBFSEN	USBFS clock enable This bit is set and reset by software. 0: Disabled USBFS clock 1: Enabled USBFS clock
11:9	Reserved	Must be kept at reset value
8	EXMCEN	EXMC clock enable This bit is set and reset by software. 0: Disabled EXMC clock 1: Enabled EXMC clock
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	FMCSPEN	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	<p>SRAM interface clock enable when sleep mode</p> <p>This bit is set and reset by software to enable/disable SRAM interface clock during Sleep mode.</p> <p>0: Disabled SRAM interface clock during Sleep mode.</p> <p>1: Enabled SRAM interface clock during Sleep mode</p>
1	DMA1EN	<p>DMA1 clock enable</p> <p>This bit is set and reset by software.</p> <p>0: Disabled DMA1 clock</p> <p>1: Enabled DMA1 clock</p>
0	DMA0EN	<p>DMA0 clock enable</p> <p>This bit is set and reset by software.</p> <p>0: Disabled DMA0 clock</p> <p>1: Enabled DMA0 clock</p>

### 5.3.7. 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															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	USART0 EN	Reserved	SPI0EN	TIMER0EN	ADC1EN	ADC0EN	Reserved	PEEN	PDEN	PCEN	PBEN	PAEN	Reserved	AFEN	
	rw		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	USART0EN	<p>USART0 clock enable</p> <p>This bit is set and reset by software.</p> <p>0: Disabled USART0 clock</p> <p>1: Enabled USART0 clock</p>
13	Reserved	Must be kept at reset value
12	SPI0EN	<p>SPI0 clock enable</p> <p>This bit is set and reset by software.</p> <p>0: Disabled SPI0 clock</p> <p>1: Enabled SPI0 clock</p>

11	TIMER0EN	TIMER0 clock enable This bit is set and reset by software. 0: Disabled TIMER0 clock 1: Enabled TIMER0 clock
10	ADC1EN	ADC1 clock enable This bit is set and reset by software. 0: Disabled ADC1 clock 1: Enabled ADC1 clock
9	ADC0EN	ADC0 clock enable This bit is set and reset by software. 0: Disabled ADC0 clock 1: Enabled ADC0 clock
8:7	Reserved	Must be kept at reset value
6	PEEN	GPIO port E clock enable This bit is set and reset by software. 0: Disabled GPIO port E clock 1: Enabled GPIO port E clock
5	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
4	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
3	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
2	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
1	Reserved	Must be kept at reset value
0	AFEN	Alternate function IO clock enable This bit is set and reset by software. 0: Disabled Alternate Function IO clock 1: Enabled Alternate Function IO clock

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

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	DACEN	PMUEN	BKPIEN	CAN1EN	CAN0EN	Reserved	I2C1EN	I2C0EN	UART4EN	UART3EN	USART2EN	USART1EN	Reserved		
	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI2EN	SPI1EN	Reserved	WWDDGTEN	Reserved				TIMER6EN	TIMER5EN	TIMER4EN	TIMER3EN	TIMER2EN	TIMER1EN		
rw	rw		rw					rw	rw	rw	rw	rw	rw	rw	

Bits	Fields	Descriptions
31:30	Reserved	Must be kept at reset value
29	DACEN	<p>DAC clock enable</p> <p>This bit is set and reset by software.</p> <p>0: Disabled DAC clock</p> <p>1: Enabled DAC clock</p>
28	PMUEN	<p>PMU clock enable</p> <p>This bit is set and reset by software.</p> <p>0: Disabled PMU clock</p> <p>1: Enabled PMU clock</p>
27	BKPIEN	<p>Backup interface clock enable</p> <p>This bit is set and reset by software.</p> <p>0: Disabled Backup interface clock</p> <p>1: Enabled Backup interface clock</p>
26	CAN1EN	<p>CAN1 clock enable</p> <p>This bit is set and reset by software.</p> <p>0: Disabled CAN1 clock</p> <p>1: Enabled CAN1 clock</p>
25	CAN0EN	<p>CAN0 clock enable</p> <p>This bit is set and reset by software.</p> <p>0: Disabled CAN0 clock</p> <p>1: Enabled CAN0 clock</p>
24:23	Reserved	Must be kept at reset value
22	I2C1EN	<p>I2C1 clock enable</p> <p>This bit is set and reset by software.</p>

		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	UART4EN	UART4 clock enable  This bit is set and reset by software. 0: Disabled UART4 clock 1: Enabled UART4 clock
19	UART3EN	UART3 clock enable  This bit is set and reset by software. 0: Disabled UART3 clock 1: Enabled UART3 clock
18	USART2EN	USART2 clock enable  This bit is set and reset by software. 0: Disabled USART2 clock 1: Enabled USART2 clock
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	WWDGT clock enable  This bit is set and reset by software. 0: Disabled WWDGT clock 1: Enabled WWDGT clock
10:6	Reserved	Must be kept at reset value
5	TIMER6EN	TIMER6 clock enable

		This bit is set and reset by software. 0: Disabled TIMER6 clock 1: Enabled TIMER6 clock
4	TIMER5EN	TIMER5 clock enable  This bit is set and reset by software. 0: Disabled TIMER5 clock 1: Enabled TIMER5 clock
3	TIMER4EN	TIMER4 clock enable  This bit is set and reset by software. 0: Disabled TIMER4 clock 1: Enabled TIMER4 clock
2	TIMER3EN	TIMER3 clock enable  This bit is set and reset by software. 0: Disabled TIMER3 clock 1: Enabled TIMER3 clock
1	TIMER2EN	TIMER2 clock enable  This bit is set and reset by software. 0: Disabled TIMER2 clock 1: Enabled TIMER2 clock
0	TIMER1EN	TIMER1 clock enable  This bit is set and reset by software. 0: Disabled TIMER1 clock 1: Enabled TIMER1 clock

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

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 (RCU\_BDCTL) are only reset after a Backup domain Reset. These bits can be modified only when the BKPRST bit in the Power control register (PMU\_CTL) is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Reserved														BKPRST		
															rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
RTCEN	Reserved				RTCSRC[1:0]		Reserved				LXTALBP	LXTALST	LXTALEN			
												S	B		rw r rw	

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
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. Once the RTC 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 source clock 10: CK_IRC40K selected as RTC source clock 11: (CK_HXTAL / 128) selected as RTC source clock
7:3	Reserved	Must be kept at reset value
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	Low speed crystal oscillator stabilization flag 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

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

Address offset: 0x24

Reset value: 0x0C00 0000, ALL reset flags reset by power Reset only, RSTFC/IRC40KEN 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	WWDGTRSTF	FWDGTRSTF	SW RSTF	POR RSTF	EP RSTF	Reserved	RSTFC					Reserved			
r	r	r	r	r	r			rw							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						Reserved							IRC40K STB	IRC40KE N	
													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 reset 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 reset generated.  Reset by writing 1 to the RSTFC bit.  0: No External PIN reset generated

		1: External PIN reset generated
25	Reserved	Must be kept at reset value
24	RSTFC	<p>Reset flag clear</p> <p>This bit is set by software to clear all reset flags.</p> <p>0: Not clear reset flags</p> <p>1: Clear reset flags</p>
23:2	Reserved	Must be kept at reset value
1	IRC40KSTB	<p>IRC40K stabilization flag</p> <p>Set by hardware to indicate if the IRC40K output clock is stable and ready for use.</p> <p>0: IRC40K is not stable</p> <p>1: IRC40K is stable</p>
0	IRC40KEN	<p>IRC40K enable</p> <p>Set and reset by software.</p> <p>0: Disable IRC40K</p> <p>1: Enable IRC40K</p>

### 5.3.11. 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															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	USBFSR ST														

rw

Bits	Fields	Descriptions
31:13	Reserved	Must be kept at reset value
12	USBFSRST	<p>USBFS reset</p> <p>This bit is set and reset by software.</p> <p>0: No reset</p> <p>1: Reset the USBFS</p>
11:0	Reserved	Must be kept at reset value

### 5.3.12. Clock 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)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Reserved														I2S2SEL	I2S1SEL	PREDV0 SEL
														rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
PLL2MF[3:0]				PLL1MF[3:0]				PREDV1[3:0]				PREDV0[3:0]				
rw				rw				rw				rw				

Bits	Fields	Descriptions
31:19	Reserved	Must be kept at reset value
18	I2S2SEL	I2S2 Clock Source Selection Set and reset by software to control the I2S2 clock source. 0: System clock selected as I2S2 source clock 1: (CK_PLL2 x 2) selected as I2S2 source clock
17	I2S1SEL	I2S1 Clock Source Selection Set and reset by software to control the I2S1 clock source. 0: System clock selected as I2S1 source clock 1: (CK_PLL2 x 2) selected as I2S1 source clock
16	PREDV0SEL	PREDV0 input Clock Source Selection Set and reset by software. 0: HXTAL selected as PREDV0 input source clock 1: CK_PLL1 selected as PREDV0 input source clock
15:12	PLL2MF[3:0]	The PLL2 clock multiplication factor Set and reset by software. 00xx: reserve 010x: reserve 0110: (PLL2 source clock x 8) 0111: (PLL2 source clock x 9) 1000 : (PLL2 source clock x 10) 1001: (PLL2 source clock x 11) 1010: (PLL2 source clock x 12) 1011: (PLL2 source clock x 13) 1100: (PLL2 source clock x 14) 1101: (PLL2 source clock x 15) 1110: (PLL2 source clock x 16)

		1111: (PLL2 source clock x 20)
11:8	PLL1MF[3:0]	<p>The PLL1 clock multiplication factor Set and reset by software.</p> <p>00xx: reserve 010x: reserve 0110: (PLL1 source clock x 8) 0111: (PLL1 source clock x 9) 1000 :(PLL1 source clock x 10) 1001: (PLL1 source clock x 11) 1010: (PLL1 source clock x 12) 1011: (PLL1 source clock x 13) 1100: (PLL1 source clock x 14) 1101: (PLL1 source clock x 15) 1110 :(PLL1 source clock x 16) 1111: (PLL1 source clock x 20)</p>
7:4	PREDV1[3:0]	<p>PREDV1 division factor This bit is set and reset by software. These bits can be written when PLL1 and PLL2 are disable</p> <p>0000: PREDV1 input source clock not divided 0001: PREDV1 input source clock divided by 2 0010: PREDV1 input source clock divided by 3 0011: PREDV1 input source clock divided by 4 0100: PREDV1 input source clock divided by 5 0101: PREDV1 input source clock divided by 6 0110: PREDV1 input source clock divided by 7 0111: PREDV1 input source clock divided by 8 1000: PREDV1 input source clock divided by 9 1001: PREDV1 input source clock divided by 10 1010: PREDV1 input source clock divided by 11 1011: PREDV1 input source clock divided by 12 1100: PREDV1 input source clock divided by 13 1101: PREDV2 input source clock divided by 14 1110: PREDV2 input source clock divided by 15 1111: PREDV2 input source clock divided by 16</p>
3:0	PREDV0[3:0]	<p>PREDV0 division factor This bit is set and reset by software. These bits can be written when PLL is disable. <b>Note:</b> The bit 0 of PREDV0 is same as bit 17 of RCU_CFG0, so modifying Bit 17 of RCU_CFG0 also modifies bit 0 of RCU_CFG1.</p> <p>0000: PREDV0 input source clock not divided 0001: PREDV0 input source clock divided by 2 0010: PREDV0 input source clock divided by 3 0011: PREDV0 input source clock divided by 4</p>

- 0100: PREDV0 input source clock divided by 5
- 0101: PREDV0 input source clock divided by 6
- 0110: PREDV0 input source clock divided by 7
- 0111: PREDV0 input source clock divided by 8
- 1000: PREDV0 input source clock divided by 9
- 1001: PREDV0 input source clock divided by 10
- 1010: PREDV0 input source clock divided by 11
- 1011: PREDV0 input source clock divided by 12
- 1100: PREDV0 input source clock divided by 13
- 1101: PREDV0 input source clock divided by 14
- 1110: PREDV0 input source clock divided by 15
- 1111: PREDV0 input source clock divided by 16

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

Address offset: 0x34

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
Reserved															DSLPVS[1:0]

rw

Bits	Fields	Descriptions
31:2	Reserved	Must be kept at reset value
1:0	DSLPVS[1:0]	<p>Deep-sleep mode voltage select</p> <p>These bits are set and reset by software</p> <p>00 : The core voltage is 1.2V in Deep-sleep mode</p> <p>01 : The core voltage is 1.1V in Deep-sleep mode</p> <p>10 : The core voltage is 1.0V in Deep-sleep mode</p> <p>11 : The core voltage is 0.9V in Deep-sleep mode</p>

## 6. Interrupt/event controller (EXTI)

### 6.1. Overview

RISC-V integrates the Enhancement Core-Local Interrupt Controller (ECLIC) for efficient interrupts processing. ECLIC is designed to provide low-latency, vectored, pre-emptive interrupts for RISC-V systems. When activated the ECLIC subsumes and replaces the existing RISC-V local interrupt scheme (CLINT). The CLIC design has a base design that requires minimal hardware, but supports additional extensions to provide hardware acceleration. The goal of the ECLIC design is to provide support for a variety of software ABI and interrupt models, without complex hardware that can impact high-performance processor implementations. It's tightly coupled to the processor core. You can read the Technical Reference Manual of RISC-V for more details about ECLIC.

EXTI (interrupt/event controller) contains up to 19 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.

### 6.2. Characteristics

- Up to 68 maskable peripheral interrupts.
- 4 bits interrupt priority configuration - 16 priority levels.
- Support interrupt pre-emption and tail-chaining.
- Wake up system from power saving mode.
- Up to 19 independent edge detectors in EXTI.
- Three trigger types: rising, falling and both edges.
- Software interrupt or event trigger.
- Trigger sources configurable.

### 6.3. Function overview

The RISC-V processor and the Enhancement Core-Local Interrupt Controller (ECLIC) prioritize and handle all interrupts in machine mode.

The processor supports tail-chaining, which enables back-to-back interrupts to be performed without the overhead of state saving and restoration. The following tables list all interrupt types.

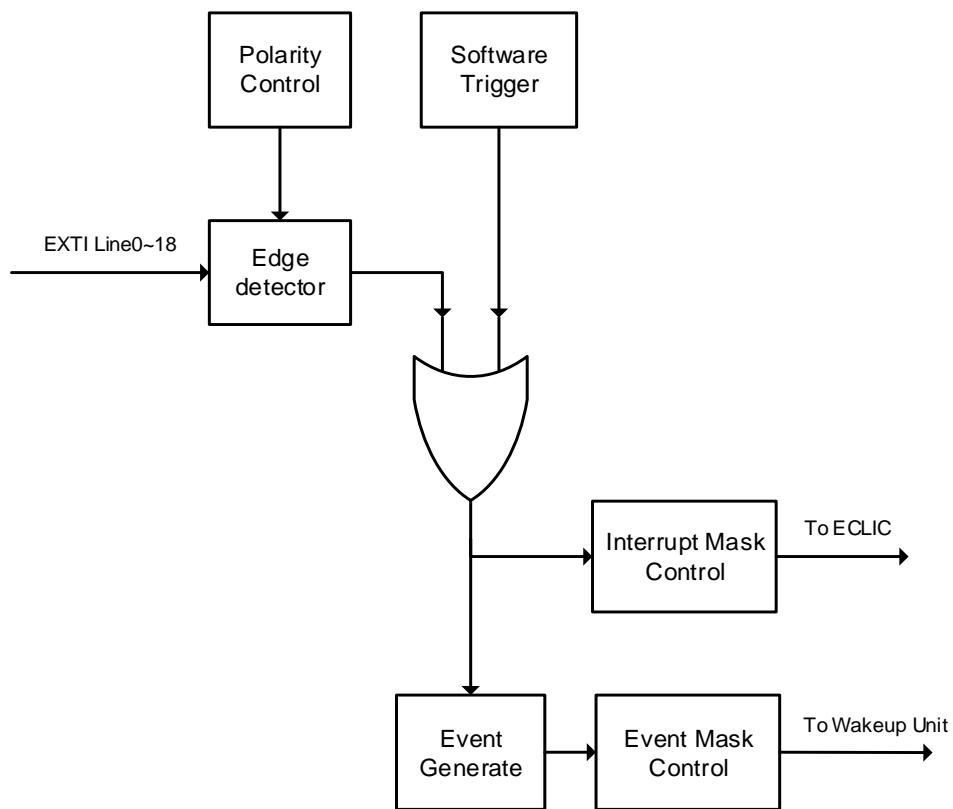
Table 6-1. Interrupt vector table

Vector Number	Interrupt Description	Vector Address
3	CLIC_INT_SFT	0x0000_000C
7	CLIC_INT_TMR	0x0000_001C
17	CLIC_INT_BWEI	0x0000_0044
18	CLIC_INT_PMOVI	0x0000_0048
19	WWDG interrupt	0x0000_004C
20	LVD from EXTI interrupt	0x0000_0050
21	Tamper interrupt	0x0000_0054
22	RTC global interrupt	0x0000_0058
23	FMC global interrupt	0x0000_005C
24	RCU global interrupt	0x0000_0060
25	EXTI Line0 interrupt	0x0000_0064
26	EXTI Line1 interrupt	0x0000_0068
27	EXTI Line2 interrupt	0x0000_006C
28	EXTI Line3 interrupt	0x0000_0070
29	EXTI Line4 interrupt	0x0000_0074
30	DMA0 channel0 global interrupt	0x0000_0078
31	DMA0 channel1 global interrupt	0x0000_007C
32	DMA0 channel2 global interrupt	0x0000_0080
33	DMA0 channel3 global interrupt	0x0000_0084
34	DMA0 channel4 global interrupt	0x0000_0088
35	DMA0 channel5 global interrupt	0x0000_008C
36	DMA0 channel6 global interrupt	0x0000_0090
37	ADC0 and ADC1 global interrupt	0x0000_0094
38	CAN0 TX interrupts	0x0000_0098
39	CAN0 RX0 interrupts	0x0000_009C
40	CAN0 RX1 interrupts	0x0000_00A0
41	CAN0 EWMC interrupts	0x0000_00A4
42	EXTI line[9:5] interrupts	0x0000_00A8
43	TIMER0 break interrupt	0x0000_00AC
44	TIMER0 update interrupt	0x0000_00B0
45	TIMER0 trigger and channel commutation interrupts	0x0000_00B4
46	TIMER0 channel capture compare interrupt	0x0000_00B8
47	TIMER1 global interrupt	0x0000_00BC
48	TIMER2 global interrupt	0x0000_00C0
49	TIMER3 global interrupt	0x0000_00C4
50	I2C0 event interrupt	0x0000_00C8
51	I2C0 error interrupt	0x0000_00CC
52	I2C1 event interrupt	0x0000_00D0
53	I2C1 error interrupt	0x0000_00D4

<b>Vector Number</b>	<b>Interrupt Description</b>	<b>Vector Address</b>
<b>54</b>	SPI0 global interrupt	0x0000_00D8
<b>55</b>	SPI1 global interrupt	0x0000_00DC
<b>56</b>	USART0 global interrupt	0x0000_00E0
<b>57</b>	USART1 global interrupt	0x0000_00E4
<b>58</b>	USART2 global interrupt	0x0000_00E8
<b>59</b>	EXTI line[15:10] interrupts	0x0000_00EC
<b>60</b>	RTC alarm from EXTI interrupt	0x0000_00F0
<b>61</b>	USBFS wakeup from EXTI interrupt	0x0000_00F4
<b>62</b>	Reserved	0x0000_00F8
<b>63</b>	Reserved	0x0000_00FC
<b>64</b>	Reserved	0x0000_0100
<b>65</b>	Reserved	0x0000_0104
<b>66</b>	Reserved	0x0000_0108
<b>67</b>	Reserved	0x0000_010C
<b>68</b>	Reserved	0x0000_0110
<b>69</b>	TIMER4 global interrupt	0x0000_0114
<b>70</b>	SPI2 global interrupt	0x0000_0118
<b>71</b>	UART3 global interrupt	0x0000_011C
<b>72</b>	UART4 global interrupt	0x0000_0120
<b>73</b>	TIMER5 global interrupt	0x0000_0124
<b>74</b>	TIMER6 global interrupt	0x0000_0128
<b>75</b>	DMA1 channel0 global interrupt	0x0000_012C
<b>76</b>	DMA1 channel1 global interrupt	0x0000_0130
<b>77</b>	DMA1 channel2 global interrupt	0x0000_0134
<b>78</b>	DMA1 channel3 global interrupt	0x0000_0138
<b>79</b>	DMA1 channel4 global interrupt	0x0000_013C
<b>80</b>	Reserved	0x0000_0140
<b>81</b>	Reserved	0x0000_0144
<b>82</b>	CAN1 TX interrupt	0x0000_0148
<b>83</b>	CAN1 RX0 interrupt	0x0000_014C
<b>84</b>	CAN1 RX1 interrupt	0x0000_0150
<b>85</b>	CAN1 EWMC interrupt	0x0000_0154
<b>86</b>	USBFS global interrupt	0x0000_0158

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

Figure 6-1. Block diagram of EXTI



## 6.5. External Interrupt and Event function overview

The EXTI contains up to 19 independent edge detectors and generates interrupts request or event 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 external lines from GPIO pins and 3 lines from internal modules (including LVD, RTC Alarm, USB Wakeup). All GPIO pins can be selected as an EXTI trigger source by configuring AFIO\_EXTI<sub>S</sub> registers in GPIO module (please refer to GPIO and AFIO section for detail).

EXTI can provide not only interrupts but also event signals to the processor. The RISC-V processor fully implements the Wait For Interrupt (WFI), Wait For Event (WFE) and the Send Event (SEV) instructions. EXTI can be used to wake up processor and the whole system when some expected event occurs, such as a special GPIO pin toggling or RTC alarm.

Table 6-2. EXTI source

EXTI Line Number	Source
0	PA0/PB0/PC0/PD0/PE0
1	PA1/PB1/PC1/PD1/PE1
2	PA2/PB2/PC2/PD2/PE2
3	PA3/PB3/PC3/PD3/PE3
4	PA4/PB4/PC4/PD4/PE4
5	PA5/PB5/PC5/PD5/PE5
6	PA6/PB6/PC6/PD6/PE6
7	PA7/PB7/PC7/PD7/PE7
8	PA8/PB8/PC8/PD8/PE8
9	PA9/PB9/PC9/PD9/PE9
10	PA10/PB10/PC10/PD10/PE10
11	PA11/PB11/PC11/PD11/PE11
12	PA12/PB12/PC12/PD12/PE12
13	PA13/PB13/PC13/PD13/PE13
14	PA14/PB14/PC14/PD14/PE14
15	PA15/PB15/PC15/PD15/PE15
16	LVD
17	RTC Alarm
18	USB Wakeup

## 6.6. Register definition

EXTI base address: 0x4001 0400

### 6.6.1. Interrupt enable register (EXTI\_INTEN)

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															INTEN18 INTEN17 INTEN16
															rw rw 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
31:19	Reserved	Must be kept at reset value
18: 0	INTENx	Interrupt enable bit 0: Interrupt from Linex is disabled. 1: Interrupt from Linex is enabled.

### 6.6.2. Event enable register (EXTI\_EVENT)

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															EVEN18 EVEN17 EVEN16
															rw rw 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
31:19	Reserved	Must be kept at reset value
18: 0	EVENx	Event enable bit 0: Event from Linex is disabled.

1: Event from Linex is enabled.

### 6.6.3. 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														RTEN18	RTEN17	RTEN16
														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	RTENO	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bits	Fields	Descriptions
31:19	Reserved	Must be kept at reset value
18:0	RTENx	Rising edge trigger enable 0: Rising edge of Linex is invalid 1: Rising edge of Linex is valid as an interrupt/event request

### 6.6.4. 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														FTEN18	FTEN17	FTEN16
														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	FTENO	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bits	Fields	Descriptions
31: 19	Reserved	Must be kept at reset value
18: 0	FTENx	Falling edge trigger enable 0: Falling edge of Linex is invalid 1: Falling edge of Linex is valid as an interrupt/event request

### 6.6.5. 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														SWIEV18	SWIEV17	SWIEV16
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	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bits	Fields	Descriptions
31:19	Reserved	Must be kept at reset value
18: 0	SWIEVx	Interrupt/Event software trigger 0: Deactivate the EXTIx software interrupt/event request 1: Activate the EXTIx software interrupt/event request

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

Address offset: 0x14

Reset value: undefined

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														PD18	PD17	PD16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
PD15	PD14	PD13	PD12	PD11	PD10	PD9	PD8	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0	
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	

Bits	Fields	Descriptions
31: 19	Reserved	Must be kept at reset value
18: 0	PDx	Interrupt pending status 0: EXTI Linex is not triggered 1: EXTI Linex is triggered. This bit is cleared to 0 by writing 1 to it.

## 7. General-purpose and alternate-function I/Os (GPIO and AFIO)

### 7.1. Overview

There are up to 80 general purpose I/O pins (GPIO), named PA0 ~ PA15, PB0 ~ PB15, PC0 ~ PC15, PD0 ~ PD15 and PE0 ~ PE15 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 external interrupt on the GPIO pins of the device have related control and configuration registers in the Interrupt/event Controller Unit (EXTI).

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), input, peripheral alternate function or 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.

### 7.2. 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.
- External interrupt with programmable trigger edge – using EXTI configuration registers.
- Analog input/output configuration.
- Alternate function input/output configuration.
- Port configuration lock.

### 7.3. Function overview

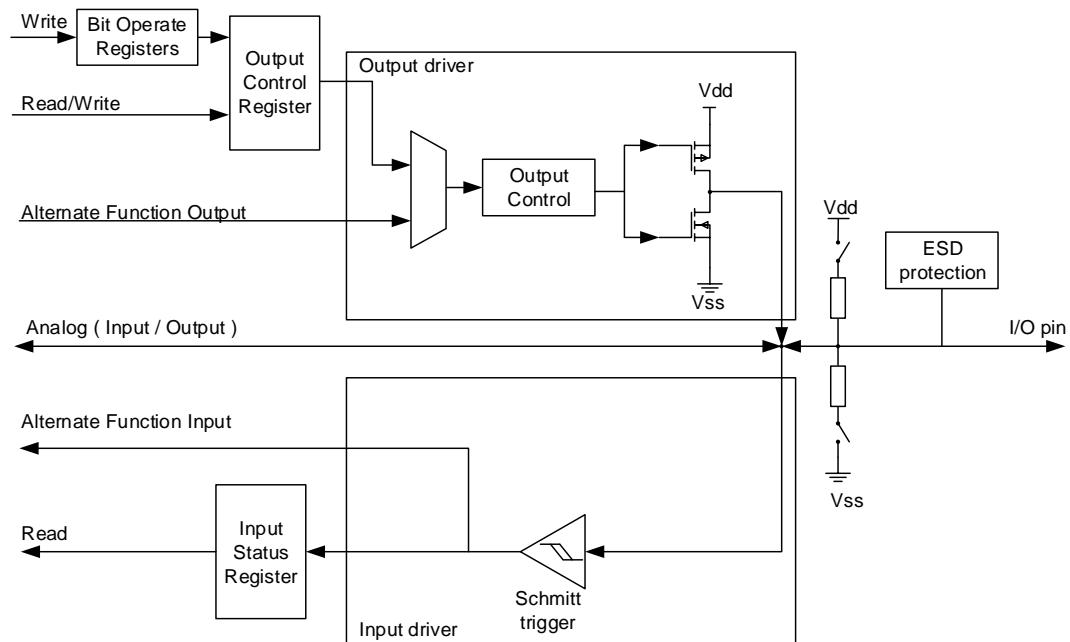
Each of the general-purpose I/O ports can be configured as 8 modes: analog inputs, input floating, input pull-down/pull-up, GPIO push-pull/open-drain or AFIO push-pull/open-drain mode by two GPIO configuration registers (GPIOx\_CTL0/GPIOx\_CTL1), and two 32-bits data registers (GPIOx\_ISTAT and GPIOx\_OCTL). [Table 7.1. GPIO configuration table](#) shows the details.

**Table 7.1. GPIO configuration table**

Configuration mode		CTL[1:0]	MD[1:0]	OCTL
Input	Analog	00	0	don't care
	Input floating	01		don't care
	Input pull-down	10		0
	Input pull-up	10		1
General purpose Output (GPIO)	Push-pull	00	00: Reserved 01: Speed up to 10MHz 10: Speed up to 2MHz 11: Speed up to 50MHz	0 or 1
	Open-drain	01		0 or 1
Alternate Function Output (AFIO)	Push-pull	10		don't care
	Open-drain	11		don't care

**Figure 7.1. Basic structure of a standard I/O port bit** shows the basic structure of an I/O port bit.

**Figure 7.1. Basic structure of a standard I/O port bit**



### 7.3.1. 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 JTAG/Serial-Wired Debug pins are in input PU/PD mode after reset:

- PA15: JTDI in PU mode.
- PA14: JTCK in PD mode.
- PA13: JTMS in PU mode.
- PB4: NJTRST in PU mode.
- PB3: JTDO in Floating mode.

The GPIO pins can be configured as inputs or outputs. When the GPIO pins are configured as input pins, all GPIO pins have an internal weak pull-up and weak pull-down which can be chosen. And the data on the external pins can be captured at every APB2 clock cycle to the port input status register (GPIOx\_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 (GPIOx\_OCTL) is output on the I/O pin.

There is no need to read-then-write when programming the GPIOx\_OCTL at bit level, user can modify only one or several bits in a single atomic APB2 write access by programming ‘1’ to the bit operate register (GPIOx\_BOP, or for clearing only GPIOx\_BC). The other bits will not be affected.

### 7.3.2. External interrupt/event lines

All ports have external interrupt capability. To use external interrupt lines, the port must be configured in input mode.

### 7.3.3. Alternate functions (AF)

When the port is configured as AFIO (set CTLy bits to “0b10” or “0b11”, and set MDy bits to “0b01”, “0b10”, or “0b11”, which is in GPIOx\_CTL0/GPIOx\_CTL1 registers), the port is used as peripheral alternate functions. The detail alternate function assignments for each port are in the device datasheet.

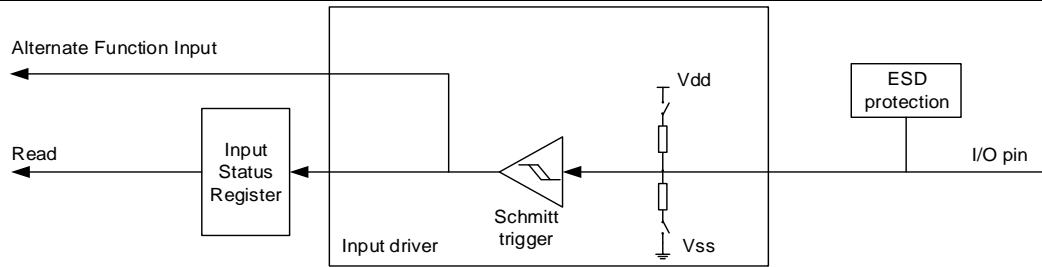
### 7.3.4. Input configuration

When GPIO pin is configured as Input:

- The schmitt trigger input is enabled.
- The weak pull-up and pull-down resistors could be chosen.
- Every APB2 clock cycle the data present on the I/O pin is got to the port input status Register.
- The output buffer is disabled.

[Figure 7.2. Input configuration](#) shows the input configuration.

**Figure 7.2. Input configuration**



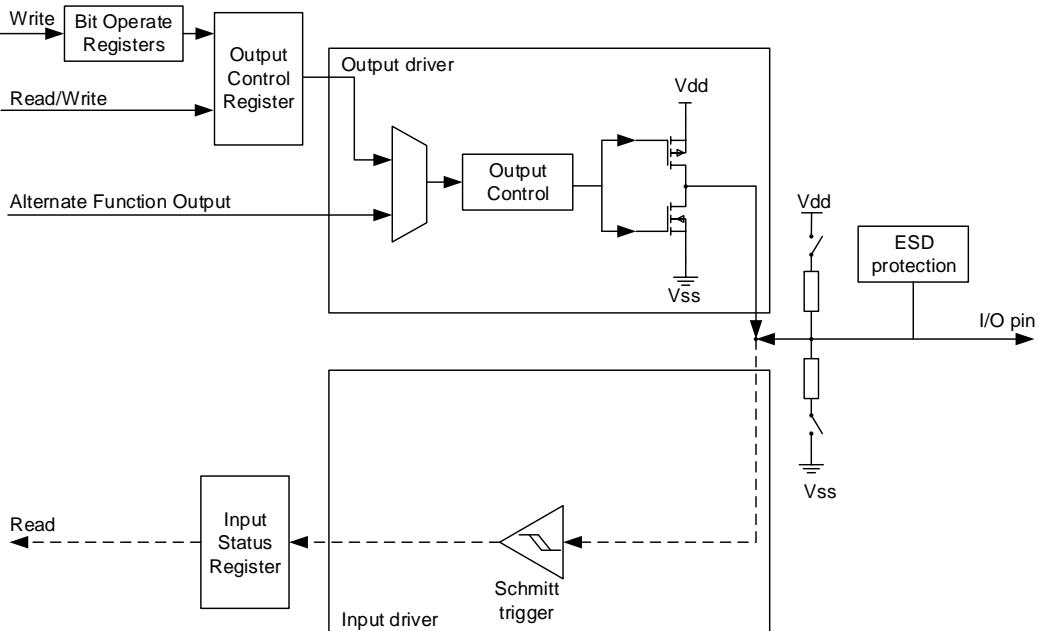
### 7.3.5. Output configuration

When GPIO pin is configured as output:

- The schmitt trigger input is enabled.
- The weak pull-up and pull-down resistors are disabled.
- The output buffer is enabled.
- Open Drain Mode: The pad output low level 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 output low level when a “0” in the output control register, while the pad output high level when a “1” in the output control register.
- A read access to the port output control register gets the last written value.
- A read access to the port input status register gets the I/O state.

[\*\*Figure 7.3. Output configuration\*\*](#) shows the output configuration.

**Figure 7.3. Output configuration**



### 7.3.6. 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 disabled.
- The port input status register of this I/O port bit is “0”.

[Figure 7.4. Analog configuration](#) shows the analog configuration.

**Figure 7.4. Analog configuration**



### 7.3.7. Alternate function (AF) configuration

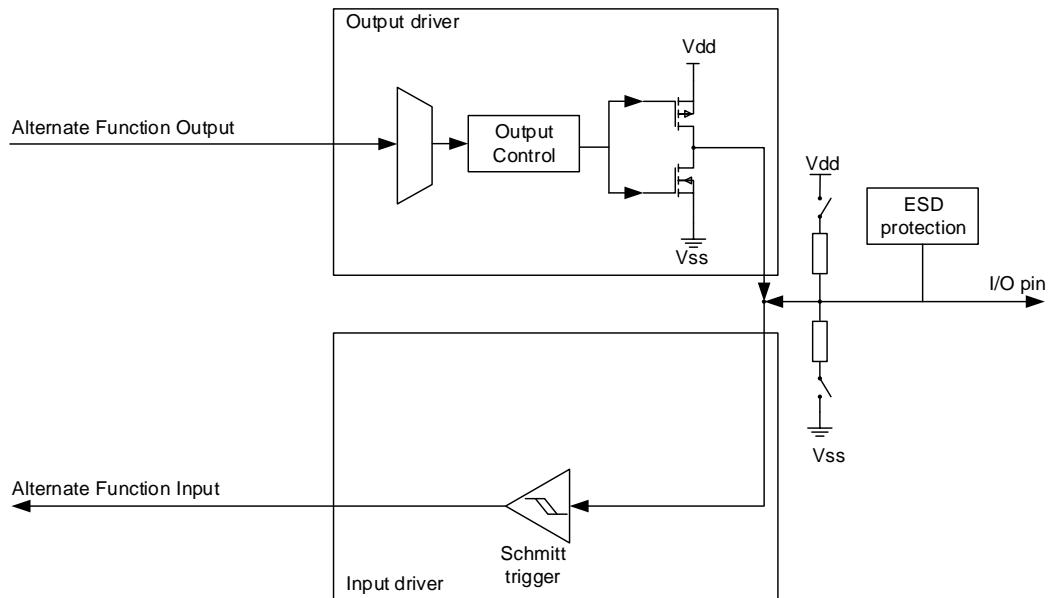
To suit for different device packages, the GPIO supports some alternate functions mapped to some other pins by software.

When be configured as alternate function:

- The output buffer is enabled in Open-Drain or Push-Pull configuration.
- The output buffer is driven by the peripheral.
- The schmitt trigger input is enabled.
- The weak pull-up and pull-down resistors could be chosen when input.
- The I/O pin data is stored into the port input status register every APB2 clock.
- A read access to the port input status register gets the I/O state.
- A read access to the port output control register gets the last written value.

[Figure 7.5. Alternate function configuration](#) shows the alternate function configuration.

**Figure 7.5. Alternate function configuration**



### 7.3.8. IO pin function selection

Each IO pin can implement many functions, each function selected by GPIO registers.

#### **GPIO:**

Each IO pin can be used for GPIO input function by configuring MDy bits to 0b00 in GPIOx\_CTL0/GPIOx\_CTL1 registers. And set output function by configuring MDy bits to 0b01, 0b10, or 0b11 and configuring CTLy bits of corresponding port in GPIOx\_CTL0/GPIOx\_CTL1 register to 0b00 (for GPIO push-pull output) or 0b01 (for GPIO open-drain output).

#### **Alternate function:**

Each IO pin can be used for AF input function by configuring MDy bits to 0b00 in GPIOx\_CTL0/GPIOx\_CTL1 registers. And set output function by configuring MDy bits to 0b01, 0b10, or 0b11 and configuring CTLy bits of corresponding port in GPIOx\_CTL0/GPIOx\_CTL1 register to 0b10 (for AF push-pull output) or 0b11 (for AF open-drain output).

### 7.3.9. GPIO locking function

The locking mechanism allows the IO configuration to be protected.

The protected registers are GPIOx\_CTL0, GPIOx\_CTL1. It allows the I/O configuration to be frozen by the 32-bit locking register (GPIOx\_LOCK). When the special LOCK sequence has been occurred on LKK bit in GPIOx\_LOCK register and the LKy bit is set in GPIOx\_LOCK register, the corresponding port is locked and the corresponding port configuration cannot be modified until the next reset. It should be recommended to be used in the configuration of driving a power module.

## 7.4. Remapping function I/O and debug configuration

### 7.4.1. Introduction

In order to expand the flexibility of the GPIO or the usage of peripheral functions, each I/O pin can be configured to have up to four different functions by setting the AFIO Port Configuration Register (AFIO\_PCF0/AFIO\_PCF1). Suitable pinout locations can be selected using the peripheral IO remapping function. Additionally, various GPIO pins can be selected to be the EXTI interrupt line by setting the relevant EXTI Source Selection Register (AFIO\_EXTI\_Sx) to trigger an interrupt or event.

### 7.4.2. Main features

- APB slave interface for register access.
- EXTI source selection.
- Each pin has up to four alternative functions for configuration.

### 7.4.3. JTAG alternate function remapping

The debug interface signals are mapped on the GPIO ports as shown in table below.

**Table 7.2. Debug interface signals**

Alternate function	GPIO port
JTMS	PA13
JTCK	PA14
JTDI	PA15
JTDO	PB3
NJTRST	PB4

To reduce the number of GPIOs used to debug, user can configure SWJ\_CFG [2:0] bits in the AFIO\_PCF0 to different value. Refer to table below.

**Table 7.3. Debug port mapping**

SWJ_CFG [2:0]	Available debug ports	SWJ I/O pin assigned				
		PA13/ JTMS	PA14/ JTCK	PA15/ JTDI	PB3/ JTDO	PB4/ NJTRST
000	JTAG-DP Reset state	•	•	•	•	•
001	JTAG-DP but without NJTRST	•	•	•	•	X
010	JTAG-DP Disabled	X	X	X	X	X
Other	Forbidden					

#### 7.4.4. TIMER AF remapping

**Table 7.4. TIMER0 alternate function remapping**

Alternate function	TIMER0_REMAP [1:0] = "00" (no remap)	TIMER0_REMAP [1:0] = "01" (partial remap)	TIMER0_REMAP [1:0] = "11" (full remap) <sup>(1)</sup>
TIMER0_ETI	PA12		PE7
TIMER0_CH0	PA8		PE9
TIMER0_CH1	PA9		PE11
TIMER0_CH2	PA10		PE13
TIMER0_CH3	PA11		PE14
TIMER0_BKIN	PB12 <sup>(2)</sup>	PA6	PE15
TIMER0_CH0_ON	PB13 <sup>(2)</sup>	PA7	PE8
TIMER0_CH1_ON	PB14 <sup>(2)</sup>	PB0	PE10
TIMER0_CH2_ON	PB15 <sup>(2)</sup>	PB1	PE12

1. For different packages and flash sizes please refer to the datasheet.

**Table 7.5. TIMER1 alternate function remapping**

Alternate function	TIMER1_REMAP [1:0] = "00" (no remap)	TIMER1_REMAP [1:0] = "01" (partial remap)	TIMER1_REMAP [1:0] = "10" (partial remap)	TIMER1_REMAP [1:0] = "11" (full remap) <sup>(2)</sup>
TIMER1_CH0/TIMER1_ETI <sup>(1)</sup>	PA0	PA15	PA0	PA15
TIMER1_CH1	PA1	PB3	PA1	PB3
TIMER1_CH2	PA2		PB10	
TIMER1_CH3	PA3		PB11	

2. For different packages and flash sizes please refer to the datasheet.

**Table 7.6. TIMER2 alternate function remapping**

Alternate function	TIMER2_REMAP [1:0] = "00" (no remap)	TIMER2_REMAP [1:0] = "10" (partial remap)	TIMER2_REMAP [1:0] = "11" (full remap) <sup>(1)</sup>
TIMER2_CH0	PA6	PB4	PC6
TIMER2_CH1	PA7	PB5	PC7
TIMER2_CH2	PB0		PC8
TIMER2_CH3	PB1		PC9

3. For different packages and flash sizes please refer to the datasheet.

**Table 7.7. TIMER3 alternate function remapping**

Alternate function	TIMER3_REMAP = 0	TIMER3_REMAP = 1 <sup>(1)</sup>
TIMER3_CH0	PB6	PD12
TIMER3_CH1	PB7	PD13
TIMER3_CH2	PB8	PD14
TIMER3_CH3	PB9	PD15

4. For different packages and flash sizes please refer to the datasheet.

**Table 7.8. TIMER4 alternate function remapping**

Alternate function	TIMER4CH3_REMAP = 0	TIMER4CH3_REMAP = 1
TIMER4_CH3	TIMER4_CH3 is connected to PA3	IRC40K internal clock is connected to TIMER4_CH3 input for calibration purpose

5. For different packages and flash sizes please refer to the datasheet.

#### 7.4.5. USART AF remapping

Refer to AFIO port configuration register 0 (AFIO\_PCF0).

**Table 7.9. USART0 alternate function remapping**

Alternate function	USART0_REMAP = 0	USART0_REMAP = 1
USART0_TX	PA9	PB6
USART0_RX	PA10	PB7

**Table 7.10. USART1 alternate function remapping**

Alternate function	USART1_REMAP = 0	USART1_REMAP = 1 <sup>(2)</sup>
USART1_CTS	PA0	PD3
USART1_RTS	PA1	PD4
USART1_TX	PA2	PD5
USART1_RX	PA3	PD6
USART1_CK	PA4	PD7

**Table 7.11. USART2 alternate function remapping**

Alternate function	USART2_REMAP [1:0] = “00” (no remap)	USART2_REMAP [1:0] =“01” (partial remap) <sup>(1)</sup>	USART2_REMAP [1:0] =“11” (full remap) <sup>(2)</sup>
USART2_TX	PB10	PC10	PD8
USART2_RX	PB11	PC11	PD9
USART2_CK	PB12	PC12	PD10
USART2_CTS	PB13		PD11
USART2_RTS	PB14		PD12

6. Remap available only for 64-pin,100-pin

7. Remap available only 100-pin

#### 7.4.6. I2C0 AF remapping

Refer to AFIO port configuration register 0 (AFIO\_PCF0).

**Table 7.12. I2C0 alternate function remapping**

Alternate function	I2C0_REMAP = 0	I2C0_REMAP = 1 <sup>(1)</sup>
I2C0_SCL	PB6	PB8

I2C0_SDA	PB7	PB9
----------	-----	-----

1. Remap not available on 36-pin package.

#### 7.4.7. SPI0 AF remapping

Refer to AFIO port configuration register 0 (AFIO\_PCF0).

**Table 7.13. SPI0 alternate function remapping**

Alternate function	SPI0_REMAP = 0	SPI0_REMAP = 1
SPI0_NSS	PA4	PA15
SPI0_SCK	PA5	PB3
SPI0_MISO	PA6	PB4
SPI0_MOSI	PA7	PB5

#### 7.4.8. SPI2/I2S2 AF remapping

Refer to AFIO port configuration register 0 (AFIO\_PCF0).

**Table 7.14. SPI2/I2S2 alternate function remapping**

Alternate function	SPI2_REMAP = 0	SPI2_REMAP = 1
SPI2_NSS/ I2S2_WS	PA15	PA4
SPI2_SCK/ I2S2_CK	PB3	PC10
SPI2_MISO	PB4	PC11
SPI2_MOSI/I2S2_SD	PB5	PC12

#### 7.4.9. CAN0 AF remapping

Refer to AFIO port configuration register 0 (AFIO\_PCF0).

**Table 7.15. CAN0 alternate function remapping**

Alternate function	CAN0_REMAP[1:0] =“00”	CAN0_REMAP[1:0] =“10” <sup>(2)</sup>	CAN0_REMAP[1:0] =“11” <sup>(1)</sup>
CAN0_RX	PA11	PB8	PD0
CAN0_TX	PA12	PB9	PD1

1. This remapping is available only on 100-pin packages, when PD0 and PD1 are not remapped on OSC\_IN and OSC\_OUT.
2. Remap not available on 36-pin package.

#### 7.4.10. CAN1 AF remapping

Refer to AFIO port configuration register 0 (AFIO\_PCF0).

**Table 7.16. CAN1 alternate function remapping**

1. Remap not available on 36-pin package.

Alternate function	CAN1_REMAP = "0" <sup>(1)</sup>	CAN1_REMAP = "1"
CAN1_RX	PB12	PB5
CAN1_TX	PB13	PB6

#### 7.4.11. CLK pins AF remapping

The LXTAL oscillator pins OSC32\_IN and OSC32\_OUT can be used as general-purpose I/O PC14 and PC15 individually, when the LXTAL oscillator is off. The LXTAL has priority over the GPIOs function.

- Note:** 1. But when the 1.8 V domain is powered off (by entering standby mode) or when the backup domain is supplied by  $V_{BAT}$  ( $V_{DD}$  no more supplied), the PC14/PC15 GPIO functionality is lost and will be set in analog mode.  
 2. Refer to the note on IO usage restrictions in Section [Battery backup domain](#).

**Table 7.17. OSC32 pins configuration**

Alternate function	LXTAL= ON	LXTAL= OFF
PC14	OSC32_IN	PC14
PC15	OSC32_OUT	PC15

The HXTAL oscillator pins OSC\_IN/OSC\_OUT can be used as general-purpose I/O PD0/PD1.

**Table 7.18. OSC pins configuration**

Alternate function	HXTAL= ON	HXTAL = OFF
PD0	OSC_IN	PD0
PD1	OSC_OUT	PD1

## 7.5. Register definition

GPIOA base address: 0x4001 0800

GPIOB base address: 0x4001 0C00

GPIOC base address: 0x4001 1000

GPIOF base address: 0x4001 1400

GPIOE base address: 0x4001 1800

### 7.5.1. Port control register 0 (GPIOx\_CTL0, x=A..E)

Address offset: 0x00

Reset value: 0x4444 4444

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
CTL7[1:0]		MD7[1:0]		CTL6[1:0]		MD6[1:0]		CTL5[1:0]		MD5[1:0]		CTL4[1:0]		MD4[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
CTL3[1:0]		MD3[1:0]		CTL2[1:0]		MD2[1:0]		CTL1[1:0]		MD1[1:0]		CTL0[1:0]		MD0[1:0]	
rw		rw		rw		rw		rw		rw		rw		rw	

Bits	Fields	Descriptions
31:30	CTL7[1:0]	Port 7 configuration bits These bits are set and cleared by software refer to CTL0[1:0]description
29:28	MD7[1:0]	Port 7 mode bits These bits are set and cleared by software refer to MD0[1:0]description
27:26	CTL6[1:0]	Port 6 configuration bits These bits are set and cleared by software refer to CTL0[1:0]description
25:24	MD6[1:0]	Port 6 mode bits These bits are set and cleared by software refer to MD0[1:0]description
23:22	CTL5[1:0]	Port 5 configuration bits These bits are set and cleared by software refer to CTL0[1:0]description
21:20	MD5[1:0]	Port 5 mode bits These bits are set and cleared by software refer to MD0[1:0]description

---

19:18	CTL4[1:0]	Port 4 configuration bits These bits are set and cleared by software refer to CTL0[1:0]description
17:16	MD4[1:0]	Port 4 mode bits These bits are set and cleared by software refer to MD0[1:0]description
15:14	CTL3[1:0]	Port 3 configuration bits These bits are set and cleared by software refer to CTL0[1:0]description
13:12	MD3[1:0]	Port 3 mode bits These bits are set and cleared by software refer to MD0[1:0]description
11:10	CTL2[1:0]	Port 2 configuration bits These bits are set and cleared by software refer to CTL0[1:0]description
9:8	MD2[1:0]	Port 2 mode bits These bits are set and cleared by software refer to MD0[1:0]description
7:6	CTL1[1:0]	Port 1 configuration bits These bits are set and cleared by software refer to CTL0[1:0]description
5:4	MD1[1:0]	Port 1 mode bits These bits are set and cleared by software refer to MD0[1:0]description
3:2	CTL0[1:0]	Port 0 configuration bits These bits are set and cleared by software Input mode ( MD[1:0] =00) 00: Analog mode 01: Floating input 10: Input with pull-up / pull-down 11: Reserved  Output mode ( MD[1:0] >00) 00: GPIO output with push-pull 01: GPIO output with open-drain 10: AFIO output with push-pull 11: AFIO output with open-drain
1:0	MD0[1:0]	Port 0 mode bits These bits are set and cleared by software

- 00: Input mode (reset state)
- 01: Output mode ,max speed 10MHz
- 10: Output mode ,max speed 2 MHz
- 11: Output mode ,max speed 50MHz

### 7.5.2. Port control register 1 (GPIOx\_CTL1, x=A..E)

Address offset: 0x04

Reset value: 0x4444 4444

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
CTL15[1:0]		MD15[1:0]		CTL14[1:0]		MD14[1:0]		CTL13[1:0]		MD13[1:0]		CTL12[1:0]		MD12[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
CTL11[1:0]		MD11[1:0]		CTL10[1:0]		MD10[1:0]		CTL9[1:0]		MD9[1:0]		CTL8[1:0]		MD8[1:0]	
rw		rw		rw		rw		rw		rw		rw		rw	

Bits	Fields	Descriptions
31:30	CTL15[1:0]	Port 15 configuration bits These bits are set and cleared by software refer to CTL0[1:0]description
29:28	MD15[1:0]	Port 15 mode bits These bits are set and cleared by software refer to MD0[1:0]description
27:26	CTL14[1:0]	Port 14 configuration bits These bits are set and cleared by software refer to CTL0[1:0]description
25:24	MD14[1:0]	Port 14 mode bits These bits are set and cleared by software refer to MD0[1:0]description
23:22	CTL13[1:0]	Port 13 configuration bits These bits are set and cleared by software refer to CTL0[1:0]description
21:20	MD13[1:0]	Port 13 mode bits These bits are set and cleared by software refer to MD0[1:0]description
19:18	CTL12[1:0]	Port 12 configuration bits These bits are set and cleared by software refer to CTL0[1:0]description

17:16	MD12[1:0]	Port 12 mode bits These bits are set and cleared by software refer to MD0[1:0]description
15:14	CTL11[1:0]	Port 11 configuration bits These bits are set and cleared by software refer to CTL0[1:0]description
13:12	MD11[1:0]	Port 11 mode bits These bits are set and cleared by software refer to MD0[1:0]description
11:10	CTL10[1:0]	Port 10 configuration bits These bits are set and cleared by software refer to CTL0[1:0]description
9:8	MD10[1:0]	Port 10 mode bits These bits are set and cleared by software refer to MD0[1:0]description
7:6	CTL9[1:0]	Port 9 configuration bits These bits are set and cleared by software refer to CTL0[1:0]description
5:4	MD9[1:0]	Port 9 mode bits These bits are set and cleared by software refer to MD0[1:0]description
3:2	CTL8[1:0]	Port 8 configuration bits These bits are set and cleared by software refer to CTL0[1:0]description
1:0	MD8[1:0]	Port 8 mode bits These bits are set and cleared by software refer to MD0[1:0]description

### 7.5.3. Port input status register (GPIOx\_ISTAT, x=A..E)

Address offset: 0x08

Reset value: 0x0000 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
Reserved															
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 r r

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

#### 7.5.4. Port output control register (GPIO<sub>x</sub>\_OCTL, x=A..E)

Address offset: 0x0C

Reset value: 0x0000 0000

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

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

### 7.5.5. Port bit operate register (GPIOx\_BOP, x=A..E)

Address offset: 0x10

Reset value: 0x0000 0000

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

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
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 to 0
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 to 1

### 7.5.6. Port bit clear register (GPIOx\_BC, x=A..E)

Address offset: 0x14

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
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

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
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 to 0

### 7.5.7. Port configuration lock register (GPIOx\_LOCK, x=A..E)

Address offset: 0x18

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
LK15	LK14	LK13	LK12	LK11	LK10	LK9	LK8	LK7	LK6	LK5	LK4	LK3	LK2	LK1	LK0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

### 7.5.8. Event control register (AFIO\_EC)

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								EOE	PORT[2:0]				PIN[3:0]		
								rw							

Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value
7	EOE	<p>Event output enable</p> <p>Set and cleared by software. When set the EVENTOUT RISC-V output is connected to the I/O selected by the PORT[2:0] and PIN[3:0] bits</p>
6:4	PORT[2:0]	<p>Event output port selection</p> <p>Set and cleared by software. Select the port used to output the RISC-V EVENTOUT signal.</p> <p>000: Select PORT A 001: Select PORT B 010: Select PORT C 011: Select PORT D</p>

100: Select PORT E

3:0	PIN[3:0]	Event output pin selection Set and cleared by software. Select the pin used to output the RISC-V EVENTOUT signal.
		0000: Select Pin 0
		0001: Select Pin 1
		0010: Select Pin 2
		...
		1111: Select Pin 15

### 7.5.9. AFIO port configuration register 0 (AFIO\_PCF0)

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	TIMER1_ITI1_REMAP_AP	SPI2_REMAP			SWJ_CFG[2:0]	Reserved	CAN1_REMAP								TIMER4C_H3_IREMAP
	rw	rw			w		rw								rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PD01_REMAP	CAN0_REMAP[1:0]	TIMER3_REMAP	TIMER2_REMAP[1:0]	TIMER1_REMAP[1:0]	TIMER0_REMAP[1:0]	USART2_REMAP[1:0]	USART1_REMAP	USART0_REMAP	I2C0_REMAP	SPI0_REMAP					
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw					rw

Bits	Fields	Descriptions
31:28	Reserved	Must be kept at reset value
29	TIMER1_ITI1_REMAP	TIMER1 internal trigger 1 remapping These bits are set and cleared by software. It control the TIMER1_ITI1 internal mapping 0: Connect to 0 1: Connect USB OTG SOF (Start of Frame) output TIMER1_ITI1 for calibration purposes
28	SPI2_REMAP	SPI2/I2S2 remapping This bit is set and cleared by software. 0: No remap (SPI2_NSS-I2S2_WS/PA15, SPI2_SCK-I2S2_CK/PB3, SPI2_MISO/PB4, SPI2_MOSI-I2S_SD/PB5) 1: Full remap (SPI2_NSS-I2S2_WS/PA4, SPI2_SCK-I2S2_CK/PC10, SPI2_MISO/PC11, SPI2_MOSI-I2S_SD/PC12)

Note: This bit is available only in Extra-density devices and High-density devices.

27	Reserved	Must be kept at reset value
26:24	SWJ_CFG[2:0]	<p>Serial wire JTAG configuration</p> <p>These bits are write-only (when read, the value is undefined). They are used to configure the SWJ alternate function I/Os. The SWJ (Serial Wire JTAG) supports JTAG access to the RISC-V debug port. The default state after reset is SWJ ON. This allows JTAG mode to be enabled by sending a specific sequence on the JTMS/JTCK pin.</p> <p>000: JTAG-DP Reset State</p> <p>001: JTAG-DP but without NJTRST</p> <p>010: JTAG-DP Disabled</p> <p>Other: Undefined</p>
23	Reserved	Must be kept at reset value
22	CAN1_REMAP	<p>CAN1 interface remapping</p> <p>This bit is set and cleared by software</p> <p>0: No remap (CAN1_RX/PB12, CAN1_TX/PB13)</p> <p>1: Remap (CAN1_RX/PB5, CAN1_TX/PB6)</p>
21:17	Reserved	Must be kept at reset value
16	TIMER4CH3_IREMA	<p>TIMER4 channel3 internal remapping</p> <p>P</p> <p>Set and cleared by software. This bit controls the TIMER4_CH3 internal mapping. When reset the timer TIMER4_CH3 is connected to PA3. When set the IRC40K internal clock is connected to TIMER4_CH3 input for calibration purpose.</p> <p>Note: This bit is available only in High-density value line devices.</p>
15	PD01_REMAP	<p>Port D0/Port D1 mapping on OSC_IN/OSC_OUT</p> <p>This bit is set and cleared by software</p> <p>0: Not remap</p> <p>1: PD0 remapped on OSC_IN, PD1 remapped on OSC_OUT</p>
14:13	CAN0_REMAP [1:0]	<p>CAN0 interface remapping</p> <p>These bits are set and cleared by software.</p> <p>00: No remap (CAN0_RX/PA11, CAN0_TX/PA12)</p> <p>01: Not used</p> <p>10: Partial remap (CAN0_RX/PB8, CAN0_TX/PB9)</p> <p>11: Full remap (CAN0_RX/PD0, CAN0_TX/PD1)</p>
12	TIMER3_REMAP	<p>TIMER3 remapping</p> <p>This bit is set and cleared by software</p> <p>0: No remap (TIMER3_CH0/PB6, TIMER3_CH1/PB7, TIMER3_CH2/PB8, TIMER3_CH3/PB9)</p> <p>1: Full remap (TIMER3_CH0/PD12, TIMER3_CH1/PD13, TIMER3_CH2/PD14, TIMER3_CH3/PD15)</p>
11:10	TIMER2_	TIMER2 remapping

	<b>REMAP[1:0]</b>	These bits are set and cleared by software 00: No remap (TIMER2_CH0/PA6,TIMER2_CH1/PA7,TIMER2_CH2/PB0, TIMER2_CH3/PB1) 01: Not used 10: Partial remap (TIMER2_CH0/PB4,TIMER2_CH1/PB5,TIMER2_CH2/PB0, TIMER2_CH3/PB1) 11: Full remap (TIMER2_CH0/PC6,TIMER2_CH1/PC7,TIMER2_CH2/PC8, TIMER2_CH3/PC9)
9:8	<b>TIMER1_REMAP</b> [1:0]	TIMER1 remapping These bits are set and cleared by software 00: No remap (TIMER1_CH0-TIMER1_ETI/PA0,TIMER1_CH1/PA1, TIMER1_CH2/PA2,TIMER1_CH3/PA3) 01: Partial remap (TIMER1_CH0-TIMER1_ETI/PA15,TIMER1_CH1/PB3, TIMER1_CH2/PA2,TIMER1_CH3/PA3) 10: Partial remap (TIMER1_CH0-TIMER1_ETI/PA0,TIMER1_CH1/PA1, TIMER1_CH2/PB10,TIMER1_CH3/PB11) 11: Full remap(TIMER1_CH0-TIMER1_ETI/PA15,TIMER1_CH1/PB3, TIMER1_CH2/PB10,TIMER1_CH3/PB11)
7:6	<b>TIMER0_REMAP</b> [1:0]	TIMER0 remapping These bits are set and cleared by software 00: No remap (TIMER0_ETI/PA12, TIMER0_CH0/ PA8, TIMER0_CH1/PA9, TIMER0_CH2/PA10,TIMER0_CH3/PA11,TIMER0_BKIN/PB12, TIMER0_CH0_ON/PB13, TIMER0_CH1_ON/PB14, TIMER0_CH2_ON/PB15) 01: Partial remap (TIMER0_ETI/PA12, TIMER0_CH0/ PA8, TIMER0_CH1/PA9, TIMER0_CH2/PA10,TIMER0_CH3/PA11,                            TIMER0_BKIN/PA6, TIMER0_CH0_ON/PA7, TIMER0_CH1_ON/PB0, TIMER0_CH2_ON/PB1) 10: Not used 11: Full remap (TIMER0_ETI/PE7, TIMER0_CH0/ PE9, TIMER0_CH1/PE11, TIMER0_CH2/PE13,TIMER0_CH3/PE14,                            TIMER0_BKIN/PE15, TIMER0_CH0_ON/PE8, TIMER0_CH1_ON/PE10, TIMER0_CH2_ON/PE12)
5:4	<b>USART2_REMAP</b> [1:0]	USART2 remapping These bits are set and cleared by software 00: No remap (USART2_TX/PB10, USART2_RX /PB11, USART2_CK/PB12, USART2_CTS/PB13, USART2_RTS/PB14) 01: Partial remap (USART2_TX/PC10, USART2_RX /PC11, USART2_CK/PC12, USART2_CTS/PB13, USART2_RTS/PB14) 10: Not used 11: Full remap (USART2_TX/PD8, USART2_RX /PD9, USART2_CK/PD10, USART2_CTS/PD11, USART2_RTS/PD12)
3	<b>USART1_REMAP</b>	USART1 remapping This bit is set and cleared by software 0: No remap (USART1_CTS/PA0, USART1_RTS/PA1,USART1_TX/PA2,

		USART1_RX /PA3, USART1_CK/PA4)
	1:	Remap (USART1_CTS/PD3, USART1_RTS/PD4, USART1_TX/PD5, USART1_RX /PD6, USART1_CK/PD7)
2	USART0_REMAP	USART0 remapping  This bit is set and cleared by software  0: No remap (USART0_TX/PA9, USART0_RX /PA10) 1: Remap (USART0_TX/PB6, USART0_RX /PB7)
1	I2C0_REMAP	I2C0 remapping  This bit is set and cleared by software  0: No remap (I2C0_SCL/PB6, I2C0_SDA /PB7) 1: Remap (I2C0_SCL/PB8, I2C0_SDA /PB9)
0	SPI0_REMAP	SPI0 remapping  This bit is set and cleared by software  0: No remap (SPI0_NSS/PA4, SPI0_SCK /PA5, SPI0_MISO /PA6, SPI0_MOSI /PA7) 1: Remap (SPI0_NSS/PA15, SPI0_SCK /PB3, SPI0_MISO /PB4, SPI0_MOSI /PB5)

### 7.5.10. EXTI sources selection register 0 (AFIO\_EXTISS0)

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															
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  0000: PA3 pin 0001: PB3 pin 0010: PC3 pin 0011: PD3 pin 0100: PE3 pin Other configurations are reserved.
11:8	EXTI2_SS [3:0]	EXTI 2 sources selection

0000: PA2 pin  
 0001: PB2 pin  
 0010: PC2 pin  
 0011: PD2 pin  
 0100: PE2 pin  
 Other configurations are reserved.

7:4	EXTI1_SS [3:0]	EXTI 1 sources selection
		0000: PA1 pin
		0001: PB1 pin
		0010: PC1 pin
		0011: PD1 pin
		0100: PE1 pin
		Other configurations are reserved.
3:0	EXTI0_SS [3:0]	EXTI 0 sources selection
		0000: PA0 pin
		0001: PB0 pin
		0010: PC0 pin
		0011: PD0 pin
		0100: PE0 pin
		Other configurations are reserved.

### 7.5.11. EXTI sources selection register 1 (AFIO\_EXTI\_SS1)

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															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI7_SS[3:0]		EXTI6_SS[3:0]		EXTI5_SS[3:0]		EXTI4_SS[3:0]									
rw		rw		rw		rw		rw		rw		rw		rw	

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:12	EXTI7_SS [3:0]	EXTI 7 sources selection 0000: PA7 pin 0001: PB7 pin 0010: PC7 pin 0011: PD7 pin 0100: PE7 pin

Other configurations are reserved.

11:8	EXTI6_SS [3:0]	EXTI 6 sources selection 0000: PA6 pin 0001: PB6 pin 0010: PC6 pin 0011: PD6 pin 0100: PE6 pin Other configurations are reserved.
7:4	EXTI5_SS [3:0]	EXTI 5 sources selection 0000: PA5 pin 0001: PB5 pin 0010: PC5 pin 0011: PD5 pin 0100: PE5 pin Other configurations are reserved.
3:0	EXTI4_SS [3:0]	EXTI 4 sources selection 0000: PA4 pin 0001: PB4 pin 0010: PC4 pin 0011: PD4 pin 0100: PE4 pin Other configurations are reserved.

### 7.5.12. EXTI sources selection register 2 (AFIO\_EXTI\_S2)

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															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI11_SS[3:0]				EXTI10_SS[3:0]				EXTI9_SS[3:0]				EXTI8_SS[3:0]			
rw				rw				rw				rw			

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:12	EXTI11_SS [3:0]	EXTI 11 sources selection 0000: PA11 pin 0001: PB11 pin

		0010: PC11 pin 0011: PD11 pin 0100: PE11 pin Other configurations are reserved.
11:8	EXTI10_SS [3:0]	EXTI 10 sources selection 0000: PA10 pin 0001: PB10 pin 0010: PC10 pin 0011: PD10 pin 0100: PE10 pin Other configurations are reserved.
7:4	EXTI9_SS [3:0]	EXTI 9 sources selection 0000: PA9 pin 0001: PB9 pin 0010: PC9 pin 0011: PD9 pin 0100: PE9 pin Other configurations are reserved.
3:0	EXTI8_SS [3:0]	EXTI 8 sources selection 0000: PA8 pin 0001: PB8 pin 0010: PC8 pin 0011: PD8 pin 0100: PE8 pin Other configurations are reserved.

### 7.5.13. EXTI sources selection register 3 (AFIO\_EXTI\_SS3)

Address offset: 0x14

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
EXTI15_SS[3:0]		EXTI14_SS[3:0]		EXTI13_SS[3:0]		EXTI12_SS[3:0]									

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value

15:12	EXTI15_SS [3:0]	EXTI 15 sources selection 0000: PA15 pin 0001: PB15 pin 0010: PC15 pin 0011: PD15 pin 0100: PE15 pin Other configurations are reserved.
11:8	EXTI14_SS [3:0]	EXTI 14 sources selection 0000: PA14 pin 0001: PB14 pin 0010: PC14 pin 0011: PD14 pin 0100: PE14 pin Other configurations are reserved.
7:4	EXTI13_SS [3:0]	EXTI 13 sources selection 0000: PA13 pin 0001: PB13 pin 0010: PC13 pin 0011: PD13 pin 0100: PE13 pin Other configurations are reserved.
3:0	EXTI12_SS [3:0]	EXTI 12 sources selection 0000: PA12 pin 0001: PB12 pin 0010: PC12 pin 0011: PD12 pin 0100: PE12 pin Other configurations are reserved.

#### 7.5.14. AFIO port configuration register 1 (AFIO\_PCF1)

Address offset: 0x1C

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					EXMC_N	Reserved					Reserved				

rw

Bits	Fields	Descriptions
31:11	Reserved	Must be kept at reset value
10	EXMC_NADV	<p>EXMC_NADV connect/disconnect</p> <p>This bit is set and cleared by software, it controls the use of optional EXMC_NADV signal.</p> <p>0: The NADV signal is connected to the output(default)</p> <p>1: The NADV signal is not connected. The I/O pin can be used by another peripheral.</p>
9:0	Reserved	Must be kept at reset value

## 8. CRC calculation unit (CRC)

### 8.1. 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.

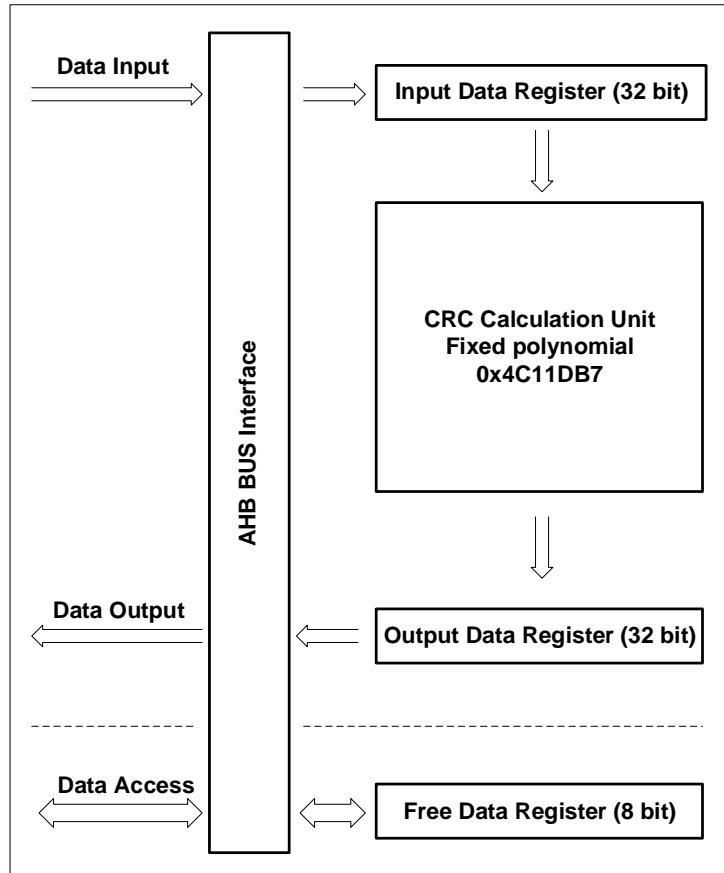
This CRC calculation unit can be used to calculate 32 bit CRC code with fixed polynomial.

### 8.2. Characteristics

- 32-bit data input and 32-bit data output. Calculation period is 4 AHB clock cycles for 32-bit input data size from data entered to the calculation result available.
- Free 8-bit register is unrelated to calculation and can be used for any other goals by any other peripheral devices.
- 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.

**Figure 8-1. Block diagram of CRC calculation unit**



### 8.3. 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 the CRC\_DATA register has not been cleared by software setting the CRC\_CTL register, the new input raw data will be calculated based on the result of previous value of CRC\_DATA.

CRC calculation will spend 4 AHB clock cycles for 32-bit data size, during this period AHB will not be hanged because of the existence of the 32-bit 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.

## 8.4. Register definition

CRC base address: 0x4002 3000

### 8.4.1. Data register (CRC\_DATA)

Address offset: 0x00

Reset value: 0xFFFF FFFF

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
DATA [31:16]															
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA [15:0]															
rw															

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

### 8.4.2. Free data register (CRC\_FDATA)

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

Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value.
7:0	FDATA [7:0]	Free Data Register bits Software writes and reads.

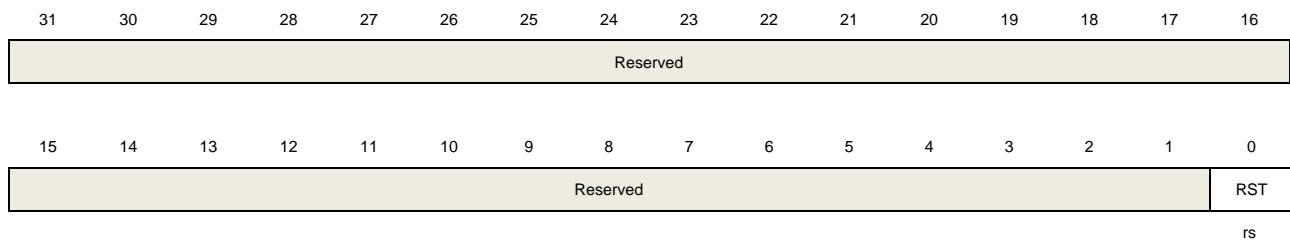
These bits are unrelated with CRC calculation. This byte can be used for any goal by any other peripheral. The CRC\_CTL register will take no effect to the byte.

### 8.4.3. Control register (CRC\_CTL)

Address offset: 0x08

Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31:1	Reserved	Must be kept at reset value.
0	RST	Set this bit can reset the CRC_DATA register to the value of 0xFFFFFFFF then automatically cleared itself to 0 by hardware. This bit will take no effect to CRC_FDATA. Software writes and reads.

## 9. Direct memory access controller (DMA)

### 9.1. 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 12 channels in the DMA controller (7 for DMA0 and 5 for DMA1). 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.

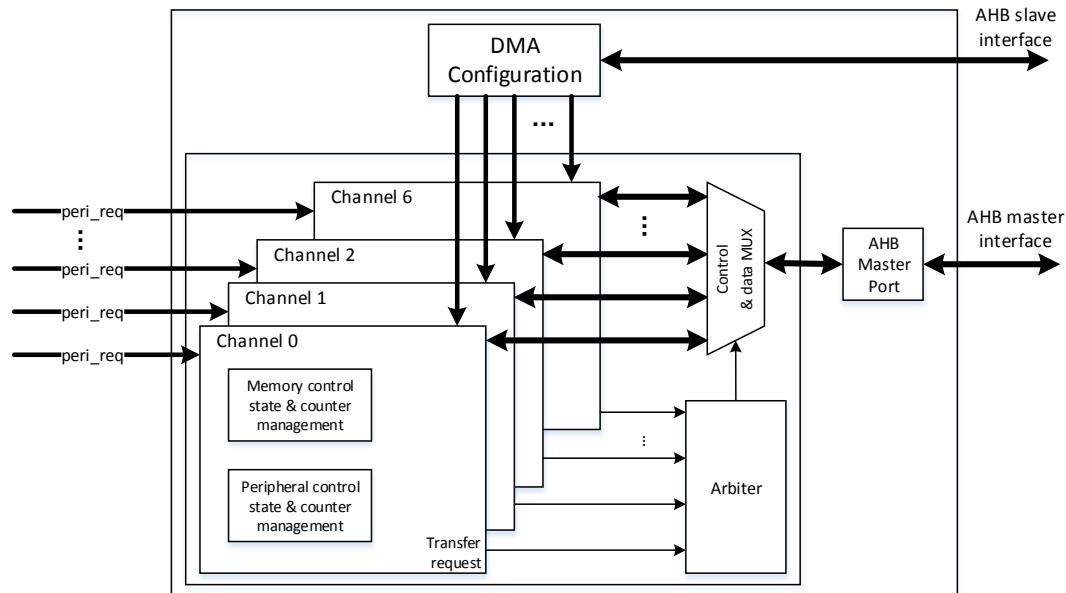
The system bus is shared by the DMA controller and the RISC-V core. When the DMA and the CPU are targeting the same destination, the DMA access may stop the CPU access to the system bus for some bus cycles. Round-robin scheduling is implemented in the bus matrix to ensure at least half of the system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

### 9.2. Characteristics

- Programmable length of data to be transferred, max to 65536.
- 12 channels and each channel are configurable (7 for DMA0 and 5 for DMA1).
- 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.

## 9.3. Block diagram

**Figure 9-1. Block diagram of DMA**



As shown in [Figure 9-1. 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 arbiter inside to manage multiple peripheral requests coming at the same time
- Channel management to control address/data selection and data counting

## 9.4. Function overview

### 9.4.1. 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 PWIDHT and MWIDHT 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 PWIDHT and MWIDHT are shown in the following [Table 9-1. DMA transfer operation](#).

Table 9-1. 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 BDDB[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 enabling 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.

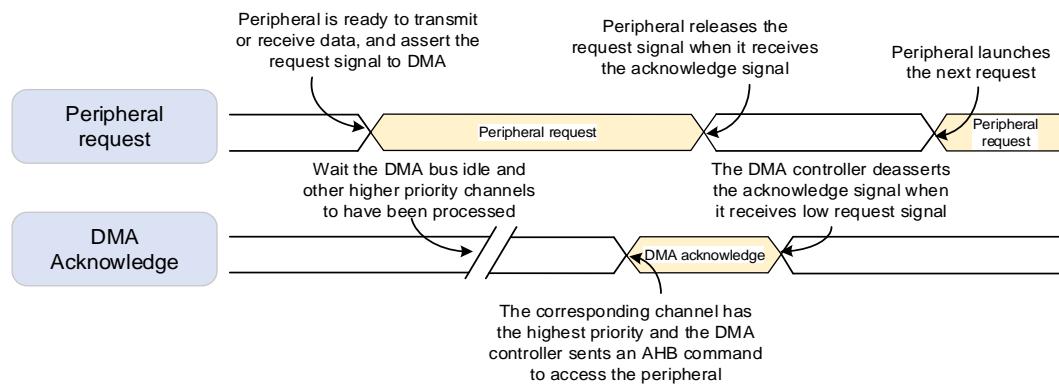
#### 9.4.2. Peripheral handshake

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 9-2. Handshake mechanism** shows how the handshake mechanism works between the DMA controller and peripherals.

**Figure 9-2. Handshake mechanism**



#### 9.4.3. 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.

#### 9.4.4. 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.

#### 9.4.5. 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 respond the peripheral request until the CHEN bit in the DMA\_CHxCTL register is cleared.

#### 9.4.6. 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.

#### 9.4.7. 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.

#### 9.4.8. 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.

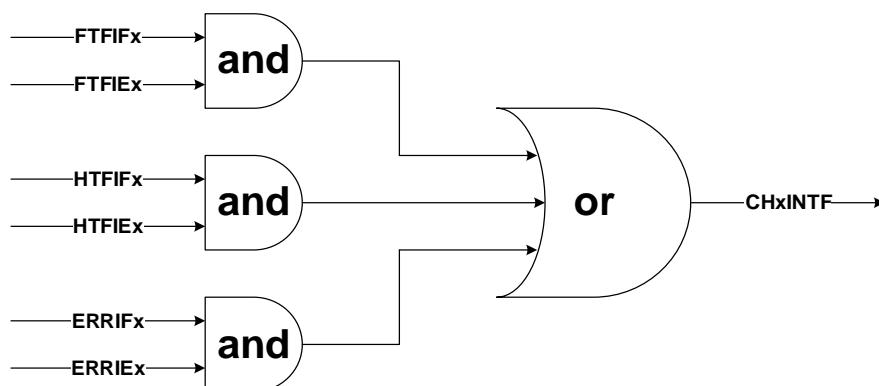
Each interrupt event has a dedicated flag bit in the DMA\_INTF register, a dedicated clear bit in the DMA\_INTC register, and a dedicated enable bit in the DMA\_CHxCTL register. The relationship is described in the following [Table 9-2. Interrupt events](#).

**Table 9-2. 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 9-3. DMA interrupt logic](#), an interrupt can be produced when any type of interrupt event occurs and enabled on the channel.

**Figure 9-3. DMA interrupt logic**



**Note:** "x" indicates channel number (for DMA0, x=0...6, for DMA1, x=0...4).

### 9.4.9. 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 9-4. DMA0 request mapping](#) and [Figure 9-5. DMA1 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 9-3. DMA0 requests for each channel](#) lists the support request from peripheral for each channel of DMA0, and [Table 9-4. DMA1 requests for each channel](#) lists the support request from peripheral for each channel of DMA1.

**Figure 9-4. DMA0 request mapping**

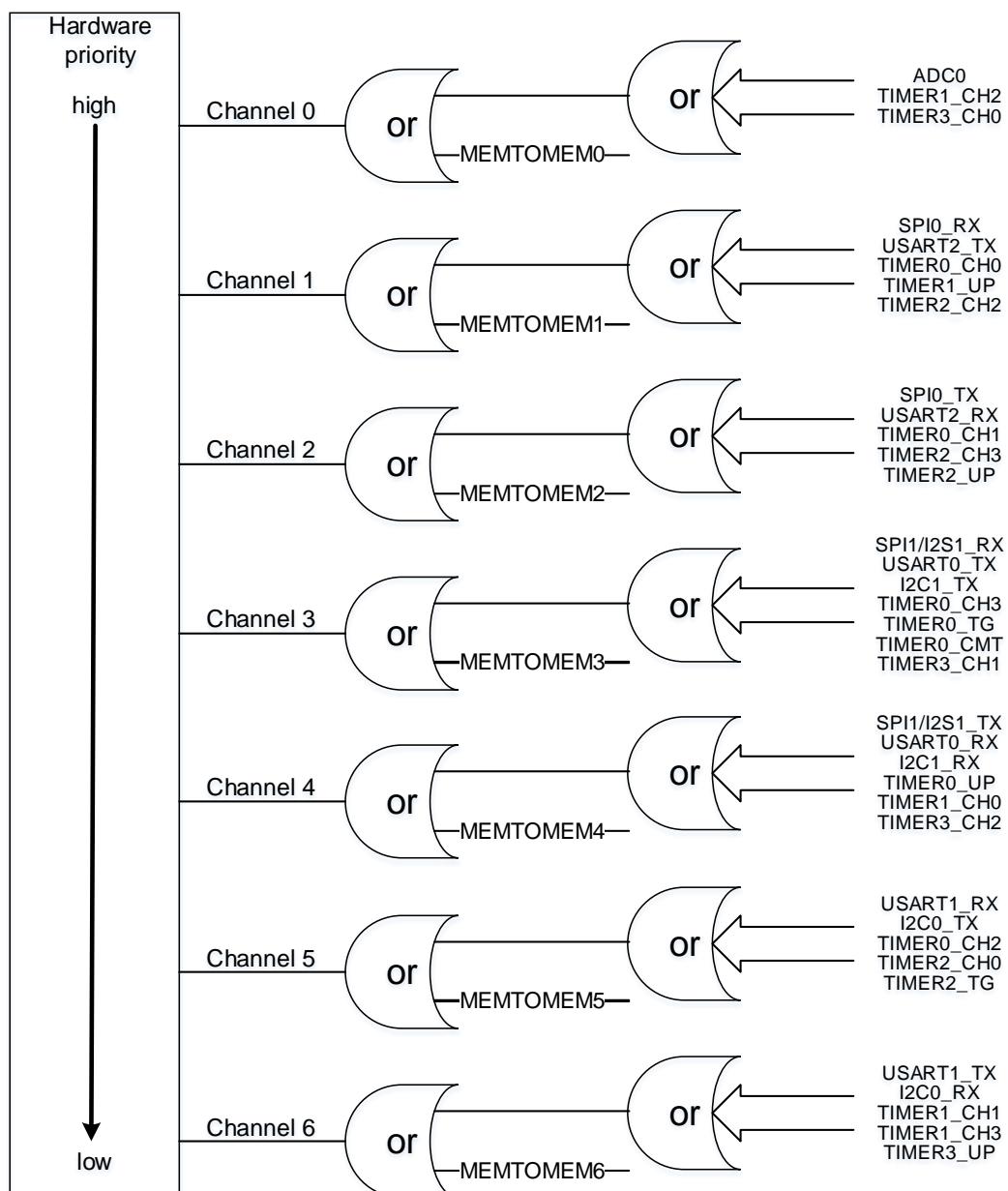


Table 9-3. DMA0 requests for each channel

Peripheral	Channel 0	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5	Channel 6
TIMER0	•	TIMER0_CH0	TIMER0_CH1	TIMER0_CH3 TIMER0_TG TIMER0_CM T	TIMER0_UP	TIMER0_CH2	•
TIMER1	TIMER1_CH2	TIMER1_UP	•	•	TIMER1_CH0	•	TIMER1_CH1 TIMER1_CH3
TIMER2	•	TIMER2_CH2	TIMER2_CH3 TIMER2_UP	•	•	TIMER2_CH0 TIMER2_TG	•
TIMER3	TIMER3_CH0	•	•	TIMER3_CH1	TIMER3_CH2	•	TIMER3_UP
ADC0	ADC0	•	•	•	•	•	•
SPI/I2S	•	SPI0_RX	SPI0_TX	SPI1/I2S1_R X	SPI1/I2S1_T X	•	•
USART	•	USART2_TX	USART2_RX	USART0_TX	USART0_RX	USART1_RX	USART1_TX
I2C	•	•	•	I2C1_TX	I2C1_RX	I2C0_RX	I2C0_RX

Figure 9-5. DMA1 request mapping

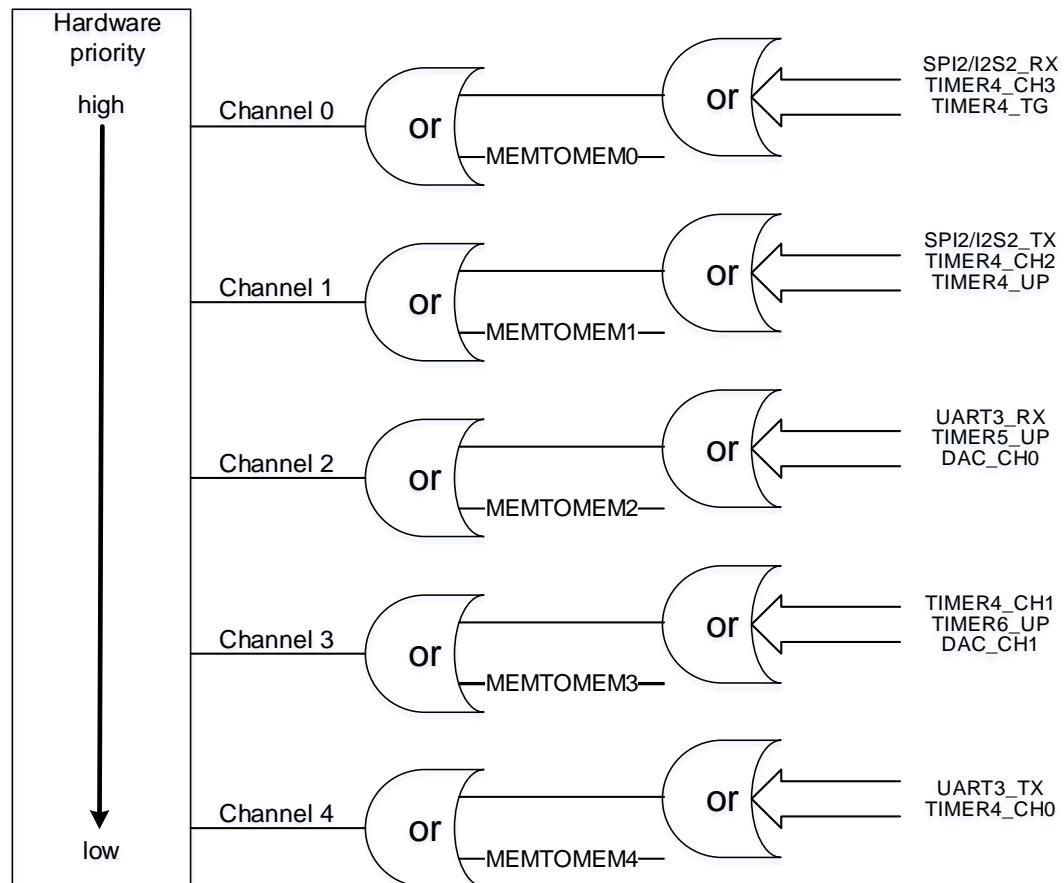


Table 9-4. DMA1 requests for each channel

Peripheral	Channel 0	Channel 1	Channel 2	Channel 3	Channel 4
<b>TIMER4</b>	TIMER4_CH3	TIMER4_CH2	•	TIMER4_CH1	TIMER4_CH0
	TIMER4_TG	TIMER4_UP			
<b>TIMER5</b>	•	•	TIMER5_UP	•	•
<b>TIMER6</b>	•	•	•	TIMER6_UP	•
<b>DAC</b>	•	•	DAC_CH0	DAC_CH1	•
<b>SPI/I2S</b>	SPI2/I2S2_RX	SPI2/I2S2_TX	•	•	•
<b>USART</b>	•	•	UART3_RX	•	UART3_TX

## 9.5. Register definition

DMA0 base address: 0x4002 0000

DMA1 base address: 0x4002 0400

**Note:** For DMA1 having 5 channels, all bits related to channel 5 and channel 6 in the following registers are not suitable for DMA1.

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

Address offset: 0x00

Reset value: 0x0000 0000

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	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/	ERRIFx	Error flag of channel x (x=0...6)
15/11/7/3		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/	HTFIFx	Half transfer finish flag of channel x (x=0...6)
14/10/6/2		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/	FTFIFx	Full Transfer finish flag of channel x (x=0...6)
13/9/5/1		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/	GIFx	Global interrupt flag of channel x (x=0...6)
12/8/4/0		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

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

Address offset: 0x04

Reset value: 0x0000 0000

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	HTFIC2	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/	ERRIFCx	Clear bit for error flag of channel x (x=0...6)
15/11/7/3		0: No effect 1: Clear error flag
26/22/18/	HTFIFCx	Clear bit for half transfer finish flag of channel x (x=0...6)
14/10/6/2		0: No effect 1: Clear half transfer finish flag
25/21/17/	FTFIFCx	Clear bit for full transfer finish flag of channel x (x=0...6)
13/9/5/1		0: No effect 1: Clear full transfer finish flag
24/20/16/	GIFCx	Clear global interrupt flag of channel x (x=0...6)
12/8/4/0		0: No effect 1: Clear GIFx, ERRIFx, HTFIFx and FTFIFx bits in the DMA_INTF register

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

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

Address offset: 0x08 + 0x14 × x

Reset value: 0x0000 0000

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	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	PRIO[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

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

$x = 0 \dots 6$ , where  $x$  is a channel number

Address offset:  $0x0C + 0x14 \times x$

Reset value: 0x0000 0000

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

rw

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.

---

15:0	CNT[15:0]	<p>Transfer counter</p> <p>These bits can not be written when CHEN in the DMA_CHxCTL register is '1'.</p> <p>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.</p>
------	-----------	--

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

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

Address offset: 0x10 + 0x14 × x

Reset value: 0x0000 0000

**Note:** Do not configure this register when channel is enabled.

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

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

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

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

Address offset: 0x14 + 0x14 × x

Reset value: 0x0000 0000

**Note:** Do not configure this register when channel is enabled.

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

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

rw

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>

## 10. Debug (DBG)

### 10.1. Overview

The GD32VF103 series provide a large variety of debug and test features. They are implemented with a standard configuration of the RISC-V module together with a daisy chained standard TAP controller. Debug functions are integrated into the RISC-V. The debug system supports standard JTAG debug. The debug refer to the following documents:

- RISC-V External Debug Support Version 0.13

The DBG hold unit helps debugger to debug power saving mode, TIMER, I2C, WWDGT, FWDGT and CAN. When corresponding bit is set, provide clock when in power saving mode or hold the state for TIMER, WWDGT, FWDGT, I2C or CAN.

### 10.2. JTAG function overview

Debug capabilities can be accessed by a debug tool via JTAG interface (JTAG - Debug Port).

#### 10.2.1. Pin assignment

The JTAG interface provides 5-pin standard JTAG, known as JTAG clock (JTCK), JTAG mode selection (JTMS), JTAG data input (JTDI), JTAG data output (JTDO) and JTAG reset (NJTRST, active low).

The pin assignment are:

PA15: JTDI

PA14: JTCK

PA13: JTMS

PB4: NJTRST

PB3: JTDO

By default, 5-pin standard JTAG debug mode is chosen after reset. Users can also use JTAG function without NJTRST pin, then the PB4 can be used to other GPIO functions. (NJTRST tied to 1 by hardware). If JTAG not used, all 5-pin can be released to other GPIO functions. Please refer to [GPIO pin configuration](#).

#### 10.2.2. JTAG daisy chained structure

The RISC-V JTAG TAP is connected to a Boundary-Scan (BSD) JTAG TAP. The BSD JTAG IR is 5-bit width, while the RISC-V JTAG IR is 4-bit width. So when JTAG in IR shift step, it

first shift 5-bit BYPASS instruction (5'b 11111) for BSD JTAG, and then shift normal 4-bit instruction for RISC-V JTAG. Because of the data shift under BSD JTAG BYPASS mode, adding 1 extra bit to the data chain is needed.

The BSD JTAG IDCODE is 0x790007A3.

### 10.2.3. Debug reset

The JTAG-DP register are in the power on reset domain. The System reset initializes the majority of the RISC-V. The NJTRST reset can reset JTAG TAP controller only.

## 10.3. Debug hold function overview

### 10.3.1. Debug support for power saving mode

When STB\_HOLD bit in DBG control register (DBG\_CTL) 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 DSLP\_HOLD bit in DBG control register (DBG\_CTL) 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 (DBG\_CTL) 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.

### 10.3.2. Debug support for TIMER, I2C, WWDGT, FWDGT and CAN

When the core halted and the corresponding bit in DBG control register (DBG\_CTL) is set, the following behaved.

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

For I2C, SMBUS timeout hold for debug.

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

For CAN, the receive register stopped counting for debug.

## 10.4. Register definition

DBG base address: 0xE004 2000

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

Address: 0xE004 2000

Read 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
ID_CODE[31:16]															
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID_CODE[15:0]															
r															

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

### 10.4.2. Control register (DBG\_CTL)

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															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I2C0_HO	CAN0_H	TIMER3_H	TIMER2_H	TIMER1_H	TIMER0_H	WWDGT_H	FWDGT_H	Reserved				STB_H	DSLP_H	SLP_H	I2C1_HO
LD	OLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	Reserved				HOLD	HOLD	HOLD	LD
rw	rw	rw	rw	rw	rw	rw	rw	rw				rw	rw	rw	rw

Bits	Fields	Descriptions
31:22	Reserved	Must be kept at reset value
21	CAN1_HOLD	CAN1 hold bit This bit is set and reset by software 0: no effect

		1: the receive register of CAN1 stops receiving data when core halted
20	TIMER6_HOLD	<p>TIMER 6 hold bit</p> <p>This bit is set and reset by software</p> <p>0: no effect</p> <p>1: hold the TIMER 6 counter for debug when core halted</p>
19	TIMER5_HOLD	<p>TIMER 5 hold bit</p> <p>This bit is set and reset by software</p> <p>0: no effect</p> <p>1: hold the TIMER 5 counter for debug when core halted</p>
18	TIMER4_HOLD	<p>TIMER 4 hold bit</p> <p>This bit is set and reset by software</p> <p>0: no effect</p> <p>1: hold the TIMER 4 counter for debug when core halted</p>
17	Reserved	Must be kept at reset value
16	I2C1_HOLD	<p>I2C1 hold bit</p> <p>This bit is set and reset by software</p> <p>0: no effect</p> <p>1: hold the I2C1 SMBUS timeout for debug when core halted</p>
15	I2C0_HOLD	<p>I2C0 hold bit</p> <p>This bit is set and reset by software</p> <p>0: no effect</p> <p>1: hold the I2C0 SMBUS timeout for debug when core halted</p>
14	CAN0_HOLD	<p>CAN0 hold bit</p> <p>This bit is set and reset by software</p> <p>0: no effect</p> <p>1: the receive register of CAN0 stops receiving data when core halted</p>
13	TIMER3_HOLD	<p>TIMER 3 hold bit</p> <p>This bit is set and reset by software</p> <p>0: no effect</p> <p>1: hold the TIMER 3 counter for debug when core halted</p>
12	TIMER2_HOLD	<p>TIMER 2 hold bit</p> <p>This bit is set and reset by software</p> <p>0: no effect</p> <p>1: hold the TIMER 2 counter for debug when core halted</p>
11	TIMER1_HOLD	<p>TIMER 1 hold bit</p> <p>This bit is set and reset by software</p> <p>0: no effect</p> <p>1: hold the TIMER 1 counter for debug when core halted</p>

---

10	TIMER0_HOLD	TIMER 0 hold bit  This bit is set and reset by software 0: no effect 1: hold the TIMER 0 counter for debug when core halted
9	WWDGT_HOLD	WWDGT hold bit  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 bit  This bit is set and reset by software 0: no effect 1: hold the FWDGT counter clock for debug when core halted
7:3	Reserved	Must be kept at reset value
2	STB_HOLD	Standby mode hold register  This bit is set and reset by software 0: no effect 1: At the standby mode, the clock of AHB bus and system clock are provided by CK_IRC8M, a system reset generated when exit 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 clock of AHB bus and system clock 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 clock of AHB is on.

## 11. Analog-to-digital converter (ADC)

### 11.1. Introduction

The 12-bit ADC is an analog-to-digital converter using the successive approximation method. It has 18 multiplexed channels making the ADC convert analog signals from 16 external channels, and 2 internal channels. The analog watchdog allows the application to detect whether the input voltage goes outside the user-defined higher or lower thresholds. The analog signals of the channels can be converted by the ADC 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.

### 11.2. Main features

- High performance
  - 12-bit, 10-bit, 8-bit or 6-bit configurable resolution
  - ADC sampling rate: 2 MSPs for 12-bit resolution
  - Self-calibration
  - Programmable sampling time
  - Data alignment with built-in data coherency
  - DMA support
- Analog input channels
  - 16 external analog inputs
  - 1 channel for internal temperature sensor ( $V_{SENSE}$ )
  - 1 channel for internal reference voltage ( $V_{REFINT}$ )
- Start-of-conversion can be initiated
  - By software
  - By hardware triggers
- Conversion modes
  - Converts a single channel or scans a sequence of channels.
  - Single mode converts selected inputs once per trigger.
  - Continuous mode converts selected inputs continuously
  - Discontinuous mode
  - SYNC mode(the device with two or more ADCs)
- Analog watchdog
- Interrupt generation:
  - at the end of regular and inserted group conversions
  - analog watchdog event
- Oversampler
  - 16-bit data register

- Oversampling ratio adjustable from 2 to 256x
- Programmable data shift up to 8-bit
- ADC supply requirements: 2.6V to 3.6V, and typical power supply voltage is 3.3V
- ADC input range:  $V_{REF-} \leq V_{IN} \leq V_{REF+}$

### 11.3. Pins and internal signals

[\*\*Figure 11-1. ADC module block diagram\*\*](#) shows the ADC block diagram. [\*\*Table 11-1. ADC internal signals\*\*](#) gives the ADC internal signals and [\*\*Table 11-2. ADC pins definition\*\*](#) gives the ADC pin description.

**Table 11-1. ADC internal signals**

Internal signal name	Signal type	Description
$V_{SENSE}$	Input	Internal temperature sensor output voltage
$V_{REFINT}$	Input	Internal voltage reference output voltage

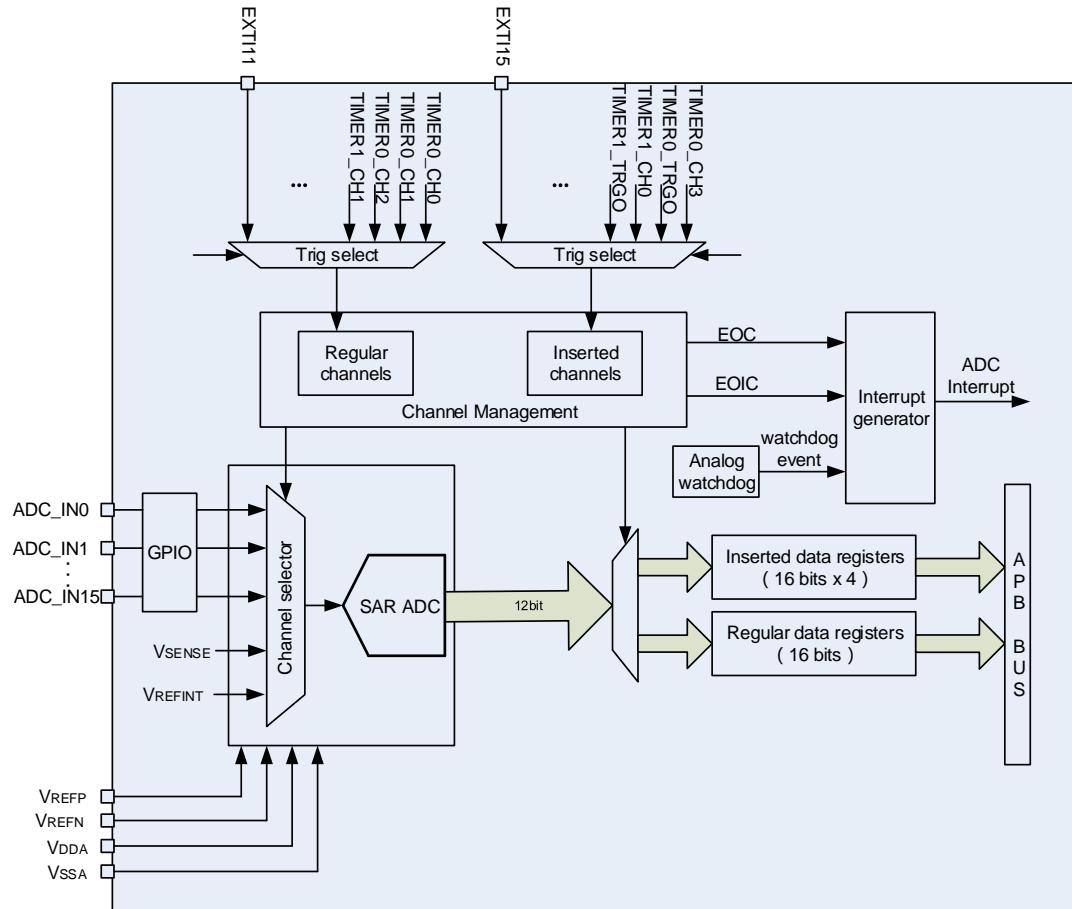
**Table 11-2. ADC pins definition**

Name	Signal type	Remarks
$V_{DDA}$	Input, analog power supply	Analog power supply equal to $V_{DD}$ and $2.6 \text{ V} \leq V_{DDA} \leq 3.6 \text{ V}$
$V_{SSA}$	Input, analog power supply ground	Ground for analog power supply equal to $V_{SS}$
$V_{REF+}$	Input, positive reference voltage	The positive reference voltage for the ADC, $2.4 \text{ V} \leq V_{REF+} \leq V_{DDA}$
$V_{REF-}$	Input, negative reference voltage	The negative reference voltage for the ADC, $V_{REF-} = V_{SSA}$
$ADCx\_IN[15:0]$	Input, Analog signals	Up to 16 external channels

**Note:** The  $ADC\_IN[15:0]$  should set as Analog Input mode.

## 11.4. Functional description

Figure 11-1. ADC module block diagram



### 11.4.1. Calibration (CLB)

The ADC has a foreground 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. The calibration is initiated by software by setting bit CLB=1. CLB bit stays at 1 during all the calibration sequence. It is then cleared by hardware as soon as the calibration is completed.

When the ADC operating conditions change (such as supply power voltage  $V_{DDA}$ , positive reference voltage  $V_{REF+}$ , temperature and so on), 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.

#### **11.4.2. ADC clock**

The ADCCLK clock provided by the clock controller is synchronous APB2 clock. The RCU controller has a dedicated programmable prescaler for the ADC clock.

#### **11.4.3. ADCON switch**

The ADCON bit on 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 sub-module will be put into power-down mode.

#### **11.4.4. Regular and inserted channel groups**

The ADC supports 18 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.

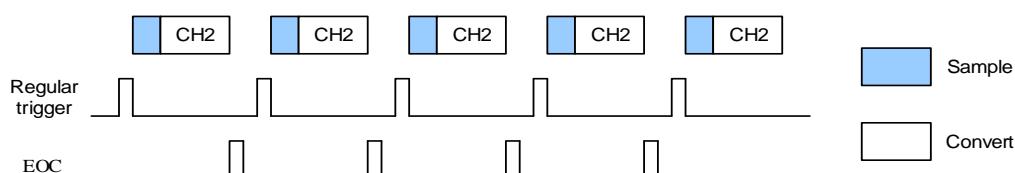
In the inserted group, a sequence of up to 4 conversions can be organized in a specific sequence. The ADC\_ISQ register specify the selected channels of the inserted group. The IL[1:0] bits in the ADC\_ISQ register specify the total conversion sequence length.

#### **11.4.5. 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 11-2. 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 inserted 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 in the ADC\_RDATA register
8. Clear the EOC flag by writing 0 to it

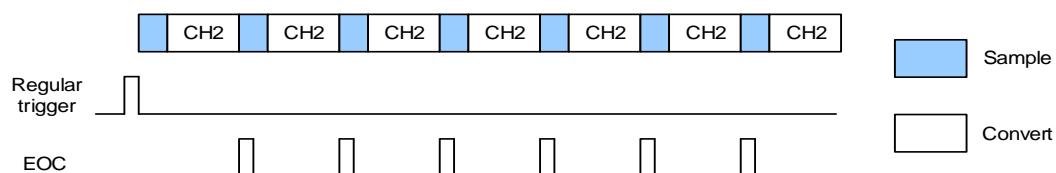
Software procedure for a single conversion of an inserted channel:

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 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 11-3. Continuous conversion mode**



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 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

To get rid of checking, DMA can be used to transfer the converted data:

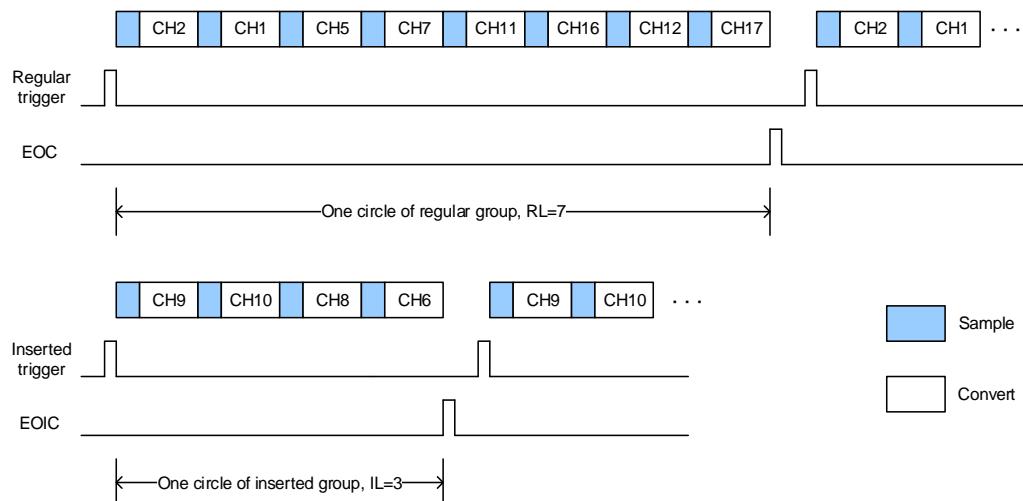
1. Set the CTN and DMA bit in the ADC\_CTL1 register
2. Configure RSQ0 with the analog channel number
3. Configure ADC\_SAMPTx register
4. Configure ETERC and ETSRC bits in the ADC\_CTL1 register if in need
5. Prepare the DMA module to transfer data from the ADC\_RDATA.
6. Set the SWRCST bit, or generate 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 11-4. Scan conversion mode, continuous disable**



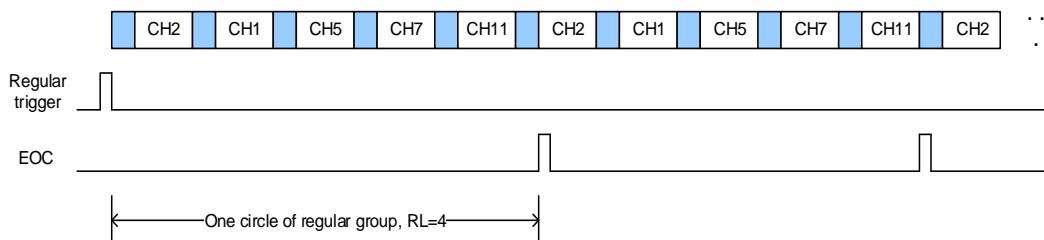
Software procedure for scan conversion on a regular channel group:

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.
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 in the ADC\_IDATAx register
7. Clear the EOC/EOIC flag by writing 0 to them

**Figure 11-5. Scan conversion mode, continuous enable**



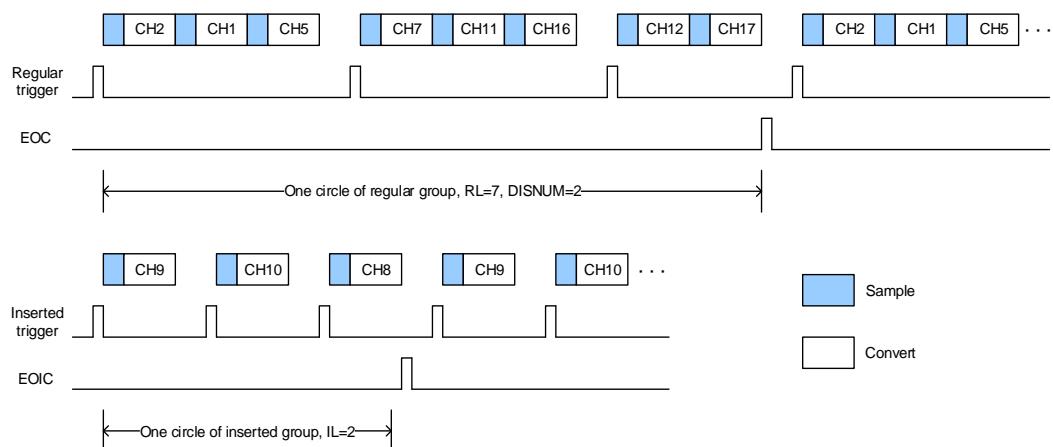
### Discontinuous mode

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 EOICIE 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 11-6. 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 in the ADC\_IDATAx register
8. Clear the EOC/EOIC flag by writing 0 to them

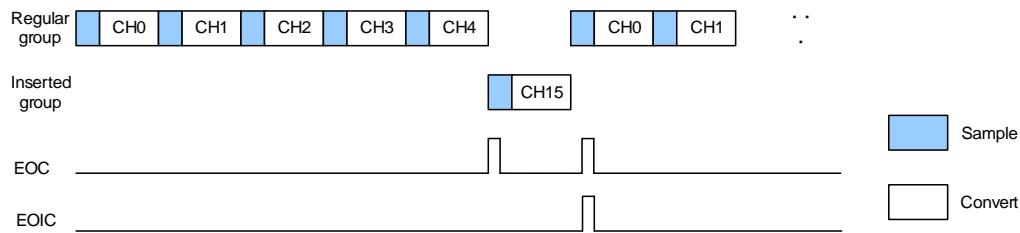
## 11.4.6. Inserted channel management

### 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 11-7. Auto-insertion, CNT = 1**

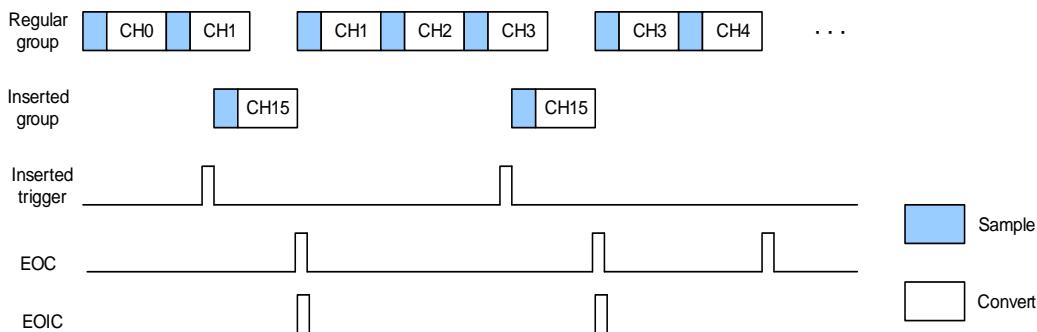


The auto insertion mode cannot be enabled when the discontinuous conversion mode is set.

### 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. After the inserted channel group is done, the regular group channel conversion is resumed from the last aborted conversion.

**Figure 11-8. Triggered insertion**



### Analog watchdog

The analog watchdog is enabled when the RWDEN and IWDEN 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, which are selected by the RWDEN, IWDEN, WDSC and WDCHSEL[4:0] bits in ADC\_CTL0 register, can be monitored by the analog watchdog.

#### 11.4.7. Data alignment

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

register.

After being 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.

**Figure 11-9. 12-bit Data alignment**

Regular group data
0 0 0 0 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0
Inserted group data
Sign Sign Sign Sign D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0
DAL=0
Regular group data
D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0 0 0 0 0
Inserted group data
Sign D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0 0 0 0
DAL=1

6-bit resolution data alignment is different from 12-bit/10-bit/8-bit resolution data alignment, shown as [Figure 11-10. 6-bit Data alignment](#).

**Figure 11-10. 6-bit Data alignment**

Regular group data
0 0 0 0 0 0 0 0 0 0 D5 D4 D3 D2 D1 D0
Inserted group data
Sign Sign Sign Sign Sign Sign Sign Sign Sign D5 D4 D3 D2 D1 D0
DAL=0
Regular group data
0 0 0 0 0 0 0 0 0 D5 D4 D3 D2 D1 D0 0 0
Inserted group data
Sign Sign Sign Sign Sign Sign Sign Sign D5 D4 D3 D2 D1 D0 0
DAL=1

#### 11.4.8. Programmable sample time

The number of ADCCLK 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. For 12-bits resolution, the total conversion time is “sampling time + 12.5” ADCCLK cycles.

Example:

ADCCLK = 10MHz and sample time is 1.5 cycles, the total conversion time is “1.5+12.5” ADCCLK cycles, that means 1.4us.

### 11.4.9. 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.

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 11-3. External trigger for regular channels for ADC0 and ADC1**

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	TIMER3_CH3	
110	EXTI11	
111	SWRCST	Software trigger

**Table 11-4. External trigger for inserted channels for ADC0 and ADC1**

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	TIMER3_TRGO	
110	EXTI15	
111	SWICST	Software trigger

### 11.4.10. DMA request

The DMA request, which is enabled by the DMA bit of ADC\_CTL1 register, is used to transfer data of regular group for conversion of more than one channel. 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.

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

When the TSVREN bit of ADC\_CTL1 register is set, the temperature sensor channel (ADC0\_CH16) and  $V_{REFINT}$  channel (ADC0\_CH17) 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 at least 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, the internal temperature sensor is more suited for applications that detect temperature variations instead of absolute temperature. When it is used to detect accurate temperature, an external temperature sensor part should be used to calibrate the offset error.

The internal voltage reference ( $V_{REFINT}$ ) provides a stable (bandgap) voltage output for the ADC and Comparators.  $V_{REFINT}$  is internally connected to the ADC0\_CH17 input channel.

To use the temperature sensor:

1. Configure the conversion sequence (ADC\_IN16) and the sampling time(17.1 $\mu$ s) 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( $V_{temperature}$ ) in the ADC data register, and get the temperature using the following formula:

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

$V_{25}$ :  $V_{temperature}$  value at 25°C, the typical value please refer to the datasheet.

Avg\_Slope: Average Slope for curve between Temperature vs.  $V_{temperature}$ , the typical value please refer to the datasheet.

#### 11.4.12. Programmable resolution (DRES) - fast conversion mode

It is possible to obtain faster conversion time ( $t_{ADC}$ ) 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\_OVSAMPCTL 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. Lower resolution reduces the conversion time needed for the successive approximation steps as shown in [Table 11-5. t<sub>CONV</sub> timings depending on resolution](#).

**Table 11-5. t<sub>CONV</sub> timings depending on resolution**

DRES[1:0] bits	t <sub>CONV</sub> (ADC clock cycles)	t <sub>CONV(ns)</sub> at $f_{ADC}=14MHz$	t <sub>SMPL(min)</sub> (ADC clock cycles)	t <sub>ADC</sub> (ADC clock cycles)	t <sub>ADC(us)</sub> at $f_{ADC}=14MHz$
12	12.5	893 ns	1.5	14	1000 ns
10	10.5	750 ns	1.5	12	857 ns
8	8.5	607 ns	1.5	10	714 ns

DRES[1:0] bits	t <sub>CONV</sub> (ADC clock cycles)	t <sub>CONV(ns)</sub> at f <sub>ADC</sub> =14MHz	t <sub>SMPL(min)</sub> (ADC clock cycles)	t <sub>ADC</sub> (ADC clock cycles)	t <sub>ADC(us)</sub> at f <sub>ADC</sub> =14MHz
6	6.5	464 ns	1.5	8	571 ns

### 11.4.13. On-chip hardware oversampling

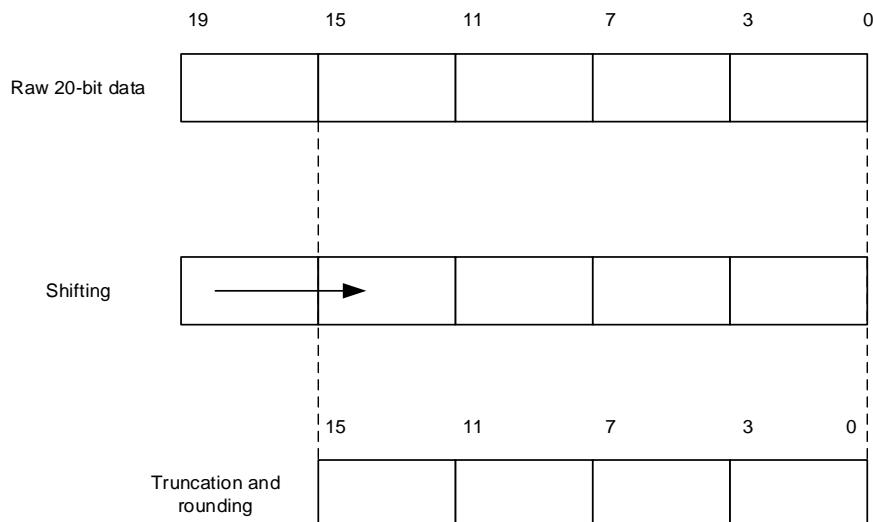
The on-chip hardware oversampling circuit performs data preprocessing to offload the CPU. It can handle multiple conversions and average them into a single data with increased data width, up to 16-bit. It provides a result with the following form, where N and M can be adjusted, and D<sub>out</sub>(n) is the n-th output digital signal of the ADC:

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

The on-chip hardware oversampling circuit performs the following functions: summing and bit right shifting. The oversampling ratio N is defined by the OVSR[2:0] bits in the ADC\_OVSAMPCTL register. It can range from 2x to 256x. The division coefficient M means bit right shifting up to 8-bit. 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 11-11. 20-bit to 16-bit result truncation**

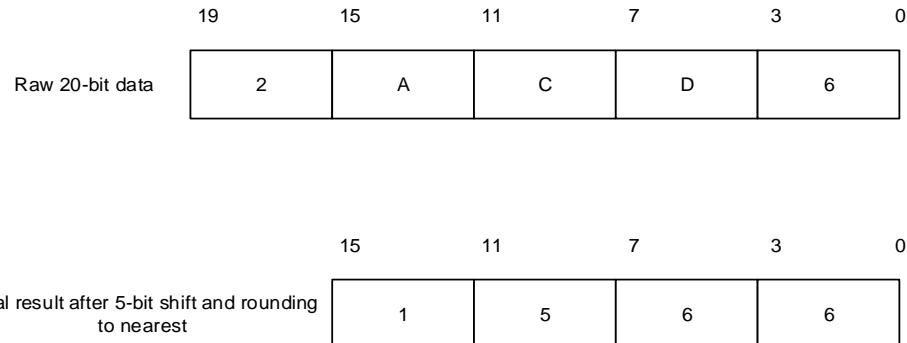


**Note:** If the intermediate result after the shifting exceeds 16 bits, the upper bits of the result are simply truncated.

[\*\*Figure 11-12. 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 11-12. Numerical example with 5-bits shift and rounding**



The [\*\*Table 11-6. Maximum output results vs N and M \(Grayed values indicates truncation\)\*\*](#) below gives the data format for the various N and M combination, for a raw conversion data equal to 0xFFFF.

**Table 11-6. 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	0xFFFF0	0xFFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100
32x	0x1FFE0	0xFFE0	0xFFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200
64x	0x3FFC0	0xFFC0	0xFFE0	0xFFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400
128x	0x7FF80	0xFF80	0xFFC0	0xFFE0	0xFFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800
256x	0xFFFF00	0xFF00	0xFF80	0xFFC0	0xFFE0	0xFFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF

The conversion timings in oversampled mode do not change compared to standard conversion mode: the sample time is maintained equal during the whole oversampling sequence. New data are provided every N conversion, with an equivalent delay equal to:

$$N \times t_{ADC} = N \times (t_{SMPL} + t_{CONV}) \quad (11-2)$$

## 11.5. ADC sync mode

In devices with two ADC, ADC sync mode can be used.

In ADC sync mode, the conversion starts alternately or simultaneously triggered by ADC0 master to ADC1 slave, according to the mode selected by the SYNC[3:0] bits in ADC1\_CTL0 register.

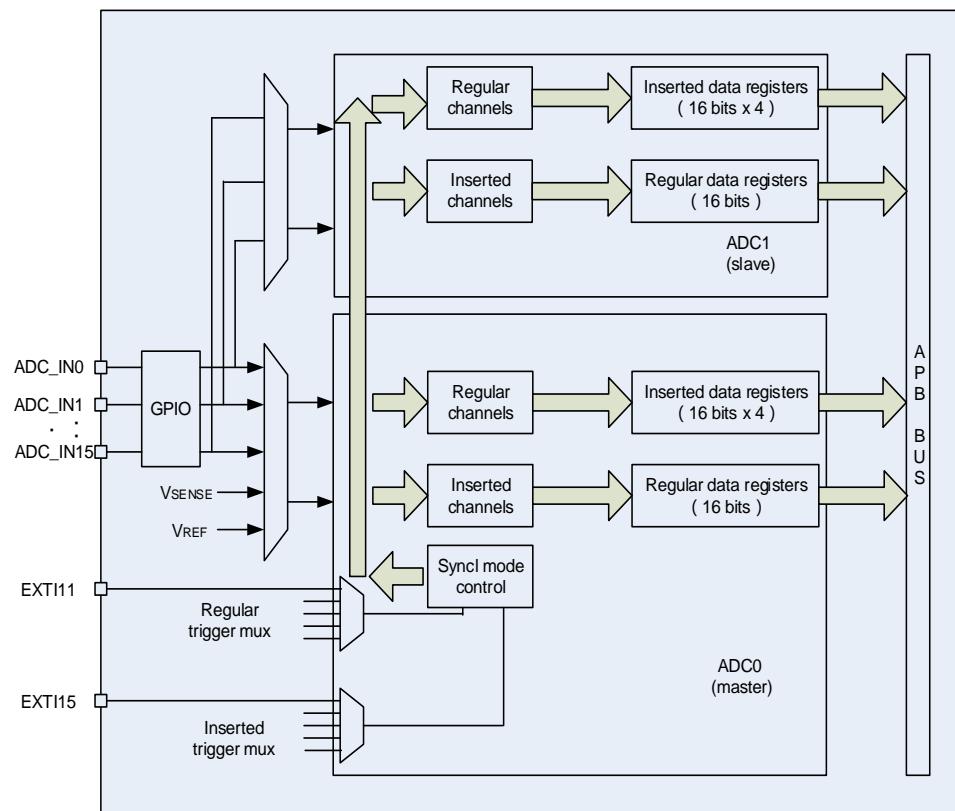
In sync mode, when configure the conversion which is triggered by an external event, the slave ADC must be configured as triggered by the software in order to prevent false triggers to start unwanted conversion. However, the external trigger must be enabled for ADC master and ADC slave.

The following modes can be configured:

- Free mode
- Regular parallel mode
- Inserted parallel mode
- Follow-up fast mode
- Follow-up slow mode
- Trigger rotation mode
- Inserted parallel mode + regular parallel mode
- Regular parallel mode + trigger rotation mode
- Inserted parallel mode + follow-up fast mode
- Inserted parallel mode + follow-up slow mode

In ADC sync mode, the DMA bit must be set even if it is not used; the converted data of ADC slave can be read from the master data register.

**Figure 11-13. ADC sync block diagram**



## 11.6. Free mode

In this mode, the ADC synchronization is bypassed, and each ADC works freely.

### 11.6.1. Regular parallel mode

This mode converts the regular channel simultaneously. The source of external trigger comes from the regular group MUX of ADC0 (selected by the ETSRC[2:0] bits in the ADC\_CTL1 register). A simultaneous trigger is provided to ADC1.

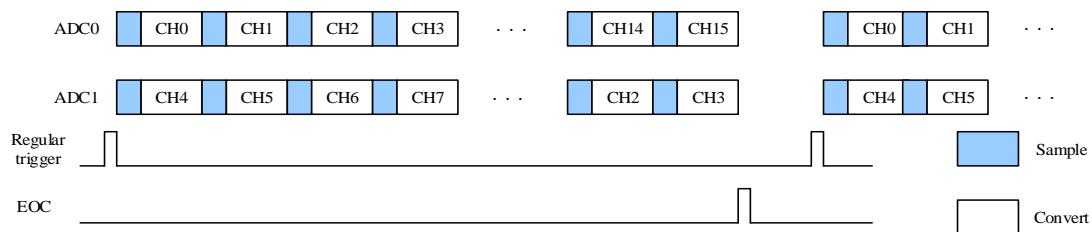
At the end of conversion event on ADC0 or ADC1 an EOC interrupt is generated (if enabled on one of the two ADC interfaces) when the ADC0/ADC1 regular channels are all converted. The behavior of regular parallel mode shows in the [Figure 11-14. Regular parallel mode on 16 channels](#).

A 32-bit DMA is used, which transfers ADC\_RDATA 32-bit register (the ADC\_RDATA 32-bit register containing the ADC1 converted data in the upper half-word and the ADC0 converted data in the lower half-word) to SRAM.

**Note:**

1. Do not convert the same channel on the two ADCs (no overlapping sampling times for the two ADCs when converting the same channel).
2. In parallel mode, exactly the same sampling time should be configured for the two channels that will be sampled simultaneously by ADC0 and ADC1.

**Figure 11-14. Regular parallel mode on 16 channels**



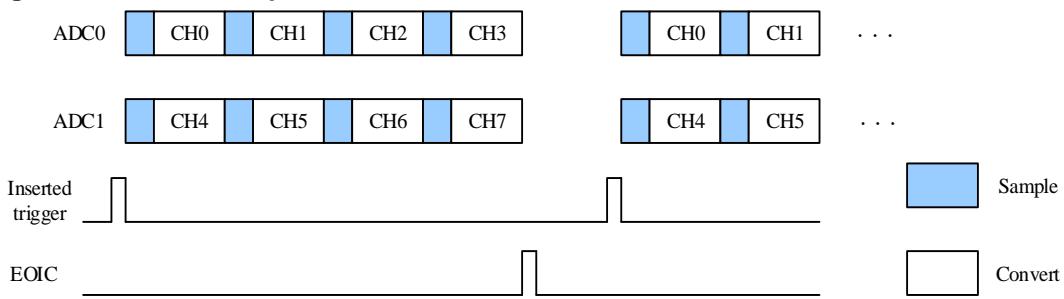
### 11.6.2. Inserted parallel mode

This mode converts the inserted channel simultaneously. The source of external trigger comes from the inserted group MUX of ADC0 (selected by the ETSIC[2:0] bits in the ADC\_CTL1 register). A simultaneous trigger is provided to ADC1.

At the end of conversion event on ADC0 or ADC1, an EOIC interrupt is generated (if enabled on one of the two ADC interfaces). ADC0/ADC1 inserted channels are all converted, and the converted data is stored in the ADC\_IDATAx registers of each ADC interface. The behavior of inserted parallel mode shows in the [Figure 11-15. Inserted parallel mode on 4 channels](#).

**Note:**

1. Do not convert the same channel on the two ADCs (no overlapping sampling times for the two ADCs when converting the same channel).
2. In parallel mode, exactly the same sampling time should be configured for the two channels that will be sampled simultaneously by ADC0 and ADC1.

**Figure 11-15. Inserted parallel mode on 4 channels**


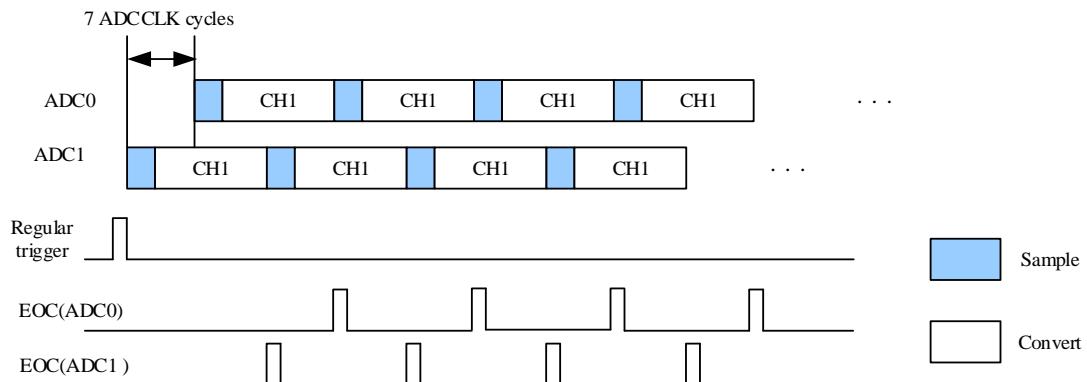
### 11.6.3. Follow-up fast mode

This mode can be running on the regular channel group (usually one channel). The source of external trigger comes from the regular channel MUX of ADC0 (selected by the ETSRC[2:0] bits in the ADC\_CTL1 register). When the trigger occurs, ADC1 runs immediately and ADC0 runs after 7 ADC clock cycles.

If the continuous mode is enabled for both ADC0 and ADC1, the selected regular channels of both ADCs are continuously converted. The behavior of follow-up fast mode shows in the [Figure 11-16. Follow-up fast mode on 1 channel in continuous conversion mode](#).

After an EOC interrupt is generated by ADC0 in case of setting the EOCIE bit, we can use a 32-bit DMA, which transfers to SRAM the ADC\_RDATA 32-bit register containing the ADC1 converted data in the upper half word and the ADC0 converted data in the lower half word.

**Note:** The maximum sampling time allowed is <7 ADCCLK cycles to avoid the overlap between ADC0 and ADC1 sampling phases in the event that they convert the same channel.

**Figure 11-16. Follow-up fast mode on 1 channel in continuous conversion mode**


### 11.6.4. Follow-up slow mode

This mode can be running on the regular channel group (usually one channel). The source of external trigger comes from the regular channel MUX of ADC0 (selected by the ETSRC[2:0] bits in the ADC\_CTL1 register). When the trigger occurs, ADC1 runs immediately, ADC0 runs after 7 ADC clock cycles.

after 14 ADC clock cycles, after the second 14 ADC clock cycles the ADC1 runs again.

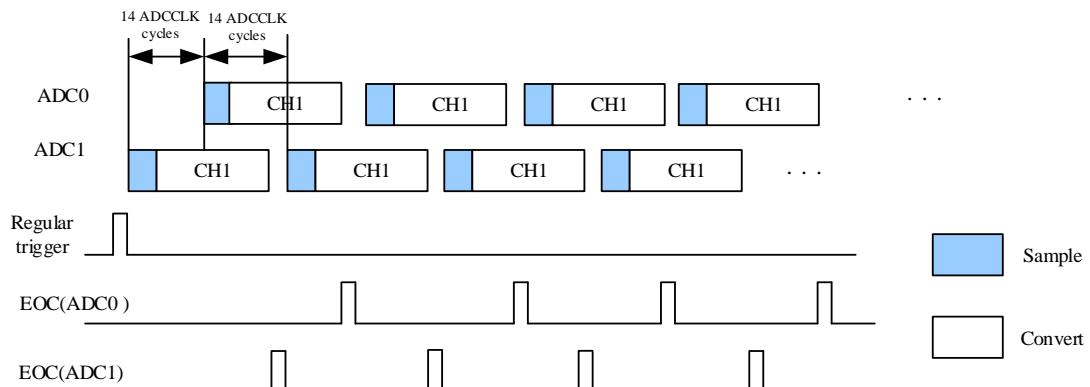
Continuous mode can't be used in this mode, because it continuously converts the regular channel. The behavior of follow-up slow mode shows in the [Figure 11-17. Follow-up slow mode on 1 channel](#).

After an EOC interrupt is generated by ADC0 (if enabled through the EOICIE bit), we can use a 32-bit DMA, which transfers to SRAM the ADC\_RDATA 32-bit register containing the ADC1 converted data in the upper half-word and the ADC0 converted data in the lower half-word.

**Note:**

1. The maximum sampling time allowed is <14 ADCCLK cycles to avoid the overlap between ADC0 and ADC1 sampling phases in the event that they convert the same channel.
2. For both the fast and follow-up slow mode, we must ensure that no external trigger for inserted channel occurs.

**Figure 11-17. Follow-up slow mode on 1 channel**



### 11.6.5. Trigger rotation mode

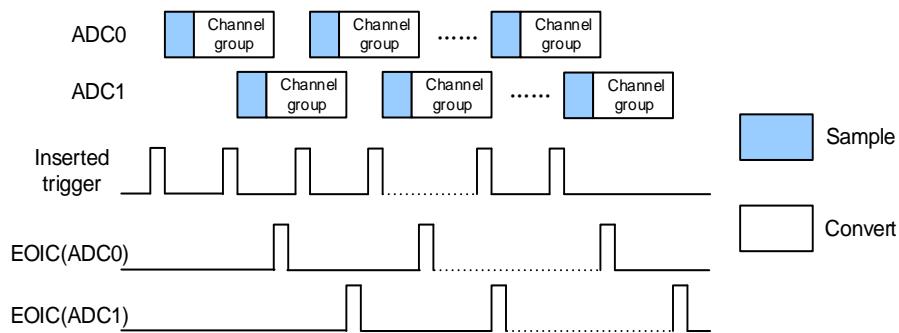
This mode can be running on the inserted channel group. The source of external trigger comes from the inserted channel MUX of ADC0 (selected by the ETSIC[2:0] bits in the ADC\_CTL1 register).

When the first trigger occurs, all the inserted channels of ADC0 are converted. When the second trigger occurs, all the inserted channels of ADC1 are converted. The behavior of trigger rotation mode shows in the [Figure 11-18. Trigger rotation: inserted channel group](#).

If the EOIC interrupt of ADC0 and ADC1 are enabled, when all the channels of ADC0 or ADC1 have been converted, the corresponded interrupt occurred.

If another external trigger occurs after all inserted group channels have been converted, the trigger rotation process restarts by converting ADC0 inserted group channels.

**Figure 11-18. Trigger rotation: inserted channel group**



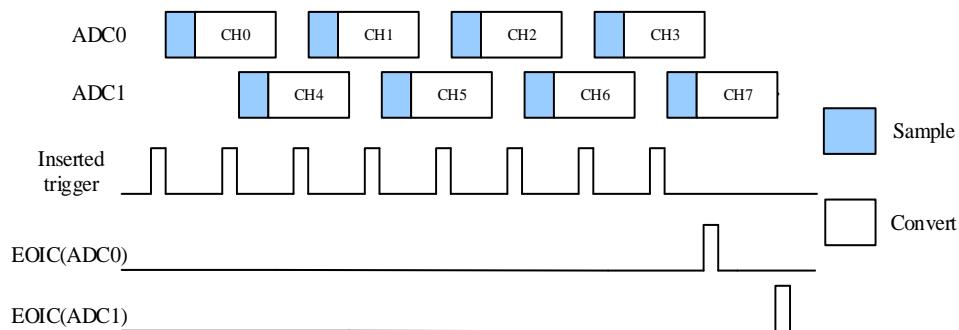
If the discontinuous mode is enabled for both ADC0 and ADC1, when the first trigger occurs, the first inserted channel in ADC0 is converted. When the second trigger occurs, the first inserted channel in ADC1 is converted. Then the second channel in ADC0, the second channel in ADC1, and so on.

The behavior of trigger rotation discontinuous mode shows in the [\*\*Figure 11-19. Trigger rotation: inserted channels in discontinuous mode\*\*](#).

If the EOIC interrupt of ADC0 and ADC1 are enabled. When all the channels of ADC0 or ADC1 have been converted, the corresponded interrupt occurred.

If another external trigger occurs after all inserted group channels have been converted then the trigger rotation process restarts.

**Figure 11-19. Trigger rotation: inserted channels in discontinuous mode**



### 11.6.6. Combined regular parallel & inserted parallel mode

In the free mode, the conversion of regular group can be interrupted by the conversion of inserted group. In the sync mode, it is also possible to interrupt parallel conversion of a regular group to insert parallel conversion of an inserted group.

**Note:** In combined regular parallel + inserted parallel mode, the sampling time for the two ADCs should be configured the same.

### 11.6.7. Combined regular parallel & trigger rotation mode

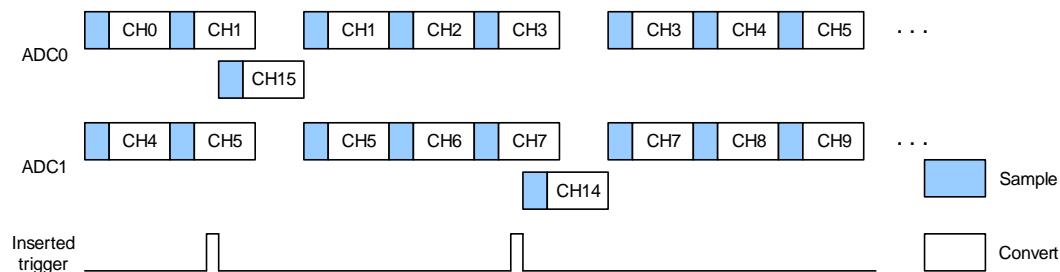
It is possible to interrupt regular group parallel conversion to start trigger rotation conversion of an inserted group. The behavior of an alternate trigger interrupt a regular parallel

conversion shows in the [Figure 11-20. Regular parallel & trigger rotation mode](#).

When the inserted event occurs, the inserted rotation conversion is immediately started. If regular conversion is already running, in order to ensure synchronization after the inserted conversion, the regular conversion of both (master/slave) ADCs is stopped and resumed synchronously at the end of the inserted conversion.

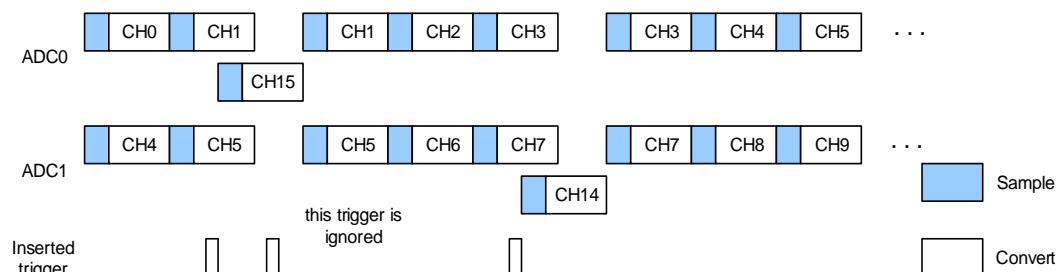
**Note:** In combined regular parallel + trigger rotation mode, the sampling time for the two ADCs should be configured the same.

**Figure 11-20. Regular parallel & trigger rotation mode**



If one inserted trigger occurs during an inserted conversion that has interrupted a regular conversion, it will be ignored [Figure 11-21. Trigger occurs during inserted conversion](#) shows the case (the third trigger is ignored).

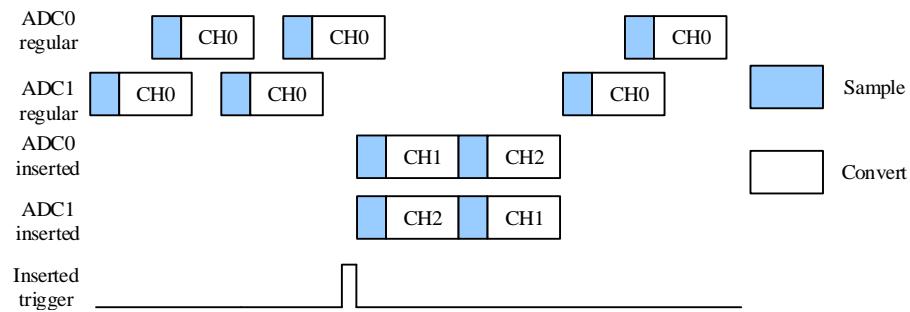
**Figure 11-21. Trigger occurs during inserted conversion**



## 11.6.8. Combined inserted parallel & follow-up mode

It is possible to interrupt a follow-up conversion (both fast and slow) with an inserted event. When the inserted trigger occurs, the follow-up conversion is interrupted and the inserted conversion starts, at the end of the inserted sequence the follow-up conversion is resumed. [Figure 11-22 Follow-up single channel with inserted sequence CH1, CH2](#) shows the behavior of this mode.

**Figure 11-22 Follow-up single channel with inserted sequence CH1, CH2**



## 11.7. ADC interrupts

The interrupt can be produced on one of the events:

- End of conversion for regular and inserted groups
- The analog watchdog event

Separate interrupt enable bits are available for flexibility.

The interrupts of ADC0, ADC1 are mapped into the same interrupt vector ISR[18].

## 11.8. ADC registers

ADC0 base address: 0x4001 2400

ADC1 base address: 0x4001 2800

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

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										STRC	STIC	EOIC	EOC	WDE	
										rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	

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.
---	-----	--

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

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								RWDEN	IWDEN	Reserved		SYNCM[3:0]			
								rw	rw						rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DISNUM[2:0]		DISIC	DISRC	ICA	WDSC	SM	EOICIE	WDEIE	EOCIE	WDCHSEL[4:0]					
	rw	rw	rw	rw	rw	rw	rw	rw	rw						rw

Bits	Fields	Descriptions
31:24	Reserved	Must be kept at reset value
23	RWDEN	Regular channel analog watchdog enable 0: Regular channel analog watchdog disable 1: Regular channel analog watchdog enable
22	IWDEN	Inserted channel analog watchdog enable 0: Inserted channel analog watchdog disable 1: Inserted channel analog watchdog enable
21:20	Reserved	Must be kept at reset value
19:16	SYNCM[3:0]	Sync mode selection These bits use to select the operating mode. 0000: Free mode. 0001: Combined regular parallel + inserted parallel mode 0010: Combined regular parallel + trigger rotation mode 0011: Combined inserted parallel + follow-up fast mode 0100: Combined inserted parallel + follow-up slow mode 0101: Inserted parallel mode only 0110: Regular parallel mode only 0111: Follow-up fast mode only 1000: Follow-up slow mode only

1001: Trigger rotation mode only

Note: These bits are reserved in ADC1.In sync mode, the change of configuration will cause unpredictable consequences. We must disable sync mode before any configuration change.

15:13	DISNUM[2:0]	Number of conversions in discontinuous mode The number of channels to be converted after a trigger will be DISNUM+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 00011: ADC channel 3 00100: ADC channel 4 00101: ADC channel 5 00110: ADC channel 6 00111: ADC channel 7

01000: ADC channel 8  
 01001: ADC channel 9  
 01010: ADC channel 10  
 01011: ADC channel 11  
 01100: ADC channel 12  
 01101: ADC channel 13  
 01110: ADC channel 14  
 01111: ADC channel 15  
 10000: ADC channel 16  
 10001: ADC channel 17  
 Other values are reserved.

Note: ADC0 analog inputs Channel16 and Channel17 are internally connected to the temperature sensor, and to V<sub>REFINT</sub> inputs. ADC1 analog inputs Channel16, and Channel17 are internally connected to V<sub>SSA</sub>.

### 11.8.3. Control register 1 (ADC\_CTL1)

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								TSVREN	SWRCST	SWICST	ETERC	ETSRC[2:0]			Reserved
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETEIC	ETSC[2:0]		DAL	Reserved.		DMA	Reserved				RSTCLB	CLB	CTN	ADCON	
rw	rw		rw	rw				rw				rw	rw	rw	rw

Bits	Fields	Descriptions
31:24	Reserved	Must be kept at reset value
23	TSVREN	Channel 16 and 17 enable of ADC0. 0: Channel 16 and 17 of ADC0 disable 1: Channel 16 and 17 of ADC0 enable
22	SWRCST	Start on regular channel. Set 1 on this bit starts a conversion of a group of regular channels if ETSRC is 111. It is set by software and cleared by software or by hardware after the conversion starts.
21	SWICST	Start on inserted channel. Set 1 on this bit starts a conversion of a group of inserted channels if ETSC[2:0] 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 For ADC0 and ADC1: 000: Timer 0 CH0 001: Timer 0 CH1 010: Timer 0 CH2 011: Timer 1 CH1 100: Timer 2 TRGO 101: Timer 3 CH3 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 For ADC0 and ADC1: 000: Timer 0 TRGO 001: Timer 0 CH3 010: Timer 1 TRGO 011: Timer 1 CH0 100: Timer 2 CH3 101: Timer 3 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 and take a stabilization time. 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

#### 11.8.4. Sample time register 0 (ADC\_SAMPT0)

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								SPT17[2:0]		SPT16[2:0]		SPT15[2:1]			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPT15[0]	SPT14[2:0]			SPT13[2:0]			SPT12[2:0]			SPT11[2:0]			SPT10[2:0]		
rw	rw			rw			rw			rw			rw		

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 sample 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
-----	------------	--

### 11.8.5. Sample time register 1 (ADC\_SAMPT1)

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		SPT9[2:0]		SPT8[2:0]		SPT7[2:0]		SPT6[2:0]		SPT5[2:1]		SPT5[2:0]		SPT4[2:0]	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPT5[0]		SPT4[2:0]		SPT3[2:0]		SPT2[2:0]		SPT1[2:0]		SPT0[2:0]		SPT0[2:0]		SPT0[2:0]	

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 sample 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

### **11.8.6. Inserted channel data offset register x (ADC\_IOFFx) (x=0..3)**

Address offset: 0x14-0x20

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

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 in the ADC_IDATAx registers.

### **11.8.7. Watchdog high threshold register (ADC\_WDHT)**

Address offset: 0x24

Reset value: 0x0000 0FFF

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				WDHT[11:0]											
rw															

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.

### 11.8.8. Watchdog low threshold register (ADC\_WDLT)

Address offset: 0x28

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				WDLT[11:0]											
rw															

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.

### 11.8.9. Regular sequence register 0 (ADC\_RSQ0)

Address offset: 0x2C

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								RL[3:0]				RSQ15[4:1]			
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSQ15[0]		RSQ14[4:0]				RSQ13[4:0]				RSQ12[4:0]					
rw		rw				rw				rw					

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

#### **11.8.10. Regular sequence register 1 (ADC\_RSQ1)**

Address offset: 0x30

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	RSQ11[4:0]				RSQ10[4:0]				RSQ9[4:1]						
	rw							rw					rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSQ9[0]	RSQ8[4:0]				RSQ7[4:0]				RSQ6[4:0]						
	rw					rw					rw				

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
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

#### 11.8.11. Regular sequence register 2 (ADC\_RSQ2)

Address offset: 0x34

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		RSQ5[4:0]				RSQ4[4:0]				RSQ3[4:1]					
rw								rw				rw			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSQ3[0]		RSQ2[4:0]				RSQ1[4:0]				RSQ0[4:0]					
rw				rw				rw				rw			

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..17) is written to these bits to select a channel as the nth conversion in the regular channel group.

### 11.8.12. Inserted sequence register (ADC\_ISQ)

Address offset: 0x38

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										IL[1:0]	ISQ3[4:1]				
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISQ3[0]	ISQ2[4:0]				ISQ1[4:0]				ISQ0[4:0]						
rw	rw				rw				rw						

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 Inserted 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..17) is 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] - 1), if IL[1:0] length is less than 4.
	IL	Insert channel order
	3	ISQ0 >> ISQ1 >> ISQ2 >> ISQ3
	2	ISQ1 >> ISQ2 >> ISQ3
	1	ISQ2 >> ISQ3

### 11.8.13. Inserted data register x (ADC\_IDATAx) (x= 0..3)

Address offset: 0x3C + 0x4\*x,(x=0..3)

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
IDATAn[15:0]															

r

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.

### 11.8.14. Regular data register (ADC\_RDATA)

Address offset: 0x4C

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
ADC1RDTR[15:0]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDATA[15:0]															

r

Bits	Fields	Descriptions
31:16	ADC1RDTR[15:0]	ADC1 regular channel data In ADC0: In sync mode, these bits contain the regular data of ADC1. In ADC1: these bits are not used.
15:0	RDATA[15:0]	Regular channel data These bits contain the conversion result from regular channel, which is read only.

### 11.8.15. Oversample control register (ADC\_OVSAMPCTL)

Address offset: 0x80

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.
13:12	DRES[1:0]	ADC resolution 00: 12bit; 01: 10bit; 10: 8bit; 11: 6bit
11:10	Reserved	Must be kept at reset value
9	TOVS	Triggered Oversampling This bit is set and cleared by software. 0: All oversampled conversions for a channel are done consecutively after a trigger 1: Each oversampled conversion for a channel needs a trigger Note: Software is allowed to write this bit only when ADCON=0 (which ensures that no conversion is ongoing).
8:5	OVSS[3:0]	Oversampling shift This bit is set and cleared by software. 0000: No shift 0001: Shift 1-bit 0010: Shift 2-bits 0011: Shift 3-bits 0100: Shift 4-bits 0101: Shift 5-bits 0110: Shift 6-bits 0111: Shift 7-bits 1000: Shift 8-bits Other codes reserved Note: Software is allowed to write this bit only when ADCON =0 (which ensures that no conversion is ongoing).

4:2	OVSR[2:0]	Oversampling ratio This bit field defines the number of oversampling ratio. 000: 2x 001: 4x 010: 8x 011: 16x 100: 32x 101: 64x 110: 128x 111: 256x  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	Oversampler Enable This bit is set and cleared by software. 0: Oversampler disabled 1: Oversampler enabled  Note: Software is allowed to write this bit only when ADCON =0 (which ensures that no conversion is ongoing).

## 12. Digital-to-analog converter (DAC)

### 12.1. Overview

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.

The two DACs can work independently or concurrently.

### 12.2. Characteristics

DAC's 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, VREF+.
- Noise wave generation (LFSR noise mode and Triangle noise mode).
- Two DACs in concurrent mode.

[\*\*Figure 12-1. DAC block diagram\*\*](#) shows the block diagram of DAC and [\*\*Table 12-1. DAC\*\*](#)

[pins](#) gives the pin description.

Figure 12-1. DAC block diagram

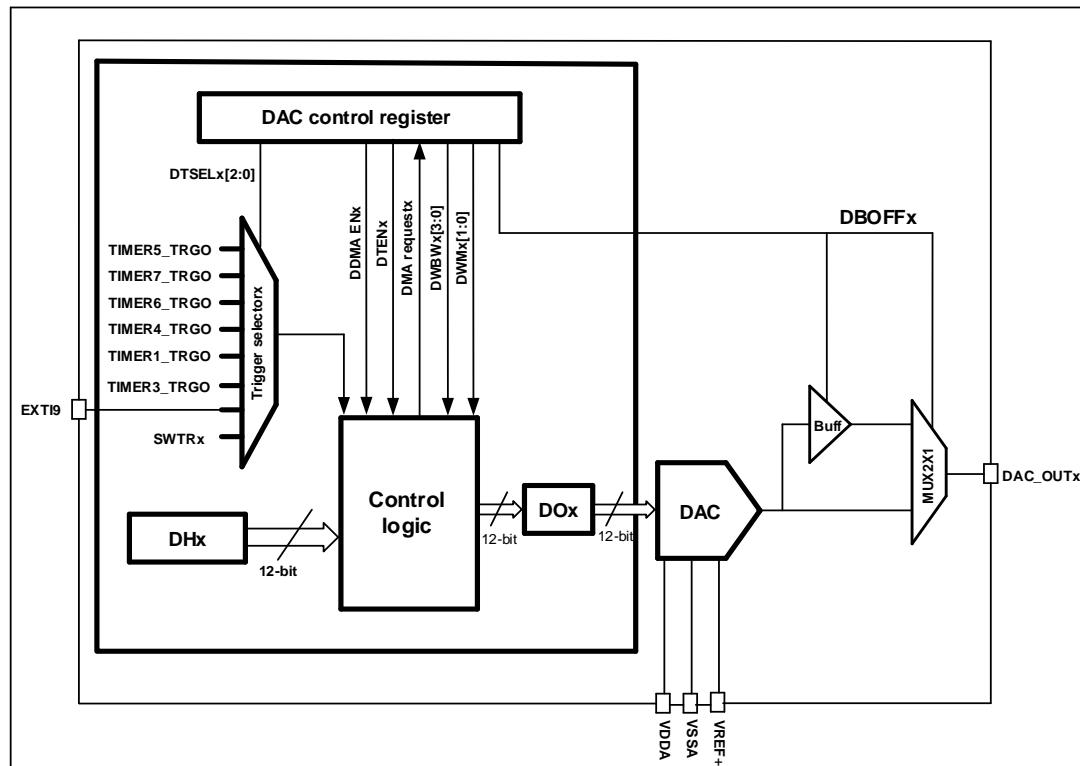


Table 12-1. DAC pins

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
V <sub>REF+</sub>	Positive reference voltage for the DAC, $2.4 \text{ V} \leq V_{\text{REF}+} \leq V_{\text{DDA}}$	Input, analog positive reference
DAC_OUTx	DAC <sub>x</sub> analog output	Analog output signal

The GPIO pins (PA4 for DAC0, PA5 for DAC1) should be configured to analog mode before enable the DAC module.

## 12.3. Function overview

### 12.3.1. DAC enable

The DACs can be powered on by setting the DEN<sub>x</sub> bit in the DAC\_CTL register. A t<sub>WAKEUP</sub> time is needed to startup the analog DAC submodule.

### 12.3.2. 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 to reduce the output impedance and improve the driving capability, can be turned off by setting the DBOFFx bits in the DAC\_CTL register.

### 12.3.3. DAC data configuration

The 12-bit DAC holding data (DAC<sub>x</sub>\_DH) can be configured by writing any one of the DAC<sub>x</sub>\_R12DH, DAC<sub>x</sub>\_L12DH and DAC<sub>x</sub>\_R8DH registers. When the data is loaded by DAC<sub>x</sub>\_R8DH register, only the MSB 8 bits are configurable, the LSB 4 bits are forced to 4'b0000.

### 12.3.4. DAC trigger

The DAC external trigger is enabled by setting the DTEN<sub>x</sub> bits in the DAC\_CTL register. The DAC external triggers are selected by the DTSEL<sub>x</sub> bits in the DAC\_CTL register.

**Table 12-2. External triggers of DAC**

DTSEL <sub>x</sub> [2:0]	Trigger Source	Trigger Type
000	TIMER5_TRGO	Internal on-chip signal
001	TIMER2_TRGO	
010	TIMER6_TRGO	
011	TIMER4_TRGO	
100	TIMER1_TRGO	
101	TIMER3_TRGO	
110	EXTI9	External signal
111	SWTRIG	Software trigger

The TIMER<sub>x</sub>\_TRGO signals are generated from the timers, while the software trigger can be generated by setting the SWTR<sub>x</sub> bits in the DAC\_SWT register.

### 12.3.5. DAC conversion

If the external trigger is enabled by setting the DTEN<sub>x</sub> bit in DAC\_CTL register, the DAC holding data is transferred to the DAC output data (DAC<sub>x</sub>\_DO) register at the selected trigger events. Otherwise, when the external trigger is disabled, the transfer is performed automatically.

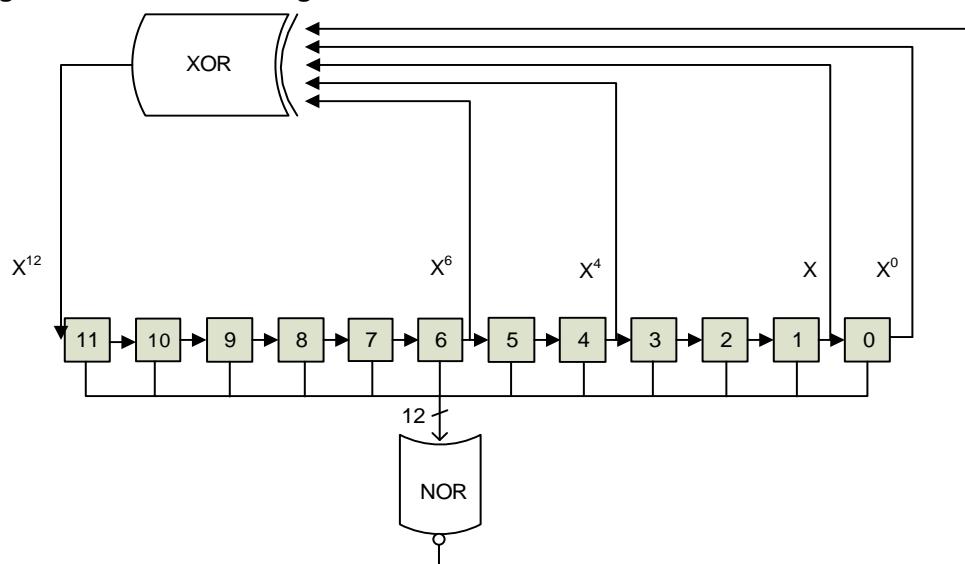
When the DAC holding data (DAC<sub>x</sub>\_DH) is loaded into the DAC<sub>x</sub>\_DO register, after the time tSETTLING, the analog output is valid, and the value of tSETTLING is related to the power supply voltage and the analog output load.

### 12.3.6. DAC noise wave

There are two methods of adding noise wave to the DAC output data: LFSR noise wave and Triangle wave. The noise wave mode can be selected by the DWMx bits in the DAC\_CTL register. The amplitude of the noise can be configured by the DAC noise wave bit width (DWBWx) bits in the DAC\_CTL register.

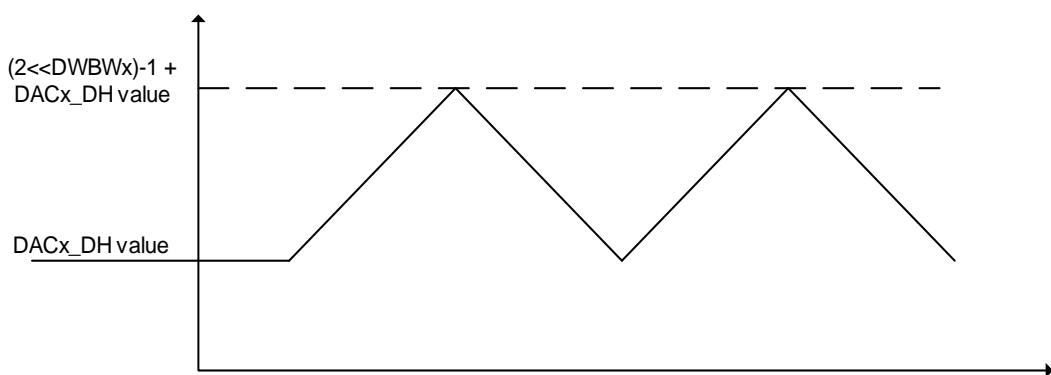
LFSR noise wave mode: there is a Linear Feedback Shift Register (LFSR) in the DAC control logic, it controls the LFSR noise signal which is added to the DACx\_DH value. When the configured DAC noise wave bit width is less than 12, the noise signal equals to the LSB DWBWx bits of the LFSR register, while the MSB bits are masked.

**Figure 12-2. DAC LFSR algorithm**



Triangle noise mode: in this mode, a triangle signal is added to the DACx\_DH value. The minimum value of the triangle signal is 0, while the maximum value of the triangle signal is  $(2^{<<DWBWx}) - 1$ .

**Figure 12-3. DAC triangle noise wave**



### 12.3.7. DAC output voltage

The analog output voltage on the DAC pin is determined by the following equation:

$$\text{DAC}_{\text{output}} = V_{\text{REF+}} * \text{DAC\_DO} / 4096 \quad (12-1)$$

The digital input is linearly converted to an analog output voltage, its range is 0 to  $V_{\text{REF+}}$ .

### 12.3.8. DMA request

When the external trigger is enabled, the DMA request is enabled by setting the DDMAENx bits of the DAC\_CTL register. A DAC DMA request will be generated when an external hardware trigger (not a software trigger) occurs.

### 12.3.9. DAC concurrent conversion

When the two DACs work at the same time, for maximum bus bandwidth utilization in specific applications, two DACs can be configured in concurrent mode. In concurrent mode, two DACs data transfer (DACx\_DH to DACx\_DO) will be at the same time.

There are three concurrent registers that can be used to load the DACx\_DH value: DACC\_R8DH, DACC\_R12DH and DACC\_L12DH. You just need to access a unique register to realize driving both DACs at the same time.

When external trigger is enabled, both DTENx bits should be set. DTSEL0 and DTSEL1 bits should be configured with the same value.

When DMA is enabled, only one of the DDMAENx bits should be set.

The noise mode and noise bit width can be configured either the same or different, depending on the usage.

## 12.4. Register definition

DAC base address: 0x4000 7400

### 12.4.1. Control register (DAC\_CTL)

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	DDMAEN1		DWBW1[3:0]		DWM1[1:0]		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	DDMAENO		DWBW0[3:0]		DWM0[1:0]		DTSEL0[2:0]		DTENO	DBOFF0	DENO		
					rw		rw		rw		rw	rw	rw		

Bits	Fields	Descriptions
31:29	Reserved	Must be kept at reset value
28	DDMAEN1	DAC1 DMA enable 0: DAC1 DMA mode disabled 1: DAC1 DMA mode enabled
27:24	DWBW1[3:0]	DAC1 noise wave bit width These bits specify bit width of the noise wave signal of DAC1. These bits indicate that unmask LFSR bit [n-1, 0] in LFSR noise mode or the amplitude of the triangle is ((2<<(n-1))-1) in triangle noise mode, where n is the bit width of wave. 0000: The bit width of the wave signal is 1 0001: The bit width of the wave signal is 2 0010: The bit width of the wave signal is 3 0011: The bit width of the wave signal is 4 0100: The bit width of the wave signal is 5 0101: The bit width of the wave signal is 6 0110: The bit width of the wave signal is 7 0111: The bit width of the wave signal is 8 1000: The bit width of the wave signal is 9 1001: The bit width of the wave signal is 10 1010: The bit width of the wave signal is 11 ≥1011: The bit width of the wave signal is 12
23:22	DWM1[1:0]	DAC1 noise wave mode These bits specify the mode selection of the noise wave signal of DAC1 when external trigger of DAC1 is enabled (DTEN1=1).

		00: wave disabled 01: LFSR noise mode 1x: Triangle noise mode
21:19	DTSEL1[2:0]	DAC1 trigger selection  These bits select the external trigger of DAC1 when DTEN1=1.  000: Timer 5 TRGO 001: Timer 2 TRGO 010: Timer 6 TRGO 011: Timer 4 TRGO 100: Timer 1 TRGO 101: Timer 3 TRGO 110: EXTI line 9 111: Software trigger
18	DTEN1	DAC1 trigger enable  0: DAC1 trigger disabled 1: DAC1 trigger enabled
17	DBOFF1	DAC1 output buffer turn off  0: DAC1 output buffer turns on to reduce the output impedance and improve the driving capability 1: DAC1 output buffer turn off
16	DEN1	DAC1 enable  0: DAC1 disabled 1: DAC1 enabled
15:13	Reserved	Must be kept at reset value
12	DDMAEN0	DAC0 DMA enable  0: DAC0 DMA mode disabled 1: DAC0 DMA mode enabled
11:8	DWBW0[3:0]	DAC0 noise wave bit width  These bits specify bit width of the noise wave signal of DAC0. These bits indicate that unmask LFSR bit [n-1, 0] in LFSR noise mode or the amplitude of the triangle is $((2^{<<(n-1)})-1)$ in triangle noise mode, where n is the bit width of wave.  0000: The bit width of the wave signal is 1 0001: The bit width of the wave signal is 2 0010: The bit width of the wave signal is 3 0011: The bit width of the wave signal is 4 0100: The bit width of the wave signal is 5 0101: The bit width of the wave signal is 6 0110: The bit width of the wave signal is 7 0111: The bit width of the wave signal is 8 1000: The bit width of the wave signal is 9

		1001: The bit width of the wave signal is 10 1010: The bit width of the wave signal is 11 ≥1011: The bit width of the wave signal is 12
7:6	DWM0[1:0]	<p>DAC0 noise wave mode</p> <p>These bits specify the mode selection of the noise wave signal of DAC0 when external trigger of DAC0 is enabled (DTEN0=1).</p> <p>00: wave disabled 01: LFSR noise mode 1x: Triangle noise mode</p>
5:3	DTSEL0[2:0]	<p>DAC0 trigger selection</p> <p>These bits select the external trigger of DAC0 when DTEN0=1.</p> <p>000: Timer 5 TRGO 001: Timer 2 TRGO 010: Timer 6 TRGO 011: Timer 4 TRGO 100: Timer 1 TRGO 101: Timer 3 TRGO 110: EXTI line 9 111: Software trigger</p>
2	DTEN0	<p>DAC0 trigger enable</p> <p>0: DAC0 trigger disabled 1: DAC0 trigger enabled</p>
1	DBOFF0	<p>DAC0 output buffer turn off</p> <p>0: DAC0 output buffer turns on to reduce the output impedance and improve the driving capability 1: DAC0 output buffer turn off</p>
0	DEN0	<p>DAC0 enable</p> <p>0: DAC0 disabled 1: DAC0 enabled</p>

### 12.4.2. Software trigger register (DAC\_SWT)

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																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved															SWTR1	SWTR0

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
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

### 12.4.3. 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)

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				DAC0_DH[11:0]											
rw															

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
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.

### 12.4.4. 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)

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
DAC0_DH[11:0]												Reserved			
rw															

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
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

#### 12.4.5. 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)

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								DAC0_DH[7:0]							

rw

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
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 MSB 8 bits of the data that is to be converted by DAC0.

#### 12.4.6. DAC1 12-bit right-aligned data holding register (DAC1\_R12DH)

Address offset: 0x14

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								DAC1_DH[11:0]							

rw

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
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.

#### **12.4.7. DAC1 12-bit left-aligned data holding register (DAC1\_L12DH)**

Address offset: 0x18

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
DAC1_DH[11:0]										Reserved					

rw

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

#### **12.4.8. DAC1 8-bit right-aligned data holding register (DAC1\_R8DH)**

Address offset: 0x1C

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								DAC1_DH[7:0]							

rw

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 MSB bits of the data that is to be converted by DAC1.

### **12.4.9. DAC concurrent mode 12-bit right-aligned data holding register (DACC\_R12DH)**

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				DAC1_DH[11:0]											
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				DAC0_DH[11:0]											
rw															

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
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.

### **12.4.10. DAC concurrent mode 12-bit left-aligned data holding register (DACC\_L12DH)**

Address offset: 0x24

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
DAC1_DH[11:0]												Reserved			
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DAC0_DH[11:0]												Reserved			
rw															

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
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

#### **12.4.11. DAC concurrent mode 8-bit right-aligned data holding register (DACC\_R8DH)**

Address offset: 0x28

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
DAC1_DH [7:0]								DAC0_DH [7:0]							
rw								rw							

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 MSB 8-bit of 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 MSB 8-bit of the data that is to be converted by DAC0.

#### **12.4.12. DAC0 data output register (DAC0\_DO)**

Address offset: 0x2C

Reset value: 0x0000 0000

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

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								DAC0_DO [11:0]							
r															

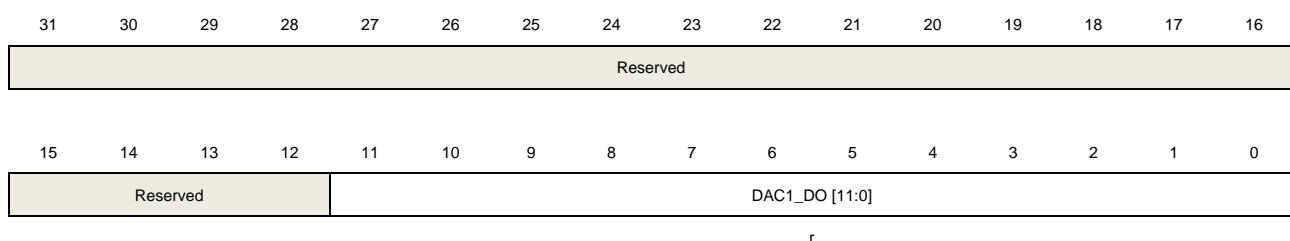
<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:12	Reserved	Must be kept at reset value
11:0	DAC0_DO [11:0]	DAC0 data output These bits, which are read only, reflect the data that is being converted by DAC0.

### 12.4.13. DAC1 data output register (DAC1\_DO)

Address offset: 0x30

Reset value: 0x0000 0000

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



<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:12	Reserved	Must be kept at reset value
11:0	DAC1_DO [11:0]	DAC1 data output These bits, which are read only, reflect the data that is being converted by DAC1.

## 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 (WWDT). 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 timer 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.

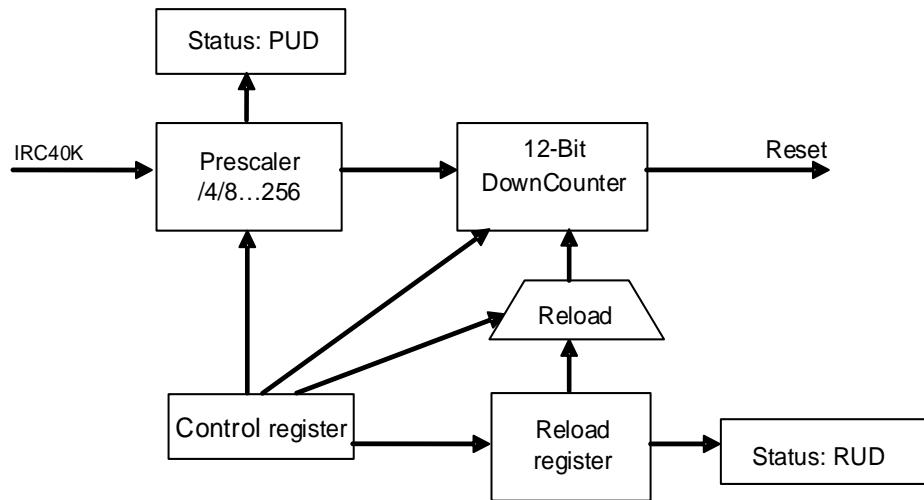
The free watchdog timer causes a reset when the internal down counter reaches 0. The register write protection function in free watchdog can be enabled to prevent it from changing the configuration unexpectedly.

#### 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 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 consists of an 8-stage prescaler and a 12-bit down-counter. Refer to the figure below for the functional block of the free watchdog module.

**Figure 13.1. Free watchdog block diagram**


The free watchdog 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 anytime. 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.

The free watchdog can automatically start at power on when the hardware free watchdog bit in the device option bytes is set. To avoid reset, the software should reload the counter before the counter reaches 0x000.

The FWDGT\_PSC register and the FWDGT\_RLD register are write-protected. Before writing these registers, the software should write the value 0x5555 to the FWDGT\_CTL register. These registers will be protected again by writing any other value to the FWDGT\_CTL register. When an update operation of the prescaler register (FWDGT\_PSC) or the reload value register (FWDGT\_RLD) is on going, the status bits in the FWDGT\_STAT register are set.

If the FWDGT\_HOLD bit in DBG module is cleared, the FWDGT continues to work even the RISC-V 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) RLD[11:0]=0x000	Max timeout (ms) RLD[11:0]=0xFFFF
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

Prescaler divider	PSC[2:0] bits	Min timeout (ms) RLD[11:0]=0x000	Max timeout (ms) RLD[11:0]=0xFFFF
1/256	110 or 111	6.4	26214.4

The FWDGT timeout can be more accurate by calibrating the IRC40K.

**Note:**

- For all the GD32VF103 devices, when after the execution of dog reload operation, if the MCU needs enter the deepsleep/standby mode immediately, (more than 3) IRC40K clock interval must be inserted in the middle of reload and deepsleep/standby mode commands by software setting.
- For all the GD32VF103 devices, when software finished the executing operation of FWDGT, if the MCU needs enter the deepsleep/standby mode immediately, it is at least 100 us interval left between the two instructions.
- For all the GD32VF103 devices, if you need access to the MCU debug mode, recommend to use hardware watchdog, or enable watchdog again after exit debug mode by software setting.

### 13.1.4. Register definition

FWDGT base address: 0x4000 3000

#### 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

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
CMD[15:0]															

w

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	CMD[15:0]	Write only. Several different fuctions are realized by writing these bits with different values: 0x5555: Disable the FWDGT_PSC and FWDGT_RLD write protection. 0xCCCC: Start the free watchdog counter. When the counter reduces to 0, the free watchdog 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

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															

rw

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. During a write operation to this register, 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 except in case of low-power mode entry.

### **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

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:12	Reserved	Must be kept at reset value
11:0	RLD[11:0]	<p>Free watchdog timer counter reload value. Write 0xAAAA in the FWDGT_CTL register will reload the FWDGT counter with the RLD value.</p> <p>These bits are write-protected. Write 0x5555 in the FWDGT_CTL register before writing these bits. During a write operation to this register, the RUD bit in the FWDGT_STAT register is set and the value read from this register is invalid.</p> <p>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 except in case of low-power mode entry.</p>

### **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

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														RUD	PUD

r r

Bits	Fields	Descriptions
31:2	Reserved	Must be kept at reset value
1	RUD	Free watchdog timer counter reload value update During a write operation to FWDGT_RLD register, this bit is set and the value read from FWDGT_RLD register is invalid. This bit is reset by hardware after the update operation of FWDGT_RLD register.
0	PUD	Free watchdog timer prescaler value update During a write operation to FWDGT_PSC register, this bit is set and the value read from FWDGT_PSC register is invalid. This bit is reset by hardware after the update operation of FWDGT_PSC register.

## 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 reduces progressively. The watchdog timer causes a reset when the counter reached 0x3F (the CNT[6] bit becomes cleared). The watchdog timer also causes 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.

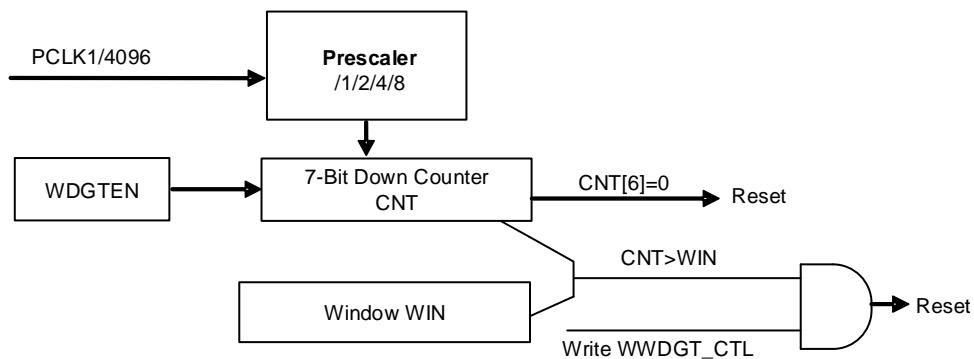
### 13.2.2. Characteristics

- 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 enabled (set the WDGTE bit in the WWDGT\_CTL), the watchdog timer cause a reset when the counter reaches 0x3F (the CNT[6] bit becomes cleared), or when the counter is refreshed before the counter reaches the window register value.

**Figure 13.2. Window watchdog timer block diagram**

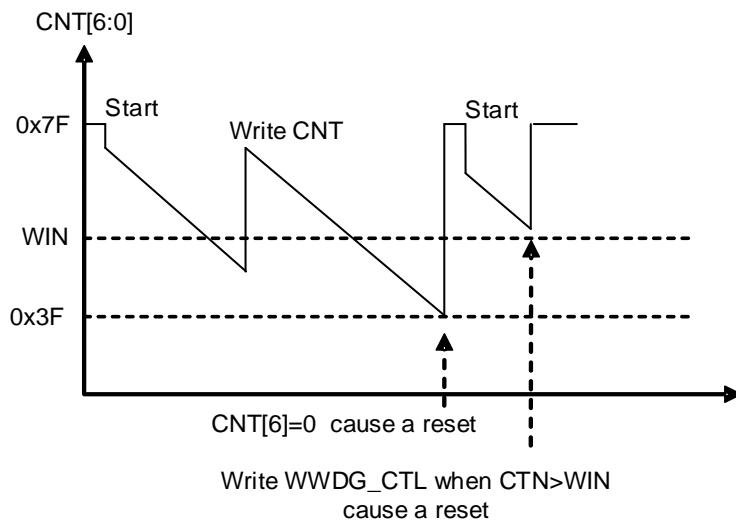


The watchdog is always disabled after power on reset. The software starts the watchdog by setting the WDGTEEN bit in the WWDGT\_CTRL register. Whenever window watchdog timer is enabled, the counter counts down all the time, the configured value of the counter should be greater than 0x3F, it implies that 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 timing diagram**


Calculate the WWDGT timeout by using the formula below.

$$t_{WWDGT} = t_{PCLK1} \times 4096 \times 2^{PSC} \times (CNT[5:0]+1) \text{ (ms)} \quad (13-1)$$

where:

$t_{WWDGT}$ : WWDGT timeout

$t_{PCLK1}$ : APB1 clock period measured in ms

Refer to the table below for the minimum and maximum values of the  $t_{WWDGT}$ .

**Table 13.2. Min/max timeout value at 54 MHz ( $f_{PCLK1}$ )**

Prescaler divider	PSC[1:0]	Min timeout value CNT[6:0] =0x40	Max timeout value CNT[6:0]=0x7F
1/1	00	75.8 $\mu$ s	4.85ms
1/2	01	151.7 $\mu$ s	9.7 ms
1/4	10	303.4 $\mu$ s	19.4 ms
1/8	11	606.8 $\mu$ s	38.8 ms

If the WWDGT\_HOLD bit in DBG module is cleared, the WWDGT continues to work even the RISC-V core halted (Debug mode). While the WWDGT\_HOLD bit is set, the WWDGT stops in Debug mode.

#### **13.2.4. Register definition**

WWDT base address: 0x4000 2C00

### **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	<p>Start the window watchdog timer. Only can be cleared by a hardware reset. Writing 0 has no effect.</p> <p>0: Window watchdog timer disabled 1: Window watchdog timer enabled</p>
6:0	CNT[6:0]	The value of the watchdog timer counter. A reset occurs when the value of this counter decreases from 0x40 to 0x3F. When the value of this counter is greater than the window value, writing this counter also causes 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)

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:10	Reserved	Must be kept at reset value.

9	EWIE	Early wakeup interrupt enable. An interrupt occurs when the counter reaches 0x40 or the counter is refreshed before it reaches the window value if the bit is set. It can be cleared by a hardware reset or by a RCU WWDGT software reset. A write operation of '0' has no effect.
8:7	PSC[1:0]	Prescaler. The time base of the watchdog timer 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 occurs 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)

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

rw

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:1	Reserved	Must be kept at reset value.
0	EWIF	Early wakeup interrupt flag. When the counter reaches 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 to it. There is no effect when writing 1 to it.

## 14. Real-time Clock (RTC)

### 14.1. Overview

The RTC is usually used as a clock-calendar. The RTC circuits are located in two power supply domains. The ones in the Backup Domain consist of a 32-bit up-counter, an alarm, a prescaler, a divider and the RTC clock configuration register. That means the RTC settings and time are kept when the device resets or wakes up from Standby mode. While the circuits in the VDD domain only include the APB interface and a control register. In the following sections, the details of the RTC function will be described.

### 14.2. Characteristics

- 32-bit programmable counter for counting elapsed time  
Programmable prescaler: Max division factor is up to  $2^{20}$
- Separate clock domains:
  - A) PCLK1 clock domain
  - B) RTC clock domain (this clock must be at least 4 times slower than the PCLK1 clock)
- RTC clock source:
  - A) HXTAL clock divided by 128
  - B) LXTAL oscillator clock
  - C) IRC40K oscillator clock
- Maskable interrupt source:
  - A) Alarm interrupt
  - B) Second interrupt
  - C) Overflow interrupt

### 14.3. Function overview

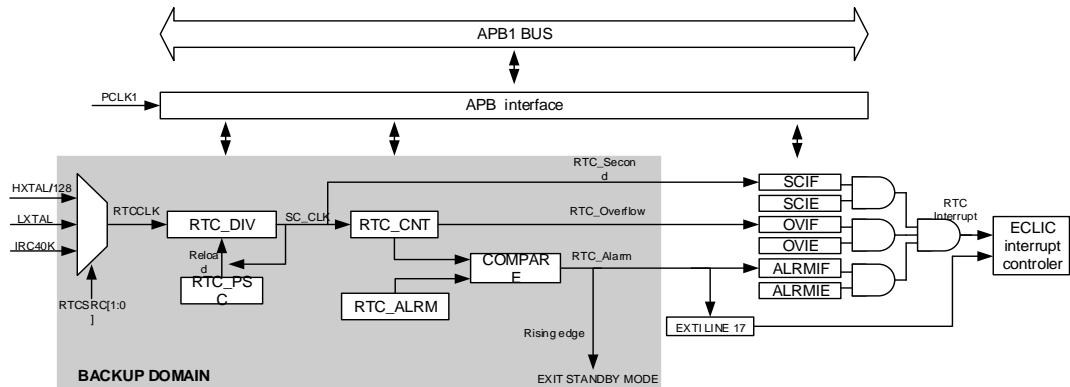
The RTC circuits consist of two major units: APB interface located in PCLK1 clock domain and RTC core located in RTC clock domain.

APB Interface is connected with the APB1 bus. It includes a set of registers, can be accessed by APB1 bus.

RTC core includes two major blocks. One is the RTC prescaler block, which generates the RTC time base clock SC\_CLK. RTC prescaler block includes a 20-bit programmable divider (RTC prescaler) which can make SC\_CLK is divided from RTC source clock. If second interrupt is enabled in the RTC\_INTEN register, the RTC will generate an interrupt at every SC\_CLK rising edge. Another block is a 32-bit programmable counter, which can be initialized

with the value of current system time. If alarm interrupt is enabled in the RTC\_INTEN register, the RTC will generate an alarm interrupt when the system time equals to the alarm time (stored in the RTC\_ALRMH/L register),

**Figure 14.1. Block diagram of RTC**



### 14.3.1. RTC reset

The APB interface and the RTC\_INTEN register are reset by system reset. The RTC core (prescaler, divider, counter and alarm) is reset only by a backup domain reset.

Steps to enable access to the backup registers and the RTC after reset are as follows:

1. Set the PMUEN and BKPIEN bits in the RCU\_APB1EN register to enable the power and backup interface clocks.
2. Enable access to the backup registers and RTC by setting the BKPWEN bit in the (PMU\_CTL).

### 14.3.2. RTC reading

The APB interface and RTC core are located in two different power supply domains.

In the RTC core, only counter and divider registers are readable registers. And the values in the two registers and the RTC flags are internally updated at each rising edge of the RTC clock, which is resynchronized by the APB1 clock.

When the APB interface is immediately enabled from a disable state, the read operation is not recommended because the first internal update of the registers has not finished. That means, when a system reset, power reset, waking up from Standby mode or Deep-sleep mode occurs, the APB interface was in disabled state, but the RTC core has been kept running. In these cases, the correct read operation should first clear the RSYNF bit in the RTC \_CTL register and wait for it to be set by hardware. While WFI and WFE have no effects on the RTC APB interface.

### 14.3.3. RTC configuration

The RTC\_PSC, RTC\_CNT and RTC\_ALRM registers in the RTC core are writable. These registers' value can be set only when the peripheral enter configuration mode. And the CMF bit in the RTC\_CTL register is used to indicate the configuration mode status. The write operation executes when the peripheral exit configuration mode, and it takes at least three RTCCLK cycles to complete. The value of the LWOFF bit in the RTC\_CTL register sets to '1', if the write operation finished. The new write operation should wait for the previous one finished.

The configuration steps are as follows:

- a) Wait until the value of LWOFF bit in the RTC\_CTL register sets to '1';
- b) Enter Configuration mode by setting the CMF bit in the RTC\_CTL register;
- c) Write to the RTC registers;
- d) Exit Configuration mode by clearing the CMF bit in the RTC\_CTL register;
- e) Wait until the value of LWOFF bit in the RTC\_CTL register sets to '1'.

### 14.3.4. RTC flag assertion

Before the update of the RTC Counter, the RTC second interrupt flag (SCIF) is asserted on the last RTCCLK cycle.

Before the counter equal to the RTC Alarm value which stored in the Alarm register increases by one, the RTC Alarm interrupt flag (ALRMIF) is asserted on the last RTCCLK cycle.

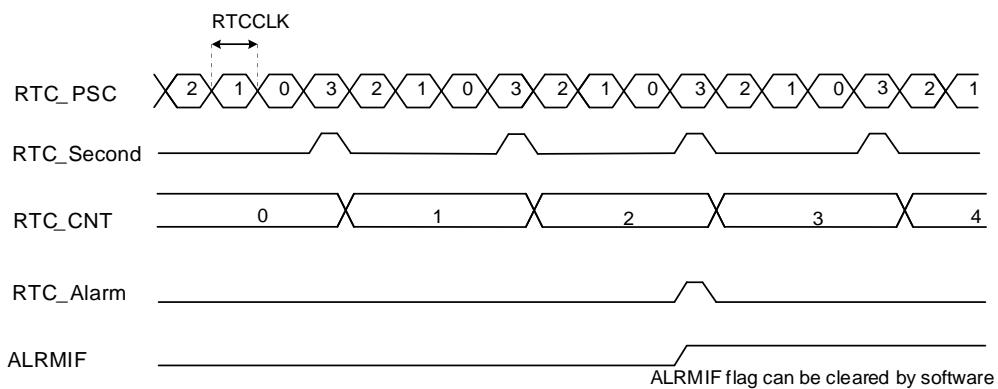
Before the counter equals to 0x0, the RTC Overflow interrupt flag (OVIF) is asserted on the last RTCCLK cycle.

The RTC Alarm write operation and Second interrupt flag must be synchronized by using either of the following sequences:

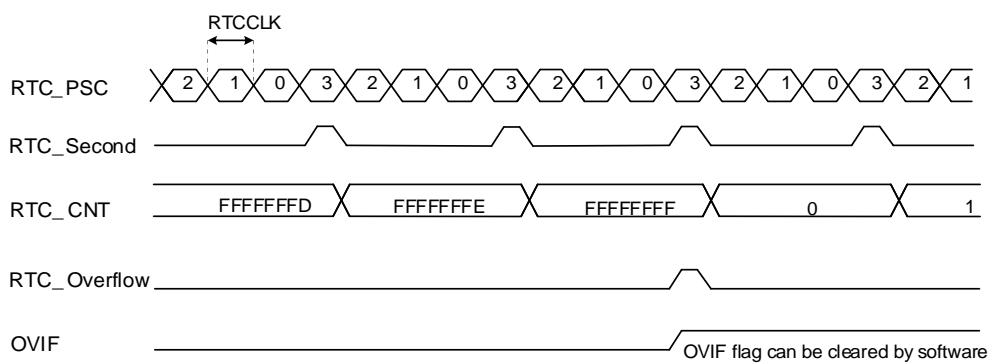
- Use the RTC alarm interrupt and update the RTC Alarm and/or RTC Counter registers inside the RTC interrupt routine;
- Update the RTC Alarm and/or the RTC Counter registers after the SCIF bit to be set in the RTC Control register.

**Figure 14.2. RTC second and alarm waveform example (RTC\_PSC = 3, RTC\_ALRM =**

2)



**Figure 14.3. RTC second and overflow waveform example (RTC\_PSC= 3)**



## 14.4. Register definition

RTC base address: 0x4000 2800

### 14.4.1. RTC interrupt enable register(RTC\_INTEN)

Address offset: 0x00

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												OVIE	ALRMIE	SCIE	
rw      rw      rw															

Bits	Fields	Descriptions
31:3	Reserved	Must be kept at reset value.
2	OVIE	Overflow interrupt enable 0: Disable overflow interrupt 1: Enable overflow interrupt
1	ALRMIE	Alarm interrupt enable 0: Disable alarm interrupt 1: Enable alarm interrupt
0	SCIE	Second interrupt enable 0: Disable second interrupt 1: Enable second interrupt

### 14.4.2. RTC control register(RTC\_CTL)

Address offset: 0x04

Reset value: 0x0020

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												LWOFF	CMF	RSYNF	OVIF	ALRMIF	SCIF
r      rw      rc_w0      rc_w0      rc_w0      rc_w0																	

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:6	Reserved	Must be kept at reset value
5	LWOFF	Last write operation finished flag 0: Last write operation on RTC registers did not finished. 1: Last write operation on RTC registers finished.
4	CMF	Configuration mode flag 0: Exit configuration mode. 1: Enter configuration mode.
3	RSYNF	Registers synchronized flag 0: Registers not yet synchronized with the APB1 clock. 1: Registers synchronized with the APB1 clock.
2	OVIF	Overflow interrupt flag 0: Overflow event not detected 1: Overflow event detected. An interrupt will occur if the OVIE bit is set in RTC_INTEN.
1	ALRMIF	Alarm interrupt flag 0: Alarm event not detected 1: Alarm event detected. An interrupt named RTC global interrupt will occur if the ALRMIE bit is set in RTC_INTEN. And another interrupt named the RTC Alarm interrupt will occur if the EXTI 17 is enabled in interrupt mode.
0	SCIF	Second interrupt flag 0: Second event not detected. 1: Second event detected. An interrupt will occur if the SCIE bit is set in RTC_INTEN. Set by hardware when the divider reloads the value in RTC_PSCH/L, thus incrementing the RTC counter.

#### 14.4.3. RTC prescaler high register (RTC\_PSCH)

Address offset: 0x08

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														PSC[19:16]	

w

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:4	Reserved	Must be kept at reset value
3:0	PSC[19:16]	RTC prescaler value high

#### 14.4.4. RTC prescaler low register (RTC\_PSCL)

Address offset: 0x0C

Reset value: 0x8000

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
PSC[15:0]															

w

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:16	Reserved	Must be kept at reset value
15:0	PSC[15:0]	RTC prescaler value low The frequency of SC_CLK is the RTCCLK frequency divided by (PSC[19:0]+1).

#### 14.4.5. RTC divider high register (RTC\_DIVH)

Address offset: 0x10

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															

r

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:4	Reserved	Must be kept at reset value
3:0	DIV[19:16]	RTC divider value high

#### 14.4.6. RTC divider low register (RTC\_DIVL)

Address offset: 0x14

Reset value: 0x8000

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
DIV[15:0]															

r

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	DIV[15:0]	RTC divider value low The RTC divider register is reloaded by hardware when the RTC prescaler or RTC counter register updated.

#### 14.4.7. RTC counter high register (RTC\_CNTH)

Address offset: 0x18

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
CNT[31:16]															

rw

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	CNT[31:16]	RTC counter value high

#### 14.4.8. RTC counter low register (RTC\_CNTL)

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
CNT[15:0]															

rw

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:16	Reserved	Must be kept at reset value
15:0	CNT[15:0]	RTC counter value low

#### 14.4.9. RTC alarm high register (RTC\_ALRMH)

Address offset: 0x20

Reset value: 0xFFFF

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
ALRM[31:16]															

w

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:16	Reserved	Must be kept at reset value
15:0	ALRM[31:16]	RTC alarm value high

#### 14.4.10. RTC alarm low register (RTC\_ALRML)

Address offset: 0x24

Reset value: 0xFFFF

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
ALRM[15:0]															

w

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:16	Reserved	Must be kept at reset value
15:0	ALRM[15:0]	RTC alarm value low

## 15. Timer(TIMERx)

Table 15-1. Timers (TIMERx) are devided into three sorts

TIMER	TIMER0	TIMER1/2/3/4	TIMER5/6
TYPE	Advanced	General-L0	Basic
Prescaler	16-bit	16-bit	16-bit
Counter	16-bit	16-bit	16-bit
Count mode	UP,DOWN, Center-aligned	UP,DOWN, Center-aligned	UP ONLY
Repetition	•	×	×
CH Capture/ Compare	4	4	0
Complementary & Dead-time	•	×	×
Break	•	×	×
Single Pulse	•	•	•
Quadrature Decoder	•	•	×
Slave Controller	•	•	×
Inter connection	• <sup>(1)</sup>	• <sup>(2)</sup>	TRGO TO DAC
DMA	•	•	• <sup>(3)</sup>
Debug Mode	•	•	•

(1) TIMER0      **ITI0:** TIMER4\_TRGO      **ITI1:** TIMER1\_TRGO      **ITI2:** TIMER2\_TRGO      **ITI3:** TIMER3\_TRGO

(2) TIMER1      **ITI0:** TIMER0\_TRGO      **ITI1:** refer to note (4)      **ITI2:** TIMER2\_TRGO      **ITI3:** TIMER3\_TRGO

                     TIMER2      **ITI0:** TIMER0\_TRGO      **ITI1:** TIMER1\_TRGO      **ITI2:** TIMER4\_TRGO      **ITI3:** TIMER3\_TRGO

                     TIMER3      **ITI0:** TIMER0\_TRGO      **ITI1:** TIMER1\_TRGO      **ITI2:** TIMER2\_TRGO      **ITI3:** 0

                     TIMER4      **ITI0:** TIMER1\_TRGO      **ITI1:** TIMER2\_TRGO      **ITI2:** TIMER3\_TRGO      **ITI3:** 0

(3) Only update events will generate DMA request. Note that TIMER5/6 do not have DMA configuration registers.

(4) The source of TIMER1 ITI1 is decided by TIMER1ITI1\_REMAP in [AFIO port configuration register 0 \(AFIO\\_PCF0\)](#):

## 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 is suitable for motor control applications.

Timers are completely independent with each other, but they may be synchronized to provide a larger timer with their counter value incrementing in unison.

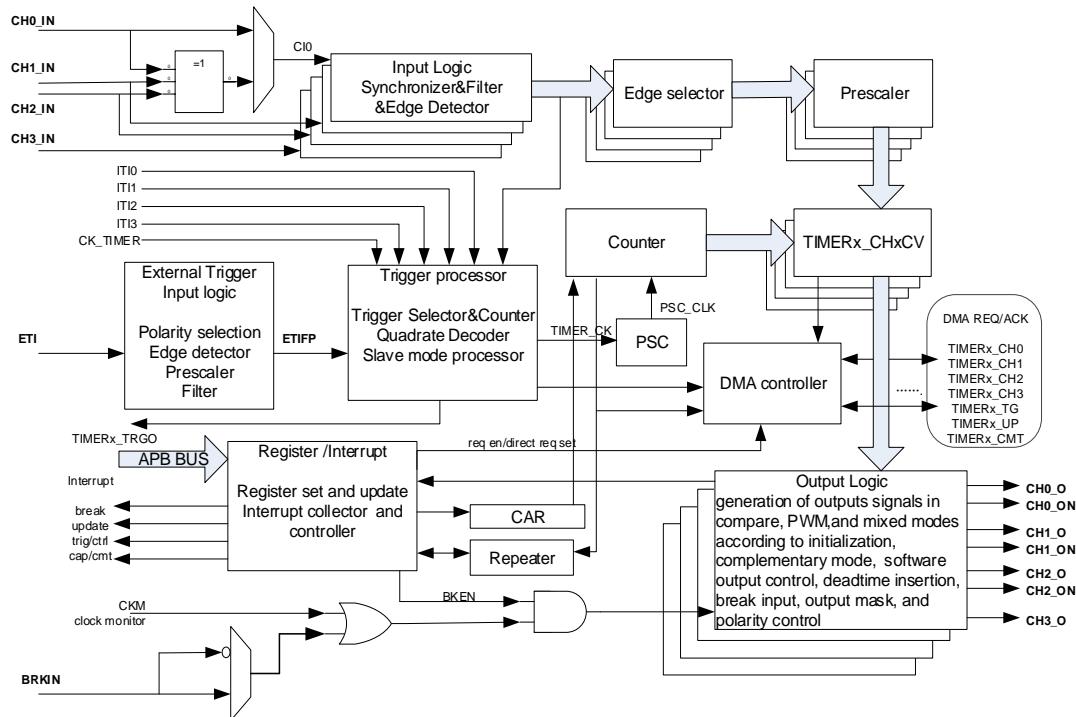
### 15.1.2. Characteristics

- Total channel num: 4.
- Counter width: 16-bit.
- Clock source of timer is selectable: internal clock, internal trigger, external input, external trigger.
- Multiple counter modes: up counting, down counting and center-aligned counting.
- Quadrature decoder: used for motion tracking and determination of both rotation direction and position.
- Hall sensor function: used for 3-phase motor control.
- Programmable prescaler: 16-bit. The factor can be changed ongoing.
- Each channel is user-configurable: input capture mode, output compare mode, programmable PWM mode and single pulse mode
- Programmable dead time insertion.
- Auto reload function.
- Programmable counter repetition function.
- Break input.
- Interrupt output or DMA request: update event, trigger event, compare/capture event and break input.
- Daisy chaining of timer module allows a single timer to start multiple timers.
- Timer synchronization allows the 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 clock source of the advanced timer can be either the CK\_TIMER or an alternate clock source controlled by SMC bits (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.

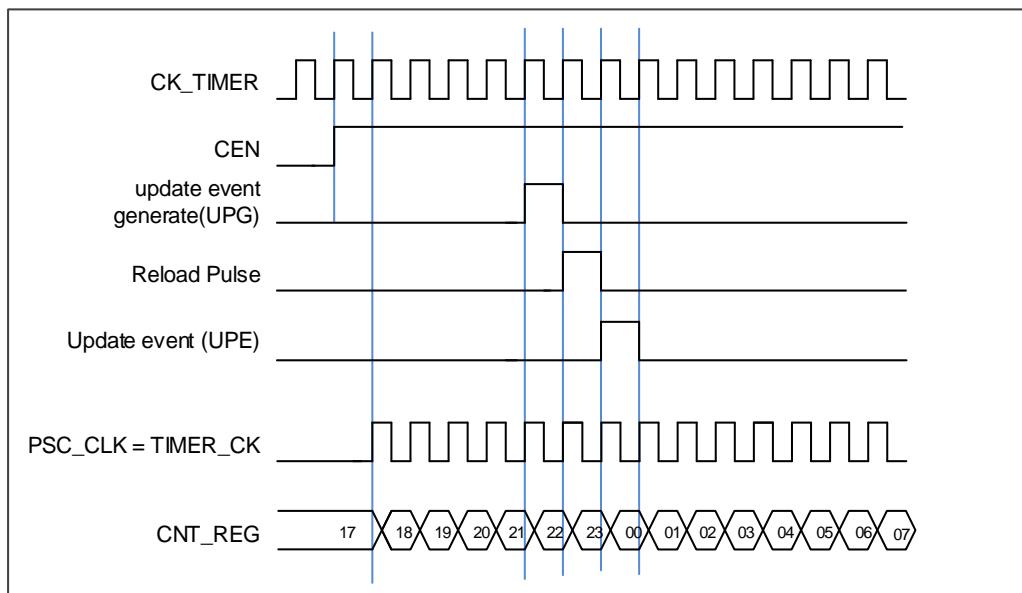
The default clock source is the CK\_TIMER for driving the counter prescaler when the slave mode is disabled (SMC [2:0] == 3'b000). When the CEN is set, the CK\_TIMER will be divided by PSC value to generate PSC\_CLK.

In this mode, the TIMER\_CK which drives counter's prescaler to count is equal to CK\_TIMER which is from RCU module.

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, more details will be

introduced later. When the slave mode control 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 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.

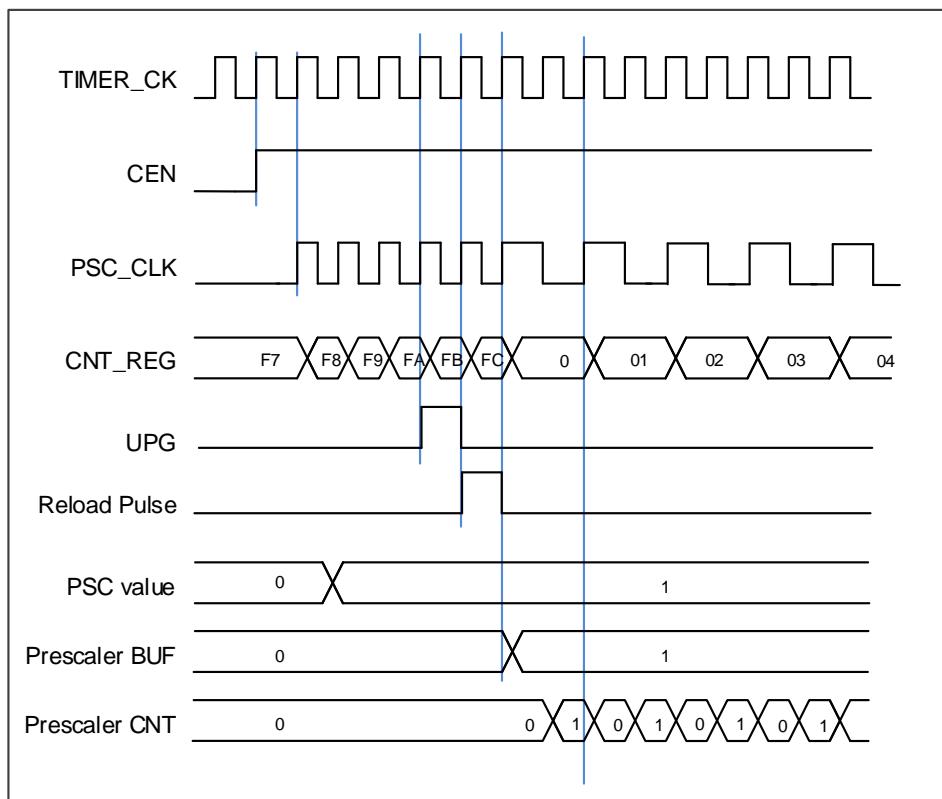
- SMC1== 1'b1 (external clock mode 1). External input ETI is selected as timer clock source

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 ETI. This mode can be selected by setting the SMC1 bit in the TIMERx\_SMCFG register to 1. The other way to select the ETI signal as the clock source is setting the SMC [2:0] to 0x7 and the TRGS [2:0] to 0x7. Note that the ETI signal is derived from the ETI pin sampled by a digital filter. When the ETI signal is selected as the 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 ranging from 1 to 65536. It is controlled by prescaler register (TIMERx\_PSC) which can be changed ongoing, but it is adopted 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 event will be generated after  $(\text{TIMERx_CREP}+1)$  times of overflow. Otherwise the update event is generated each time when counter 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 an update event will be generated.

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 register, auto reload register, prescaler register) are updated.

[Figure 15-4. Timing chart of up counting mode, PSC=0/1](#) and [Figure 15-5. Timing chart of up counting mode, change TIMERx\\_CAR ongoing](#) show some examples of the counter behavior for different clock prescaler factors when `TIMERx_CAR=0x63`.

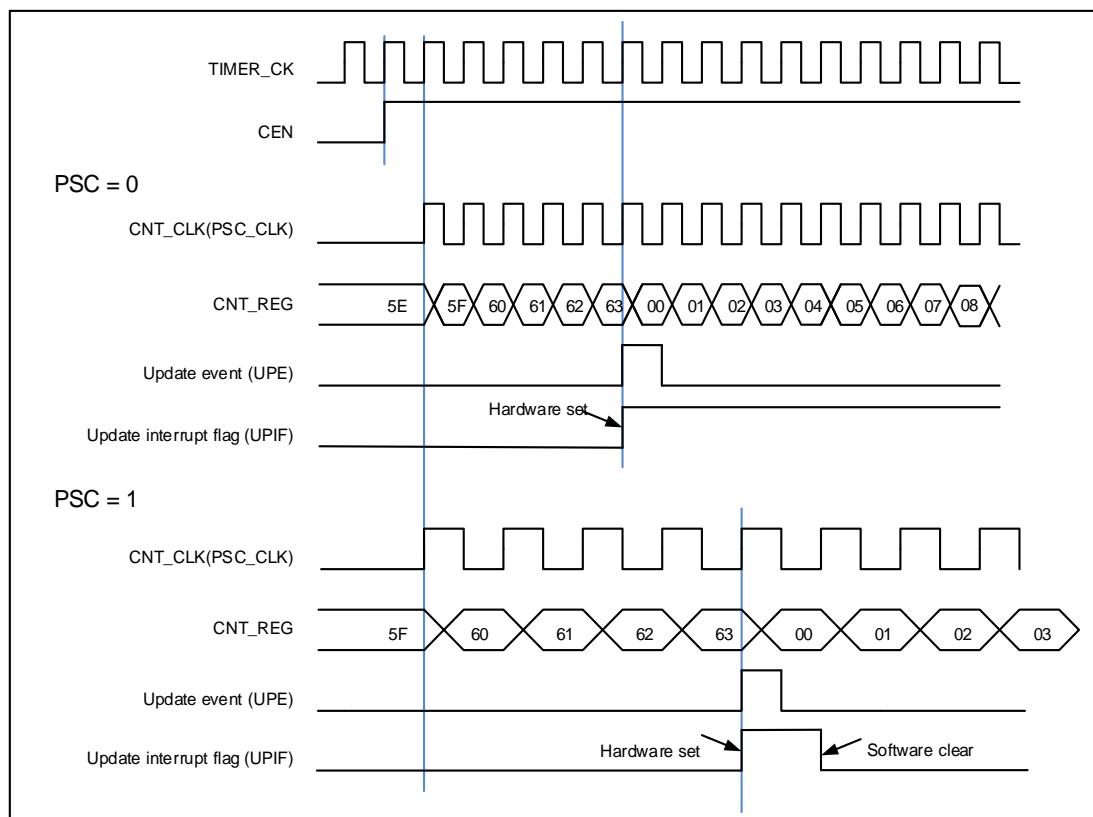
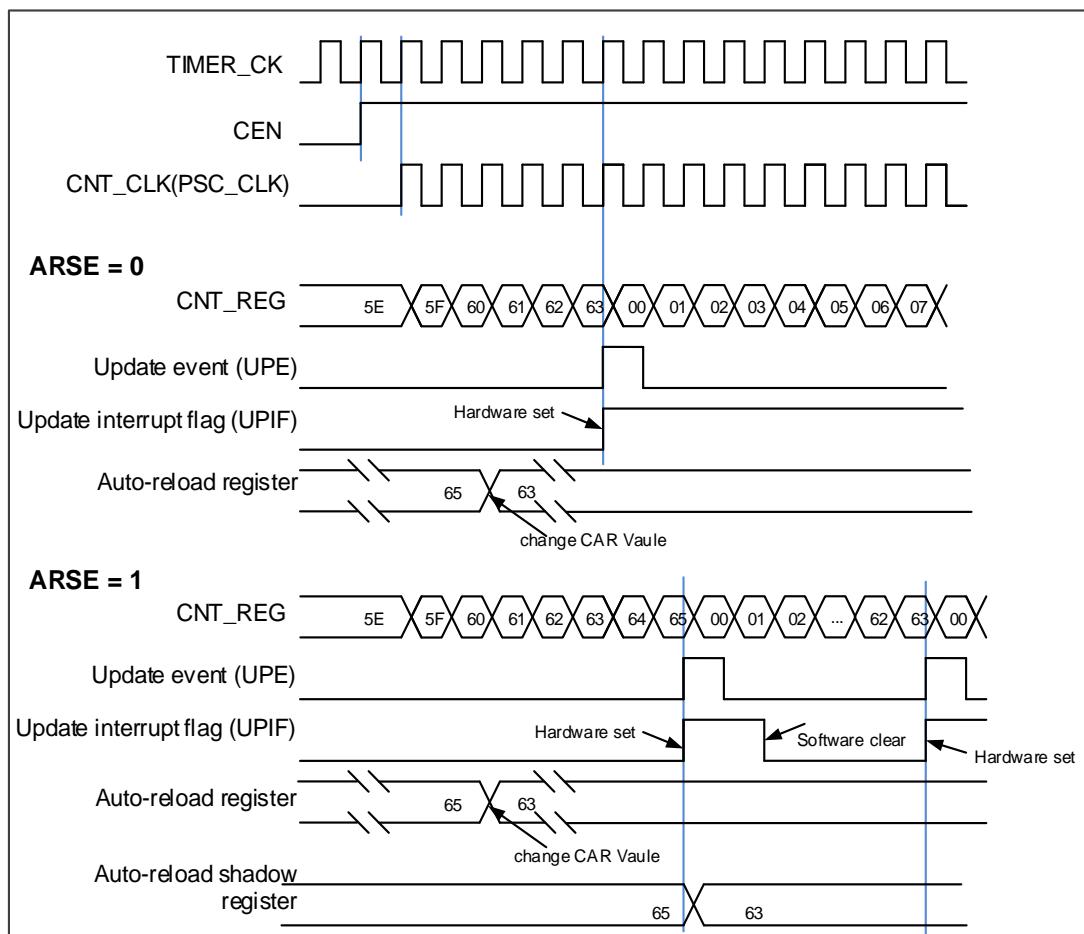
**Figure 15-4. Timing chart of up counting mode, PSC=0/1**


Figure 15-5. Timing chart of up counting mode, change TIMERx\_CAR ongoing



### Down counting mode

In this mode, the counter counts down continuously from the counter reload value, which is defined in the TIMERx\_CAR register, in a count-down direction. Once the counter reaches 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 counter 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 an update event will be generated.

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 register, auto reload register, prescaler register) are updated.

[Figure 15-6. Timing chart of down counting mode, PSC=0/1](#) and [Figure 15-7. Timing chart of down counting mode, change TIMERx CAR ongoing](#) show some examples of the counter behavior in different clock frequencies when TIMERx\_CAR = 0x63.

Figure 15-6. Timing chart of down counting mode, PSC=0/1

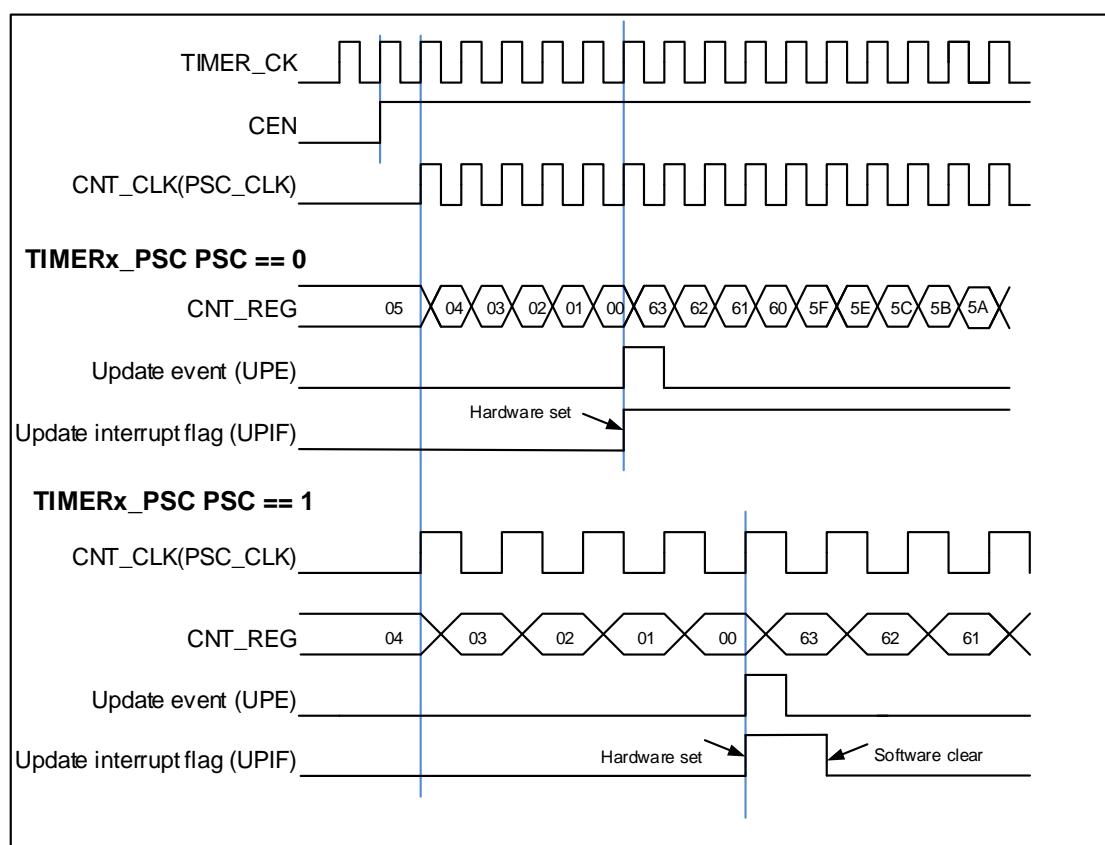
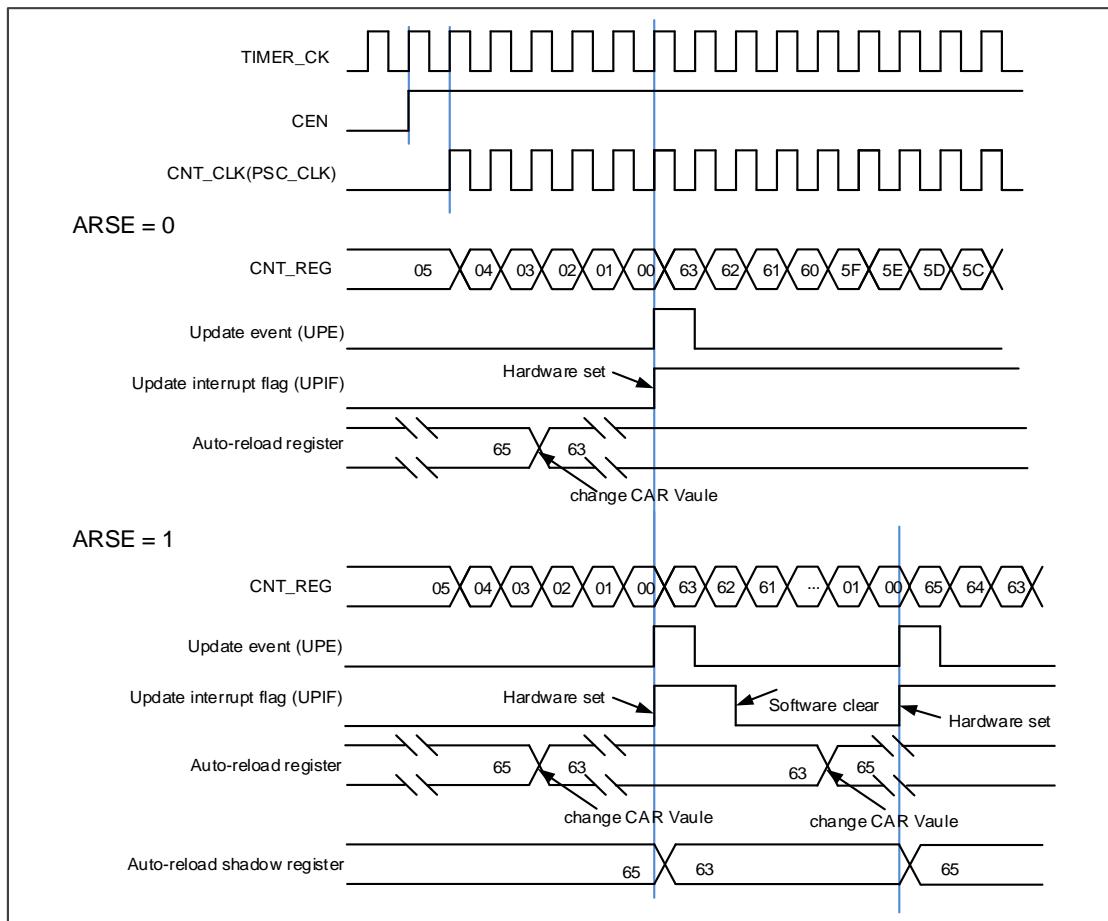


Figure 15-7. Timing chart of down counting mode, change TIMERx\_CAR ongoing



### 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 (TIMERx\_CREP-1) in the count-up direction and generates an underflow event when the counter counts to 1 in the count-down direction. The counting direction bit DIR in the TIMERx\_CTL0 register is read-only and indicates the counting direction when in the center-aligned counting 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 generate an update event irrespective of whether the counter is counting up or down in the center-aligned counting mode.

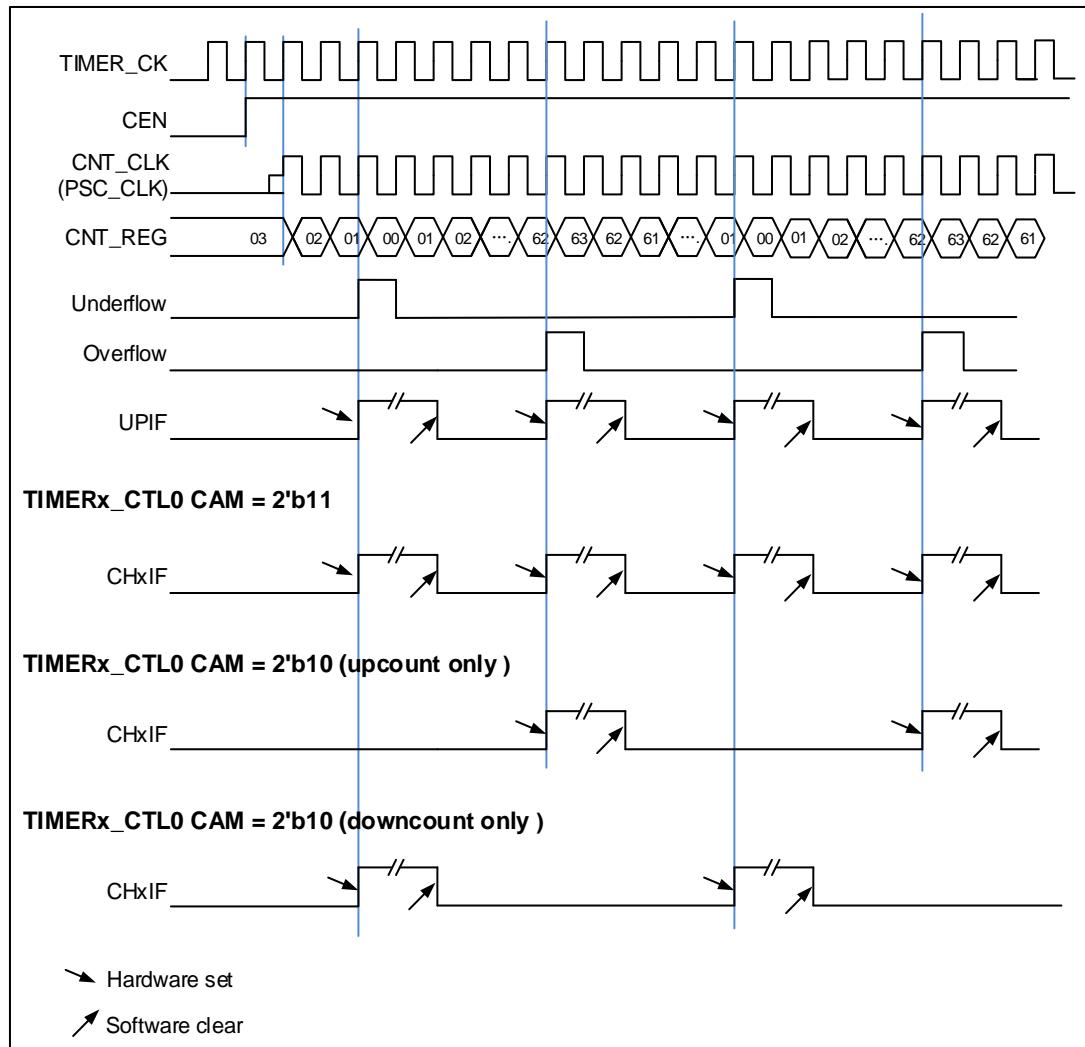
The UPIF bit in the TIMERx\_INTF register will be set to 1 either when an underflow event or an overflow event occurs. While the CHxIF bit is associated with the value of CAM in TIMERx\_CTL0. The details refer to [Figure 15-8. Timing chart of center-aligned counting mode](#).

If the UPDIS bit in the TIMERx\_CTL0 register is set, the update event is disabled.

When an update event occurs, all the registers (repetition counter register, auto-reload register, prescaler register) are updated.

**Figure 15-8. Timing chart of center-aligned counting mode** show some examples of the counter behavior when TIMERx\_CAR=0x63. TIMERx\_PSC=0x0

**Figure 15-8. Timing chart of center-aligned counting mode**



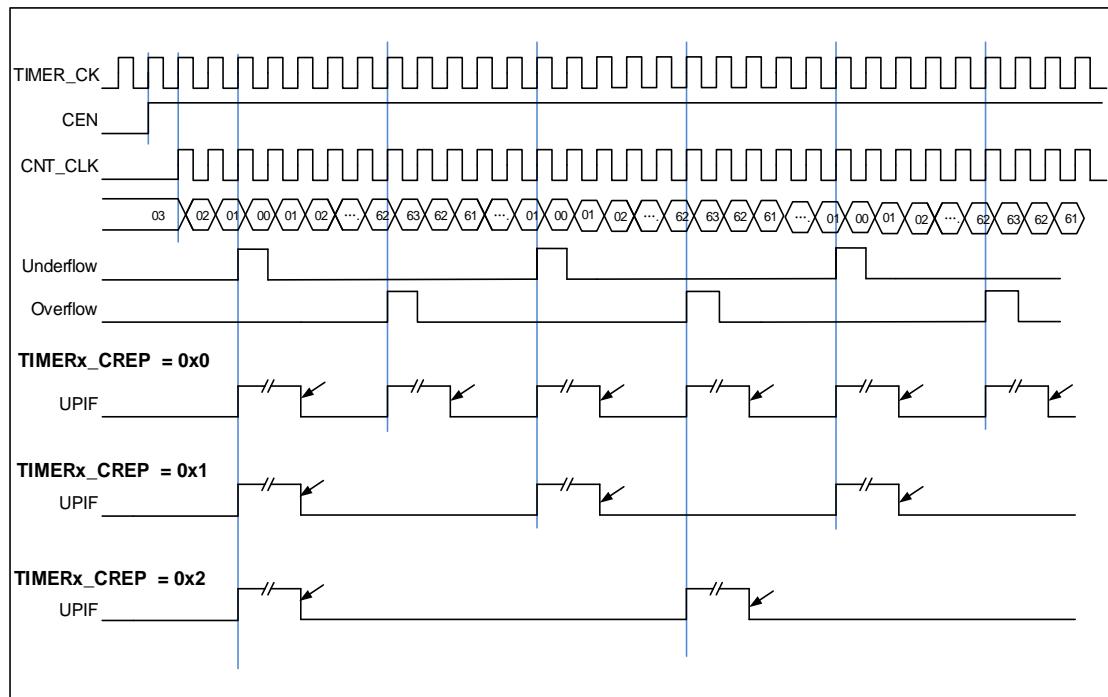
## Repetition counter

Repetition counter is used to generate the update event or update the timer registers only after a given number ( $N+1$ ) cycles of the counter, where  $N$  is the value of CREP bit 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 is 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 counter timing chart of center-aligned counting mode**



**Figure 15-10. Repetition counter timing chart of up counting mode**

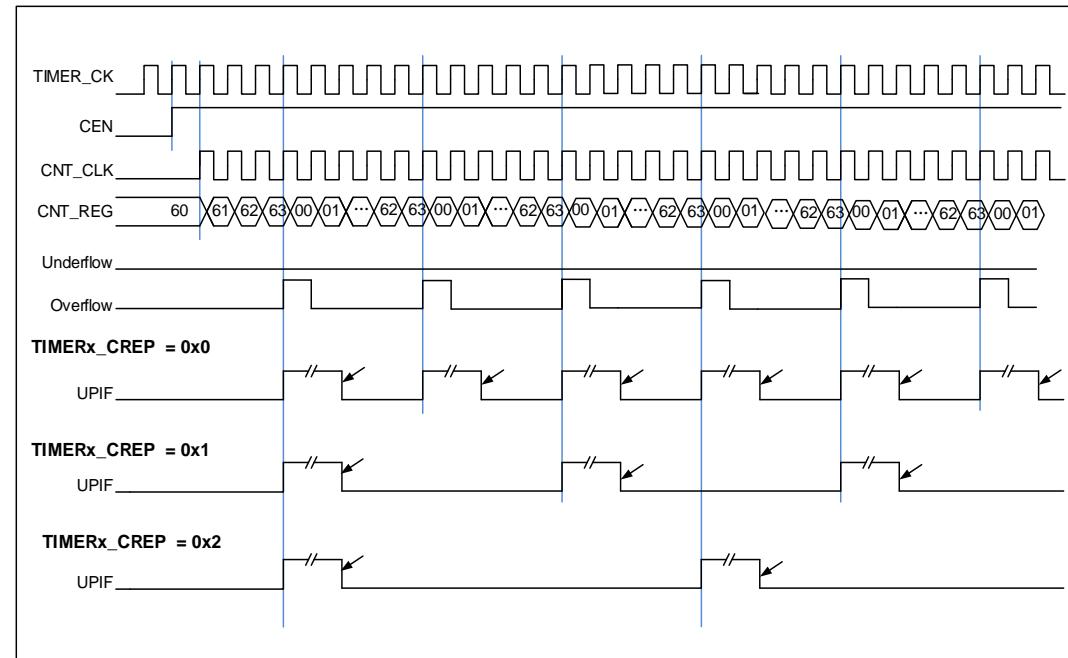
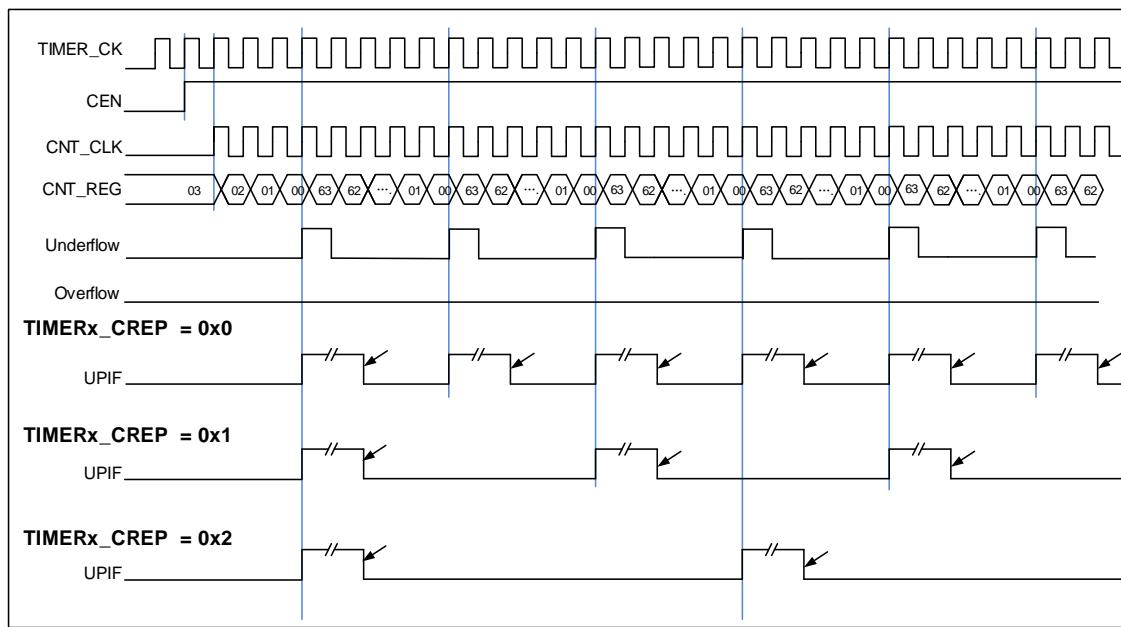


Figure 15-11. Repetition counter timing chart of down counting mode



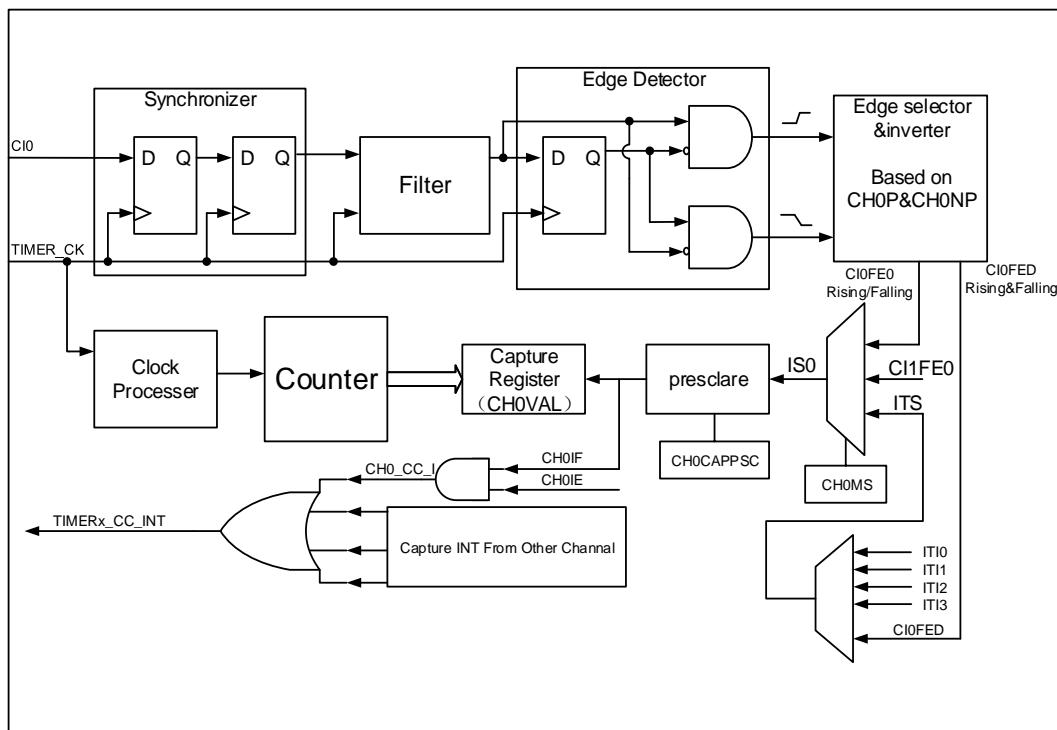
### Capture/compare channels

The advanced timer has four independent channels which can be used as capture inputs or compare outputs. Each channel is built around a channel capture compare register including an input stage, a channel controller and an output stage.

- Input capture mode

Input 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 it is enabled when CHxIE=1.

Figure 15-12. Input capture logic



The input signals of channelx (CIx) can be the TIMER<sub>x</sub>\_CHx signal or the XOR signal of the TIMER<sub>x</sub>\_CH0, TIMER<sub>x</sub>\_CH1 and TIMER<sub>x</sub>\_CH2 signals. First, the input signal of channel (CIx) is synchronized to TIMER\_CK signal, and then sampled by a digital filter to generate a filtered input signal. Then through the edge detector, the rising or falling edge is detected by configuring CHxP bit. The input capture signal can also be selected from the input signal of other channel or the internal trigger signal by configuring CHxMS bits. The IC prescaler makes several input events generate one effective capture event. On the capture event, TIMER<sub>x</sub>\_CHxCV will store the value of counter.

So the process can be divided to several steps as below:

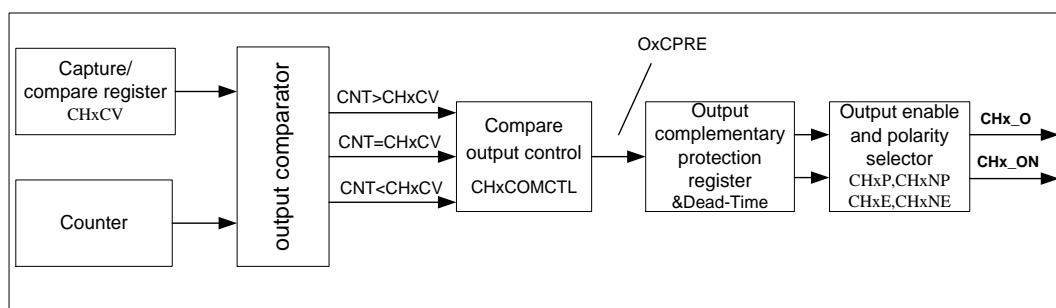
- Step1:** Filter configuration (CHxCAPFLT in TIMER<sub>x</sub>\_CHCTL0).  
Based on the input signal and quality of requested signal, configure compatible CHxCAPFLT.
- Step2:** Edge selection (CHxP/CHxNP in TIMER<sub>x</sub>\_CHCTL2).  
Rising edge or falling edge, choose one by configuring CHxP/CHxNP bits.
- Step3:** Capture source selection (CHxMS in TIMER<sub>x</sub>\_CHCTL0).  
As soon as selecting one input capture source by CHxMS, the channel must be set to input mode (CHxMS! =0x0) and TIMER<sub>x</sub>\_CHxCV cannot be written any more.
- Step4:** Interrupt enable (CHxIE and CHxDEN in TIMER<sub>x</sub>\_DMAINTEN).  
Enable the related interrupt to get the interrupt and DMA request.
- Step5:** Capture enable (CHxEN in TIMER<sub>x</sub>\_CHCTL2).
- Result:** When the wanted input signal is captured, TIMER<sub>x</sub>\_CHxCV will be set by counter's value and CHxIF is asserted. If the CHxIF is 1, the CHxOF will also be asserted. The interrupt and DMA request will be asserted or not based on the configuration of CHxIE and CHxDEN in TIMER<sub>x</sub>\_DMAINTEN.

**Direct generation:** A DMA request or interrupt is generated by setting CHxG directly.

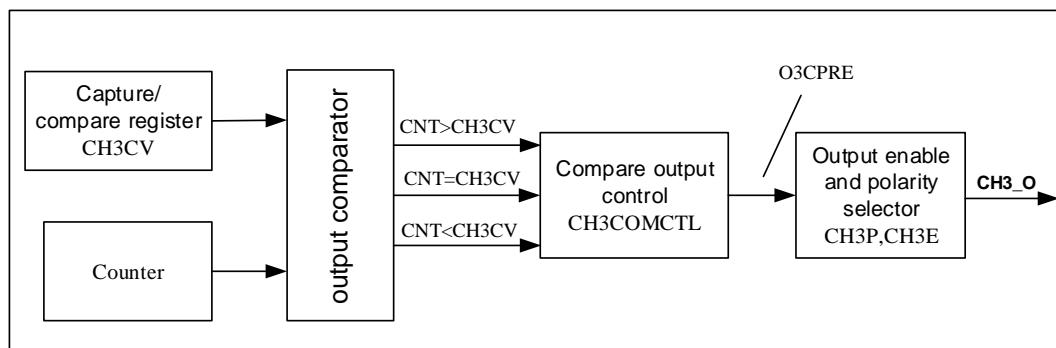
The input capture mode can be also used for pulse width measurement from signals on the TIMERx\_CHx pins. For example, PWM signal connects to CI0 input. Select CI0 as channel 0 capture signals by setting CH0MS to 2'b01 in the channel control register (TIMERx\_CHCTL0) and set capture on rising edge. Select CI0 as channel 1 capture signal by setting CH1MS to 2'b10 in the channel control register (TIMERx\_CHCTL0) and set capture on falling edge. The counter is set to restart mode and is restarted on channel 0 rising edge. Then the TIMERX\_CH0CV can measure the PWM period and the TIMERx\_CH1CV can measure the PWM duty cycle.

- Output compare mode

**Figure 15-13. Output compare logic (with complementary output, x=0,1,2)**



**Figure 15-14. Output compare logic (CH3\_O)**



[Figure 15-13. Output compare logic \(with complementary output, x=0,1,2\)](#) and [Figure 15-14. Output compare logic \(CH3\\_O\)](#) show the logic circuit of output compare mode. The relationship between the channel output signal CHx\_O/CHx\_ON and the OxCPRE signal (more details refer to [Channel output prepare signal](#)) is described as below: The active level of O0CPRE is high, the output level of CH0\_O/CH0\_ON depends on OxCPRE signal, CHxP/CHxNP bit and CH0E/CH0NE bit (please refer to the TIMERx\_CHCTL2 register for more details). For examples,

- 1) Configure CHxP=0 (the active level of CHx\_O is high, the same as OxCPRE), CHxE=1 (the output of CHx\_O is enabled),
  - If the output of OxCPRE is active(high) level, the output of CHx\_O is active(high) level;
  - If the output of OxCPRE is inactive(low) level, the output of CHx\_O is active(low) level.

- 2) Configure CHxNP=0 (the active level of CHx\_ON is low, contrary to OxCPRE), CHxNE=1 (the output of CHx\_ON is enabled),  
If the output of OxCPRE is active(high) level, the output of CHx\_O is active(low) level;  
If the output of OxCPRE is inactive(low) level, the output of CHx\_O is active(high) level.

When CH0\_O and CH0\_ON are output at the same time, the specific outputs of CH0\_O and CH0\_ON are related to the relevant bits (ROS, IOS, POE and DTCFG bits) in the TIMERx\_CCHP register. Please refer to Complementary outputs for more details.

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 TIMERx\_CHxCV 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 TIMERx\_CHxCV register, the CHxIF bit will be set and the channel (n) interrupt is generated if CHxIE = 1. And the DMA request will be asserted, if CxDCE=1.

So the process can be divided into 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 polarity by CHxP/CHxNP
- Enable the output by CHxEN

**Step3:** Interrupt/DMA-request enable configuration by CHxIE/CHxDEN.

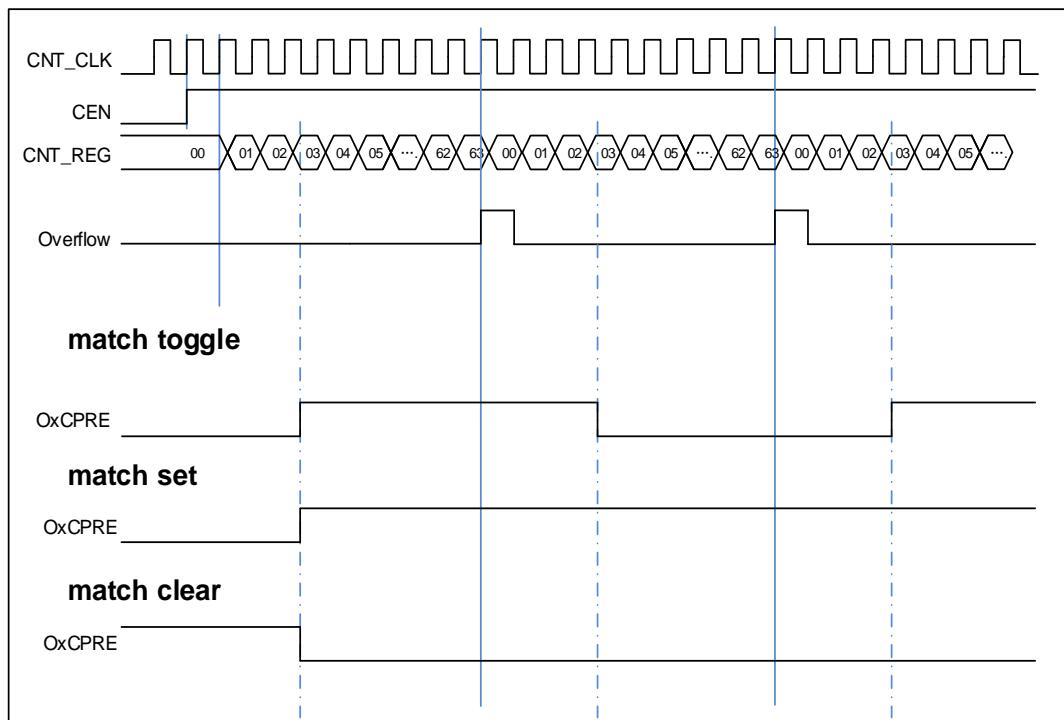
**Step4:** Compare output timing configuration by TIMERx\_CAR and TIMERx\_CHxCV

The TIMERx\_CHxCV can be changed ongoing to meet the expected waveform.

**Step5:** Start the counter by configuring CEN to 1.

[Figure 15-15. Output-compare in three modes](#) show the three compare modes toggle/set/clear. CAR=0x63, CHxVAL=0x3

Figure 15-15. Output-compare in three modes



## PWM mode

In the PWM output mode (by setting the CHxCOMCTL bit to 3'b110 (PWM mode 0) or to 3'b111(PWM mode 1)), the channel can generate PWM waveform according to the TIMERx\_CAR registers and TIMERx\_CHxCV registers.

Based on the counter mode, PWM can also be divided into EAPWM (Edge-aligned PWM) and CAPWM (Center-aligned PWM).

The EAPWM's period is determined by TIMERx\_CAR and the duty cycle is determined by TIMERx\_CHxCV. [Figure 15-16. Timing chart of EAPWM](#) shows the EAPWM output and interrupts waveform.

The CAPWM's period is determined by 2\*TIMERx\_CAR, the duty cycle is determined by 2\*TIMERx\_CHxCV. [Figure 15-17. Timing chart of CAPWM](#) shows the CAPWM output and interrupts waveform.

In up counting mode, if the value of TIMERx\_CHxCV is greater than the value of TIMERx\_CAR, the output will be always inactive in PWM mode 0 (CHxCOMCTL=3'b110). And if the value of TIMERx\_CHxCV is greater than the value of TIMERx\_CAR, the output will be always active in PWM mode 1 (CHxCOMCTL=3'b111).

Figure 15-16. Timing chart of EAPWM

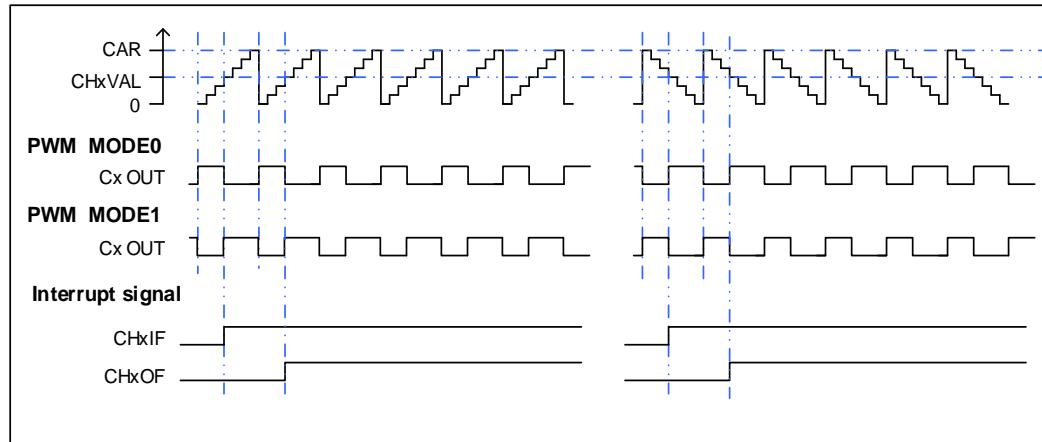
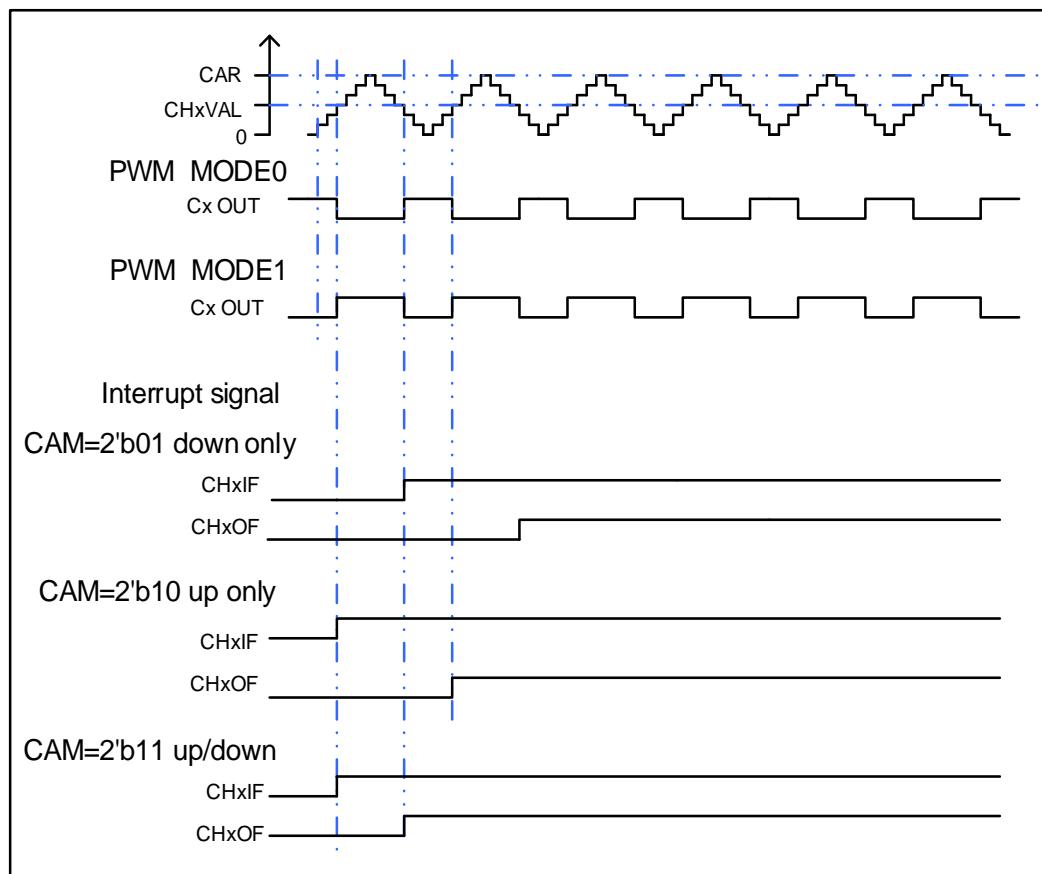


Figure 15-17. Timing chart of CAPWM



### Channel output prepare signal

As is shown in [Figure 15-13. Output compare logic \(with complementary output, x=0,1,2\)](#), when TIMERx is configured in compare match output mode, a middle signal which is OxCPRE signal (Channel x output prepare signal) will be generated before the channel outputs signal. The OxCPRE signal type is defined by configuring the CHxCOMCTL bit. The OxCPRE signal

has several types of output function. These include keeping the original level by configuring the CHxCOMCTL field to 0x00, setting to high by configuring the CHxCOMCTL field to 0x01, setting to low by configuring the CHxCOMCTL field to 0x02 or toggling signal by configuring the CHxCOMCTL field to 0x03 when the counter value matches the content of the TIMERx\_CHxCV register.

The PWM mode 0/PWM mode 1 output is another output type of OxCPRE which is setup by configuring 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. Refer to the definition of relative bit for more details.

Another special function of the OxCPRE signal is a forced output which can be achieved by configuring the CHxCOMCTL field to 0x04/0x05. The output can be forced to an inactive/active level irrespective of the comparison condition between the values of the counter and the TIMERx\_CHxCV.

Configure the CHxCOMCEN bit to 1 in the TIMERx\_CHCTL0 register, the OxCPRE signal can be forced to 0 when the ETIFP signal derived from the external ETI pin is set to a high level. The OxCPRE signal will not return to its active level until the next update event occurs.

### Complementary outputs

Function of complementary is for a pair of channels, CHx\_O and CHx\_ON, the 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, the POEN, ROS and IOS bits in the TIMERx\_CCHP register, ISOx and ISOxN bits in the TIMERx\_CTL1 register. The output 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 CHx_O/CHx_ON output disable. If clock is enable: CHx_O = ISOx CHx_ON = ISOxN	
			1	0		
				1		
		1	0	0	CHx_O = CHxP CHx_ON = CHxNP CHx_O/CHx_ON output disable.	
				1	CHx_O = CHxP CHx_ON = CHxNP CHx_O/CHx_ON output enable. If clock is enable: CHx_O = ISOx CHx_ON = ISOxN	
			1	0		
				1		
1	0/1	0	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

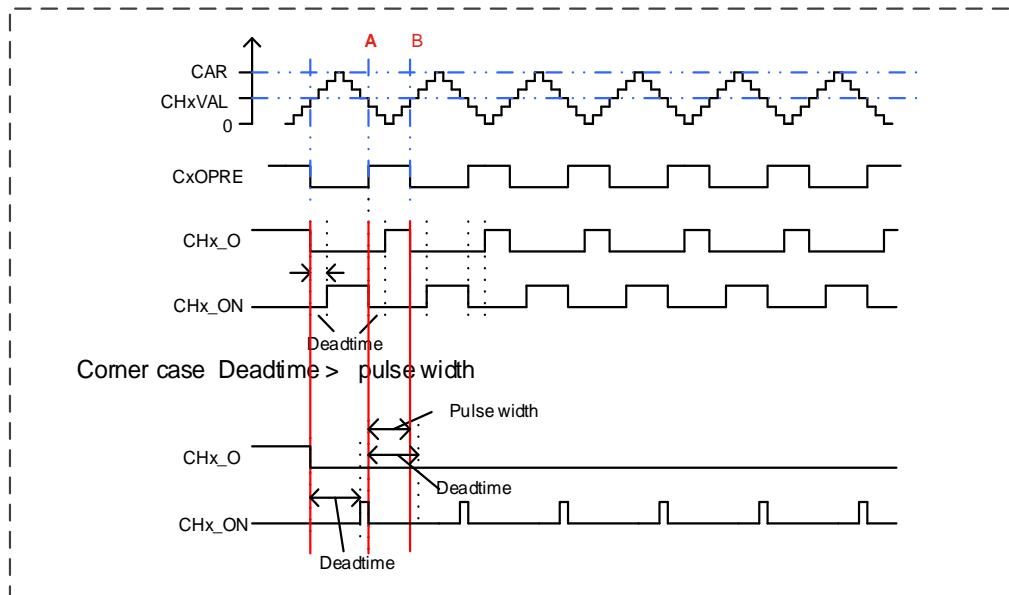
The dead time insertion is enabled when both CHxEN and CHxNEN are configured to 1'b1, it is also necessary to configure POEN to 1. The field named DTCFG defines the dead time delay that can be used for all channels except channel 3. Refer to the TIMERx\_CCHP register for details about the delay time.

The dead time delay insertion ensures that two complementary signals are not active at the same time.

When the channelx match event (TIMERx counter = CHxVAL) occurs, OxCPRE will be toggled in PWM mode 0. At point A in [Figure 15-18. Complementary output with dead time insertion](#) CHx\_O signal remains at the low level until the end of the dead time delay, while CHx\_ON signal will be cleared at once. Similarly, at point B when the channelx match event (TIMERx counter = CHxVAL) occurs again, OxCPRE is cleared, and CHx\_O signal will be cleared at once, while CHx\_ON signal remains at the low level 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 duty cycle of the CHx\_O signal, then the CHx\_O signal is always inactive. (as show in the [Figure 15-18. Complementary output with dead time insertion](#))

**Figure 15-18. Complementary output with dead time insertion**



## Break function

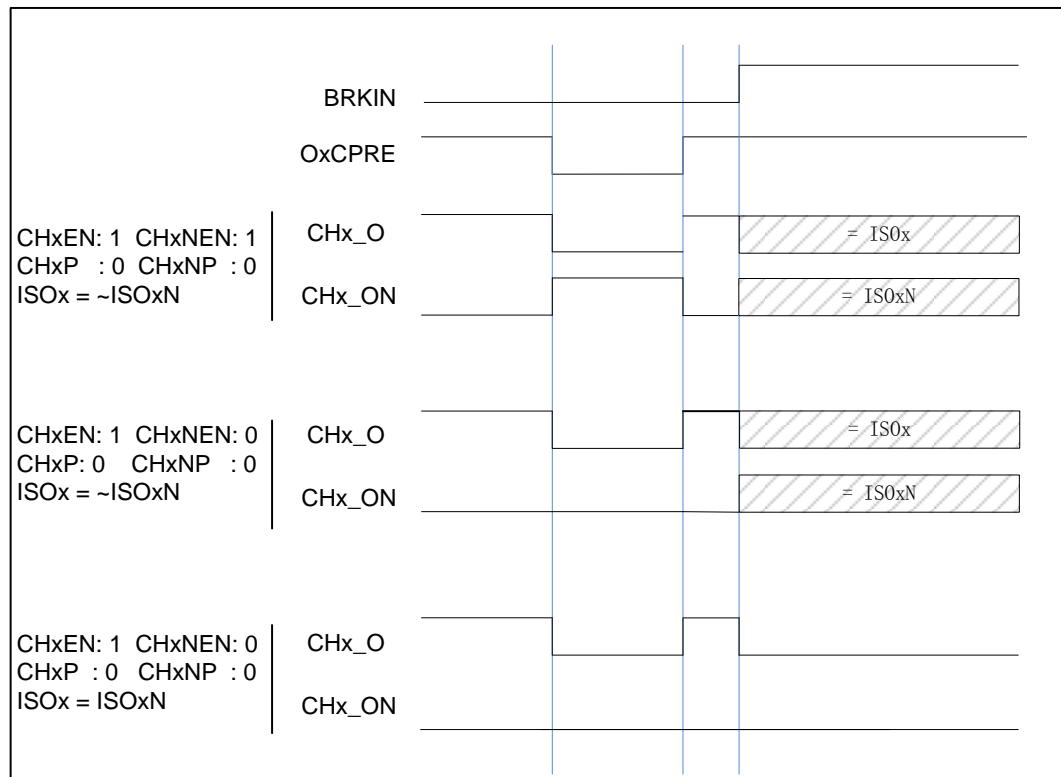
In this function, CHx\_O and CHx\_ON are controlled by the POEN, IOS and ROS bits in the TIMERx\_CCHP register, ISOx and ISOxN bits in the TIMERx\_CTL1 register. In any case, CHx\_O and CHx\_ON signals cannot be set to active level at the same time. The break sources are input break pin and HXTAL stuck event which is generated by Clock Monitor

(CKM) in RCU. The break function is enabled by setting the BRKEN bit in the TIMERx\_CCHP register. The break input polarity is configured by the BRKP bit in TIMERx\_CCHP register.

When a break occurs, the POEN bit is cleared asynchronously. As soon as POEN is 0, the level of the CHx\_O and CHx\_ON outputs are determined by the ISOx and ISOxN bits in the TIMERx\_CTL1 register. If IOS is 0, the timer releases the enable output, otherwise, the enable output remains high. The complementary outputs are first in the reset state, and then the dead time generator is reactivated 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 will be set. If BRKIE is 1, an interrupt will be generated.

**Figure 15-19. Output behavior of the channel 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 TIMERx\_CH0 and TIMERx\_CH1 pins respectively to interact with each other to generate the counter value. Setting SMC=0x01, 0x02, or 0x03 to select that the counting direction of timer is determined only by the CI0, only by the CI1, or by the CI0 and the CI1. The DIR bit is modified by hardware automatically during the voltage level change of each direction selection source. The mechanism of changing the counter direction is shown in [Table 15-3. Counting direction versus encoder signals](#). The quadrature decoder can be regarded as an external clock with a direction selection. This means that the counter counts continuously from 0 to the counter-reload value. Therefore, users must configure the TIMERx\_CAR register

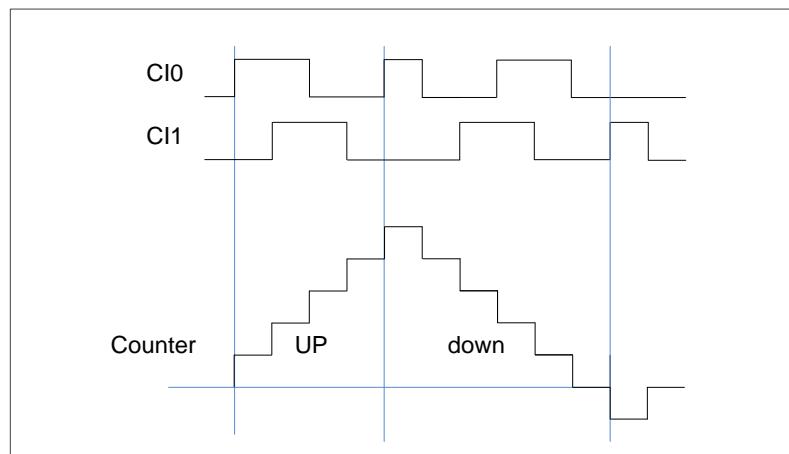
before the counter starts to count.

**Table 15-3. Counting direction versus encoder signals**

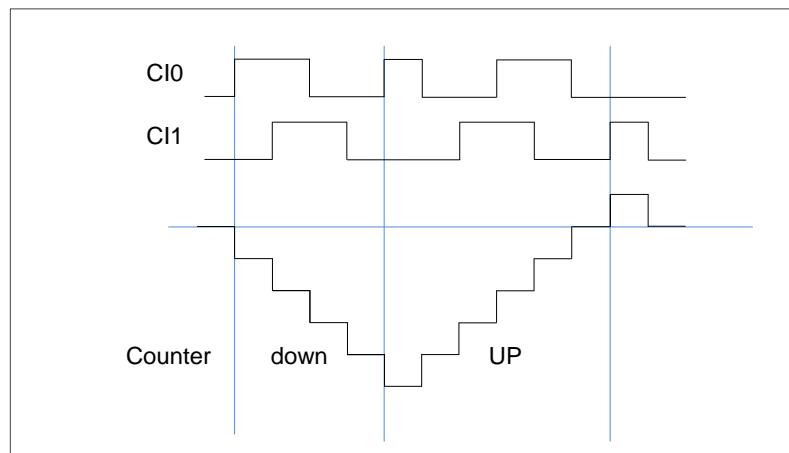
Counting mode	Level	CI0FE0		CI1FE1	
		Rising	Falling	Rising	Falling
CI0 only counting	CI1FE1=High	Down	Up	-	-
	CI1FE1=Low	Up	Down	-	-
CI1 only counting	CI0FE0=High	-	-	Up	Down
	CI0FE0=Low	-	-	Down	Up
CI0 and CI1 counting	CI1FE1=High	Down	Up	X	X
	CI1FE1=Low	Up	Down	X	X
	CI0FE0=High	X	X	Up	Down
	CI0FE0=Low	X	X	Down	Up

Note: "-" means "no counting"; "X" means impossible.

**Figure 15-20. Example of counter operation in encoder interface mode**



**Figure 15-21. 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 15-22. Hall sensor is used to BLDC motor\*\*](#) show how to connect. And we can see we need two timers. First TIMER\_in (Advanced/GeneralL0 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/GeneralL0 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.

TIMER\_in (TIMER2) -> TIMER\_out (TIMER0 ITI2)

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 15-22. Hall sensor is used to BLDC motor**

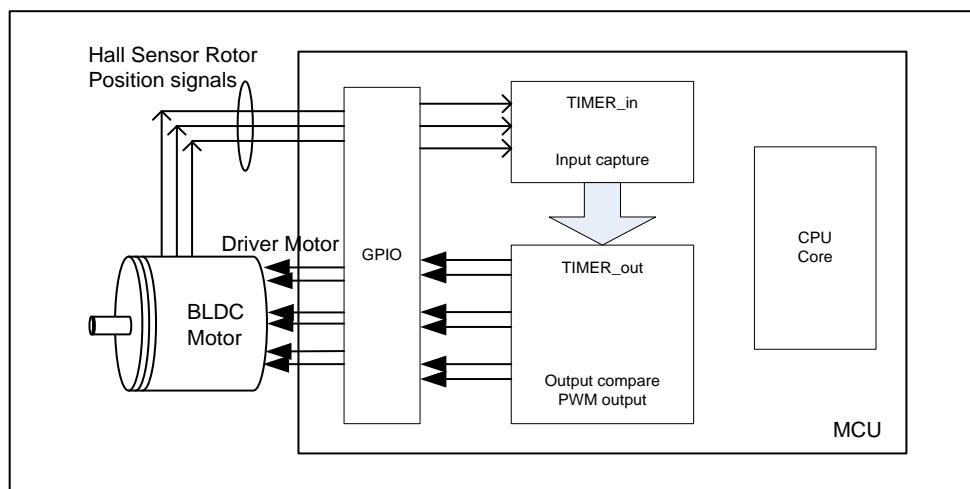
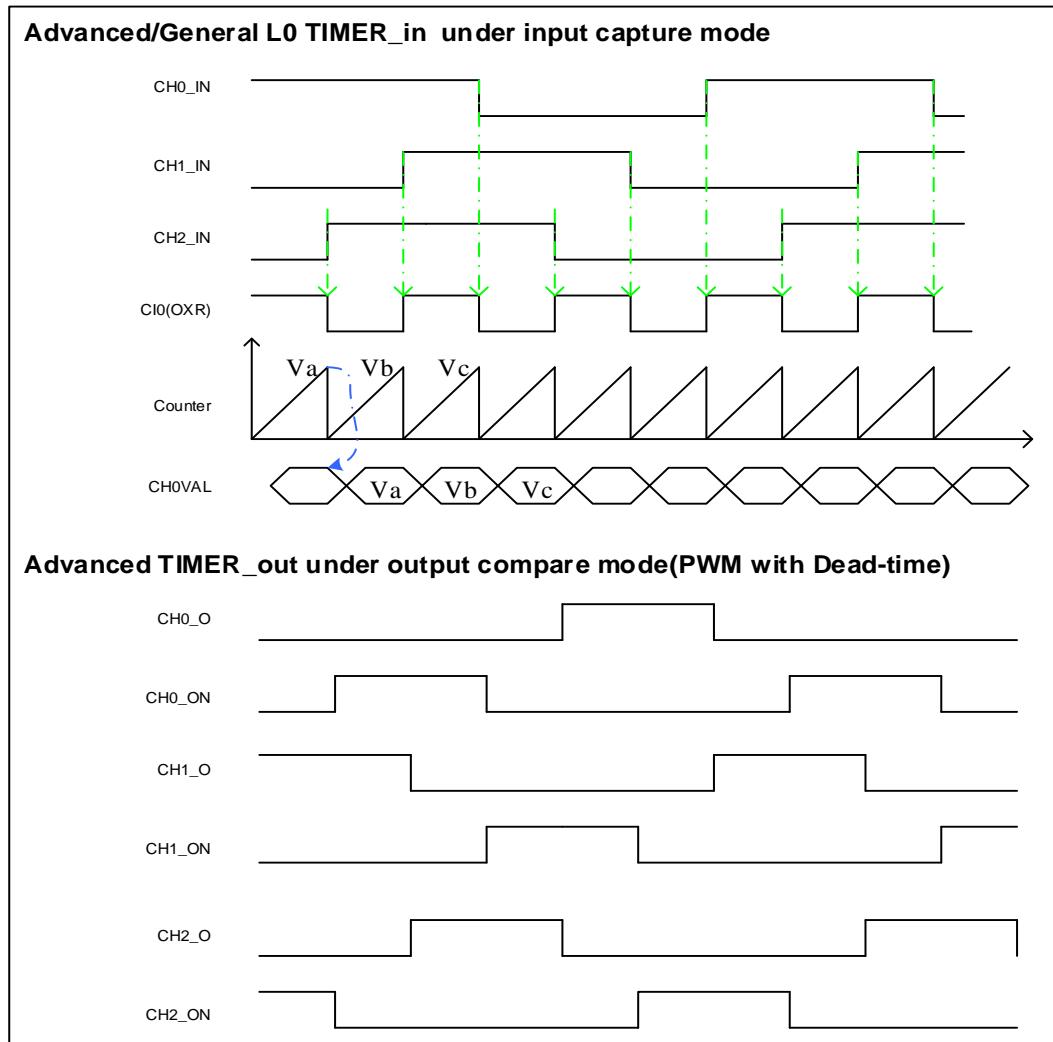


Figure 15-23. Hall sensor timing between two timers

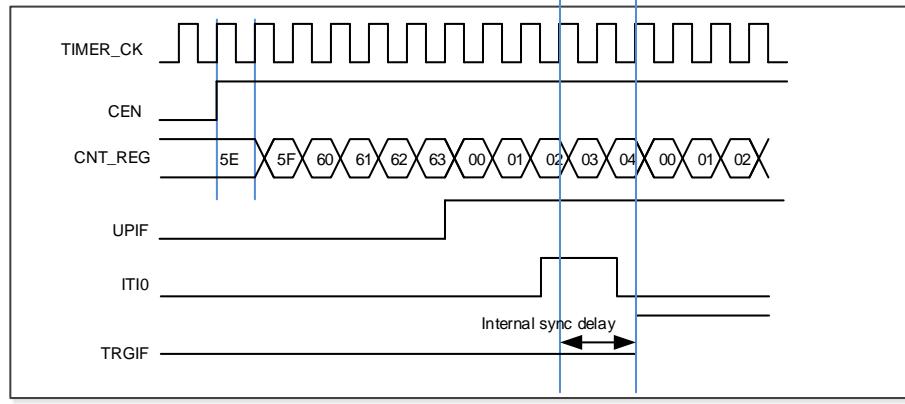
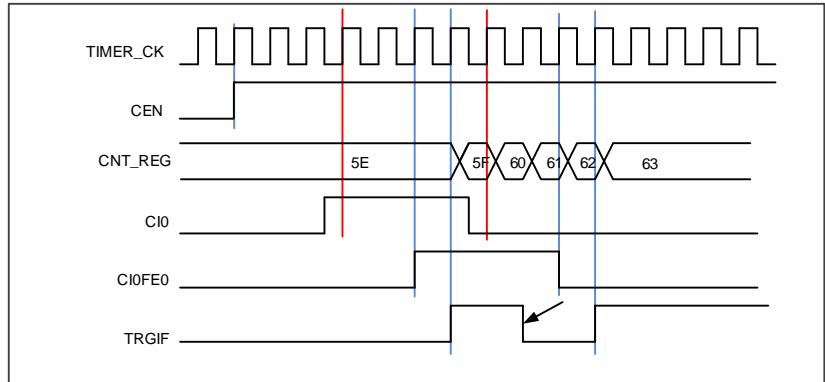


### Slave controller

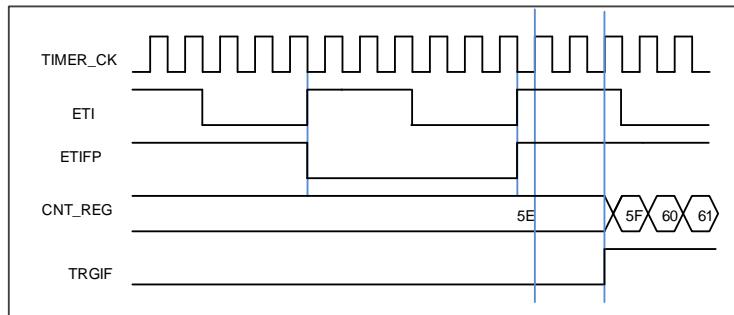
The TIMERx can be synchronized with a trigger in several modes including restart mode, pause mode and event mode which is selected by the SMC[2:0] bits in the TIMERx\_SMCFG register. The input trigger of these modes can be selected by the TRGS[2:0] bits in the TIMERx\_SMCFG register.

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	If CI0FE0 or CI1FE1 is selected as the trigger source, configure the CHxP and CHxNP for the polarity selection and inversion.	For the ITIx, no filter and prescaler can be used. For the Clx, filter can be used by configuring CHxCAPFLT, no prescaler can be used.

	Mode Selection	Source Selection	Polarity Selection	Filter and Prescaler
		110: CI1FE1 111: ETIPP	If ETIPP is selected as the trigger source, configure the ETP for polarity selection and inversion.	For the ETIPP, filter can be used by configuring ETFC and prescaler can be used by configuring ETPSC.
	<b>Restart mode</b>  The counter will be cleared and restart when a rising edge of trigger input comes.	TRGS[2:0] = 3'b000 ITI0 is selected.	For ITI0, no polarity selector can be used.	For the ITI0, no filter and prescaler can be used.
<b>Figure 15-24. Restart mode</b>				
Exam1				
	<b>Pause mode</b>  The counter will be paused when the trigger input is low, and it will start when the trigger input is high.	TRGS[2:0]=3'b101 CI0FE0 is selected.	TI0S=0 (Non-xor) [CH0NP=0, CH0P=0] CI0FE0 does not invert. The capture event will occur on the rising edge only.	Filter is bypassed in this example.
<b>Figure 15-25. Pause mode</b>				
Exam2				

	Mode Selection	Source Selection	Polarity Selection	Filter and Prescaler
	<b>Event mode</b> The counter will start to count when a rising edge of trigger input comes. TRGS[2:0] =3'b111 ETIFP is selected.	ETP = 0, the polarity of ETI does not change.	ETPSC = 1, ETI is divided by 2. ETFC = 0, ETI does not filter.	
<b>Figure 15-26. Event mode</b>				

**Exam3**


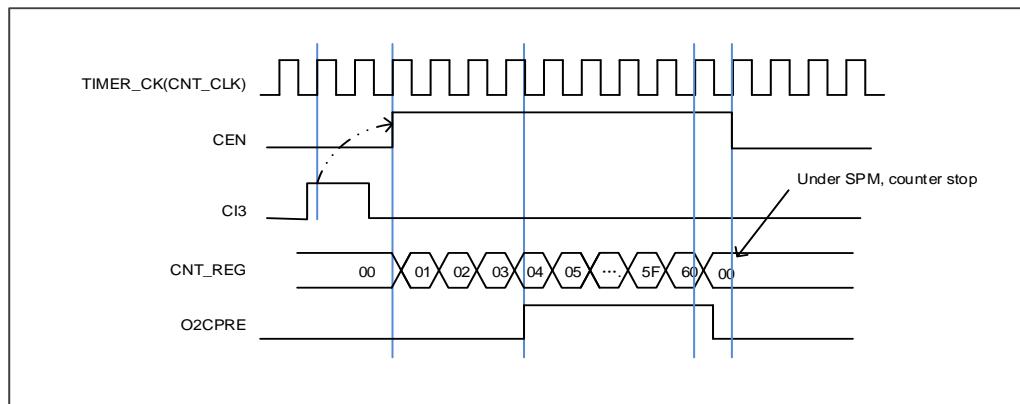
### Single pulse mode

Single pulse mode is enabled by setting SPM in **TIMERx\_CTL0**. If SPM is set, the counter will be cleared and stopped automatically when the next update event occurs. In order to get a pulse waveform, the **TIMERx** is configured to PWM mode or compare mode by **CHxCOMCTL**.

Once the timer is set to the single pulse mode, it is not necessary to configure the timer enable bit CEN in the **TIMERx\_CTL0** register to 1 to enable the counter. Setting the CEN bit to 1 or a trigger signal 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 by software, the counter will be stopped and its value will be 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 active edge of trigger which sets the CEN bit to 1 will enable the counter. However, there exists 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 **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 the PWM mode 0 or PWM mode 1 and the trigger source is derived from the trigger signal.

**Figure 15-27. Single pulse mode TIMERx\_CHxCV=0x04, TIMERx\_CAR=0x60**

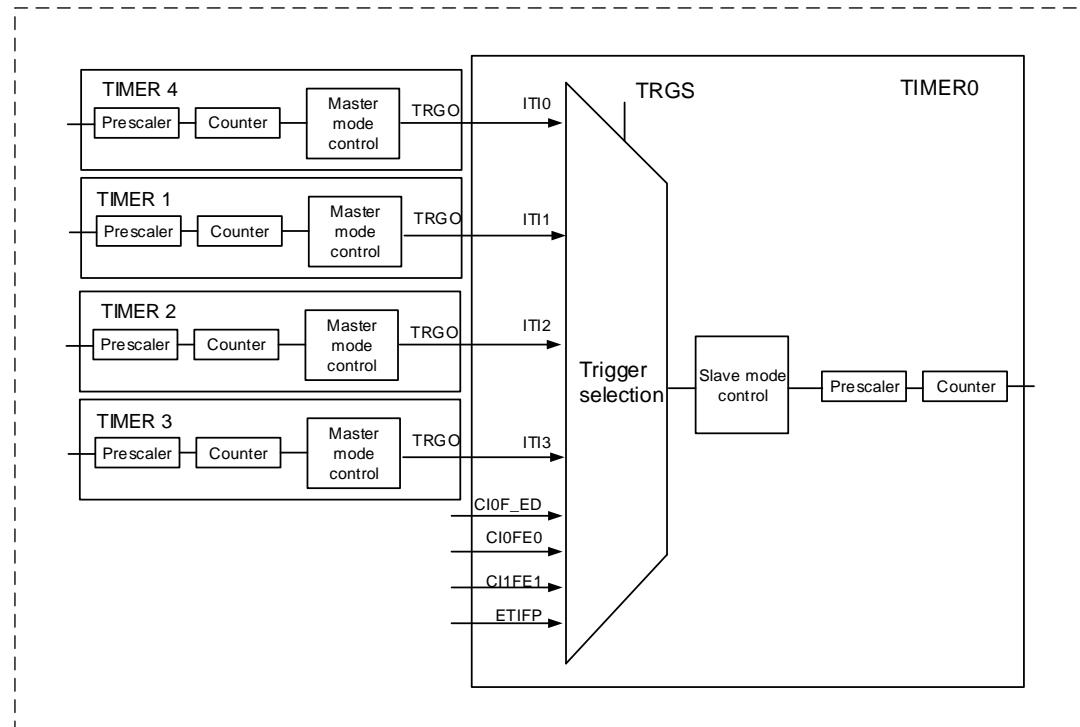


### Timers interconnection

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 15-28. TIMER0 Master/Slave mode timer example** shows the timer0 trigger selection when it is configured in slave mode.

**Figure 15-28. TIMER0 Master/Slave mode timer example**



Other interconnection examples:

- TIMER2 as prescaler for TIMER0

We configure TIMER2 as a prescaler for TIMER0. Refer to [Figure 15-28. TIMER0 Master/Slave mode timer example](#) for connections. Do as follow:

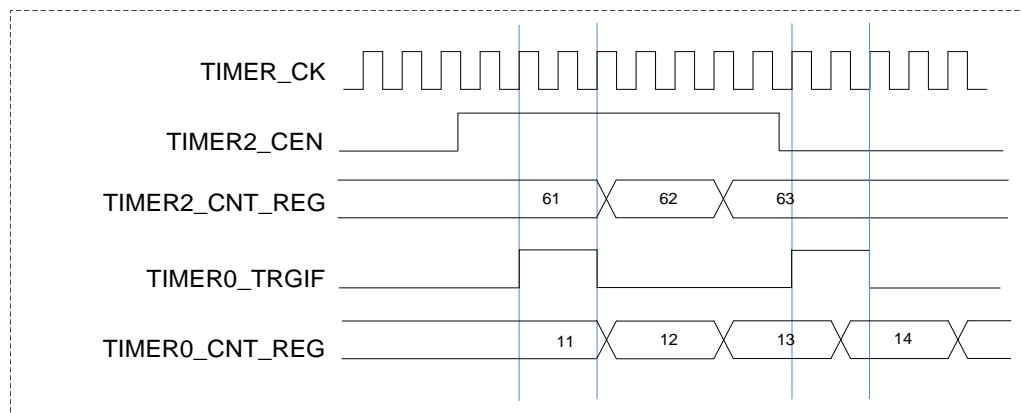
1. Configure TIMER2 in master mode and select its update event (UPE) as trigger output (MMC=3'b010 in the TIMER2\_CTL1 register). Then TIMER2 drives a periodic signal on each counter overflow.
  2. Configure the TIMER2 period (TIMER2\_CAR registers).
  3. Select the TIMER0 input trigger source from TIMER2 (TRGS=3'b010 in the TIMERx\_SMCFG register).
  4. Configure TIMER0 in external clock mode 0 (SMC=3'b111 in TIMER0\_SMCFG register).
  5. Start TIMER0 by writing '1 in the CEN bit (TIMER0\_CTL0 register).
  6. Start TIMER2 by writing '1 in the CEN bit (TIMER2\_CTL0 register).
- Start TIMER0 with TIMER2's Enable/Update signal

First, we enable TIMER0 with the enable out of TIMER2. Refer to [Figure 15-29. Triggering TIMER0 with enable signal of TIMER2](#). TIMER0 starts counting from its current value on the divided internal clock after trigger by TIMER2 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 TIMER2 master mode to send its enable signal as trigger output(MMC=3'b001 in the TIMER2\_CTL1 register)
2. Configure Timer0 to select the input trigger from TIMER2 (TRGS=3'b010 in the TIMERx\_SMCFG register).
3. Configure Timer0 in event mode (SMC=3'b 110 in TIMERx\_SMCFG register).
4. Start TIMER2 by writing 1 in the CEN bit (TIMER2\_CTL0 register).

**Figure 15-29. Triggering TIMER0 with enable signal of TIMER2**

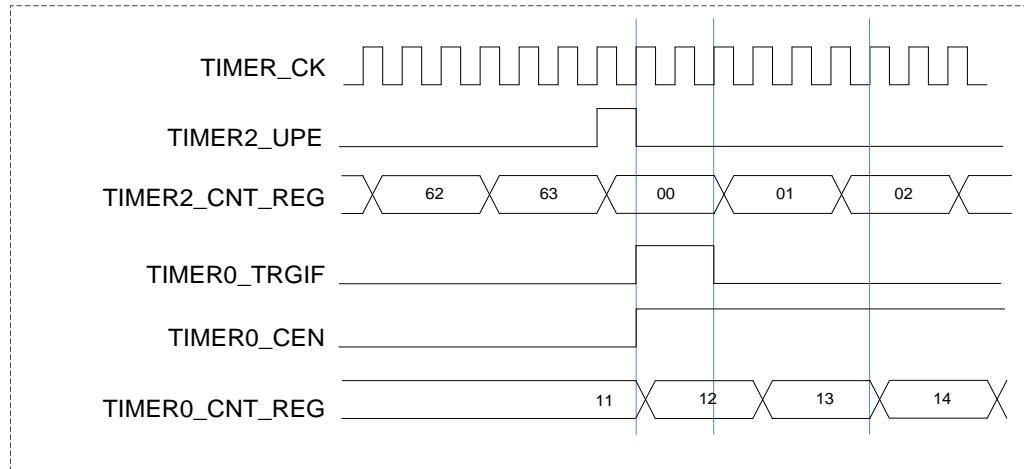


In this example, we also can use update event as trigger source instead of enable signal. Refer to [Figure 15-30. Triggering TIMER0 with update signal of TIMER2](#). Do as follow:

1. Configure TIMER2 in master mode and send its update event (UPE) as trigger output (MMC=3'b010 in the TIMER2\_CTL1 register).

2. Configure the Timer2 period (TIMER2\_CARL registers).
3. Configure TIMER0 to get the input trigger from Timer2 (TRGS=3'b010 in the TIMER0\_SMCFG register).
4. Configure TIMER0 in event mode (SMC=3'b110 in TIMERx\_SMCFG register).
5. Start TIMER2 by writing '1 in the CEN bit (TIMER2\_CTL0 register).

**Figure 15-30. Triggering TIMER0 with update signal of TIMER2**

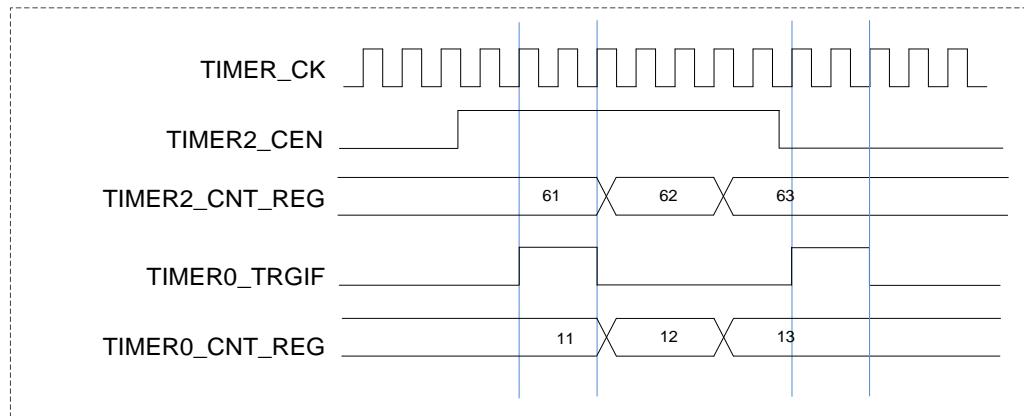


■ Enable TIMER0 count with TIMER2's enable/O0CPRE signal

In this example, we control the enable of TIMER0 with the enable output of TIMER2 .Refer to [Figure 15-31. Pause TIMER0 with enable signal of TIMER2](#). TIMER0 counts on the divided internal clock only when TIMER2 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 TIMER2 input master mode and output enable signal as trigger output (MMC=3'b001 in the TIMER2\_CTL1 register).
2. Configure TIMER0 to get the input trigger from TIMER2 (TRGS=3'b010 in the TIMER0\_SMCFG register).
3. Configure TIMER0 in pause mode (SMC=3'b101 in TIMERx\_SMCFG register).
4. Enable TIMER0 by writing '1 in the CEN bit (TIMER0\_CTL0 register)
5. Start TIMER2 by writing '1 in the CEN bit (TIMER2\_CTL0 register).
6. Stop TIMER2 by writing '0 in the CEN bit (TIMER2\_CTL0 register).

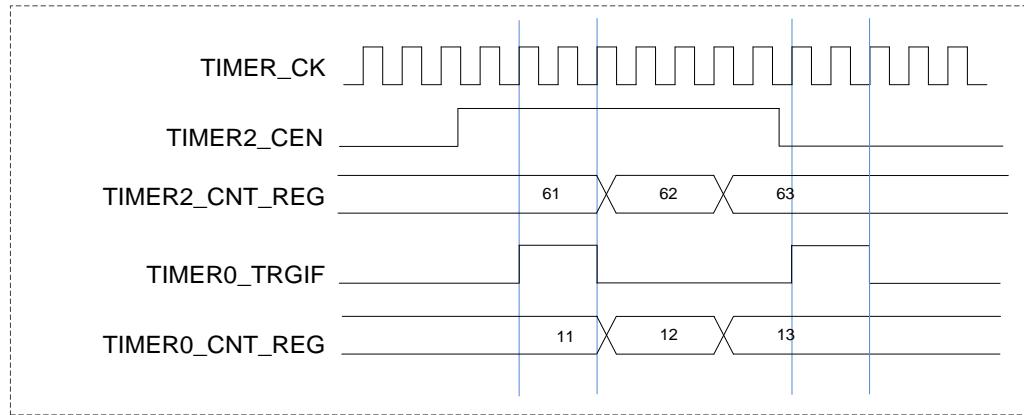
**Figure 15-31. Pause TIMER0 with enable signal of TIMER2**



In this example, we also can use O0CPRE as trigger source instead of enable signal output.  
Do as follow:

1. Configure TIMER2 in master mode and output compare 0 prepare signal (O0CPRE) as trigger output (MMS=3'b100 in the TIMER2\_CTL1 register).
2. Configure the TIMER2 O0CPRE waveform (TIMER2\_CHCTL0 register).
3. Configure TIMER0 to get the input trigger from TIMER2 (TRGS=3'b010 in the TIMERx\_SMCFG register).
4. Configure TIMER0 in pause mode (SMC=3'b101 in TIMER0\_SMCFG register).
5. Enable TIMER0 by writing '1 in the CEN bit (TIMER0\_CTL0 register).
6. Start TIMER2 by writing '1 in the CEN bit (TIMER2\_CTL0 register).

**Figure 15-32. Pause TIMER0 with O0CPREF signal of Timer2**



■ Using an external trigger to start 2 timers synchronously

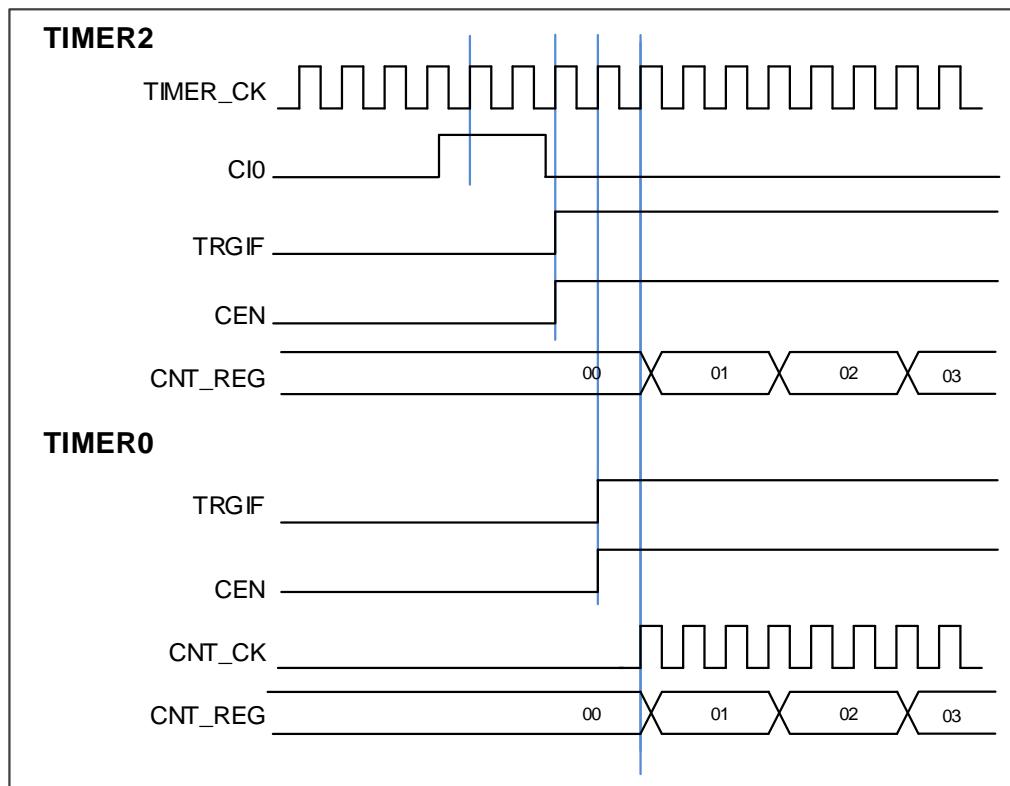
We configure the start of TIMER0 triggered by the enable signal of TIMER2, and TIMER2 is triggered by its CI0 input rises edge. To ensure 2 timers start synchronously, TIMER2 must be configured in Master/Slave mode. Do as follow:

1. Configure TIMER2 in slave mode to get the input trigger from CI0 (TRGS=3'b100 in the TIMER2\_SMCFG register).
2. Configure TIMER2 in event mode (SMC=3'b110 in the TIMER2\_SMCFG register).

3. Configure the TIMER2 in Master/Slave mode by writing MSM=1 (TIMER2\_SMCFG register).
4. Configure TIMER0 to get the input trigger from TIMER2 (TRGS=3'b010 in the TIMER0\_SMCFG register).
5. Configure TIMER0 in event mode (SMC=3'b110 in the TIMER0\_SMCFG register).

When a rising edge occurs on TIMER2's CI0, two timer's counters start counting synchronously on the internal clock and both TRGIF flags are set.

**Figure 15-33. Triggering TIMER0 and TIMER2 with TIMER2's CI0 input**



### Timer DMA mode

Timer DMA mode is the function that configures timer's register by DMA module. The relative registers are **TIMERx\_DMACFG** and **TIMERx\_DMATB**. Corresponding DMA request bit should be asserted to enable DMA request for internal interrupt event. **TIMERx** will send a request to DMA when the interrupt event occurs. DMA is configured to M2P (memory to peripheral) mode and the address of **TIMERx\_DMATB** is configured to **PADDR** (peripheral base address), then DMA will access the **TIMERx\_DMATB**. In fact, **TIMERx\_DMATB** register 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), the timer sends only one DMA request. 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** and **DMATA+0xC** at the next 3 accesses to **TIMERx\_DMATB**. In a word, one-time DMA internal interrupt event asserts, (**DMATC**+1) times

---

request will be sent by TIMERx.

If one more DMA request event occurs, TIMERx will repeat the process above.

### **Timer debug mode**

When the RISC-V core halted, and the TIMERx\_HOLD configuration bit in DBG\_CTL register is set to 1, the TIMERx counter stops.

### 15.1.5. TIMERx registers(x=0)

TIMER0 base address: 0x4001 2C00

#### 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

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: $f_{DTS} = f_{\text{TIMER\_CK}}$ 01: $f_{DTS} = f_{\text{TIMER\_CK}} / 2$ 10: $f_{DTS} = f_{\text{TIMER\_CK}} / 4$ 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: The UPG bit is set The counter generates an overflow or underflow event The slave mode controller generates an update event. 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: 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	ISO3	ISO2N	ISO2	ISO1N	ISO1	ISOON	ISO0	TIOS		MMC[2:0]		DMAS	CCUC	Reserved	CCSE

rw rw

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
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]		ETFC[3:0]		MSM		TRGS[2:0]	Reserved		SMC[2:0]				

rw      rw      rw      rw      rw      rw      rw      rw      rw      rw

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
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 ETIFP signal.</p> <p>0: External clock mode 1 disabled</p> <p>1: External clock mode 1 enabled.</p> <p>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.</p> <p>The external clock input will be ETIFP if external clock mode 0 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-field.</p>
13:12	ETPSC[1:0]	<p>External trigger prescaler</p> <p>The frequency of external trigger signal ETIFP 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 ETIFP frequency.</p> <p>00: Prescaler disable</p> <p>01: ETIFP frequency will be divided by 2</p> <p>10: ETIFP frequency will be divided by 4</p> <p>11: ETIFP 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 ETIFP signal and the length of the digital filter applied to ETIFP.</p> <p>0000: Filter disable. <math>f_{SAMP} = f_{DTS}</math>, <math>N=1</math>.</p> <p>0001: <math>f_{SAMP} = f_{TIMER\_CK}</math>, <math>N=2</math>.</p> <p>0010: <math>f_{SAMP} = f_{TIMER\_CK}</math>, <math>N=4</math>.</p> <p>0011: <math>f_{SAMP} = f_{TIMER\_CK}</math>, <math>N=8</math>.</p> <p>0100: <math>f_{SAMP} = f_{DTS}/2</math>, <math>N=6</math>.</p> <p>0101: <math>f_{SAMP} = f_{DTS}/2</math>, <math>N=8</math>.</p> <p>0110: <math>f_{SAMP} = f_{DTS}/4</math>, <math>N=6</math>.</p> <p>0111: <math>f_{SAMP} = f_{DTS}/4</math>, <math>N=8</math>.</p> <p>1000: <math>f_{SAMP} = f_{DTS}/8</math>, <math>N=6</math>.</p> <p>1001: <math>f_{SAMP} = f_{DTS}/8</math>, <math>N=8</math>.</p> <p>1010: <math>f_{SAMP} = f_{DTS}/16</math>, <math>N=5</math>.</p> <p>1011: <math>f_{SAMP} = f_{DTS}/16</math>, <math>N=6</math>.</p> <p>1100: <math>f_{SAMP} = f_{DTS}/16</math>, <math>N=8</math>.</p> <p>1101: <math>f_{SAMP} = f_{DTS}/32</math>, <math>N=5</math>.</p> <p>1110: <math>f_{SAMP} = f_{DTS}/32</math>, <math>N=6</math>.</p>

1111:  $f_{SAMP}=f_{DTS}/32$ , N=8.

7	MSM	Master-slave mode 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. 0: Master-slave mode disable 1: Master-slave mode enable
6:4	TRGS[2:0]	Trigger selection This bit-field specifies which signal is selected as the trigger input, which is used to synchronize the counter. 000: Internal trigger input 0 (ITI0) 001: Internal trigger input 1 (ITI1) 010: Internal trigger input 2 (ITI2) 011: Internal trigger input 3 (ITI3) 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) These bits must not be changed when slave mode is enabled.
3	Reserved	Must be kept at reset value.
2:0	SMC[2:0]	Slave mode control 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. 111: External clock mode 0. The counter counts on the rising edges of the selected trigger.

### DMA and interrupt enable register (TIMERx\_DMINTEN)

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	CHOIE	UPIE

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	CHOIF	UPIF	

rc\_w0      rc\_w0      rc\_w0      rc\_w0           rc\_w0      rc\_w0      rc\_w0      rc\_w0      rc\_w0      rc\_w0      rc\_w0      rc\_w0      rc\_w0      rc\_w0

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 CHOIF 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	<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> <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)

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      w

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> <p>1: Generate a break event</p>
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</p> <p>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</p> <p>1: Generate channel's c/c control update event</p>
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 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</p>

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

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 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: Timing mode. 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 “Timing mode” 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 CH0MS 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 TIMERx_CH0CV 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 (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</p>

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.

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.

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.

1:0	CH0MS[1:0]	Channel 0 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. (CH0EN bit in TIMERx_CHCTL2 register is reset). 00: Channel 0 is configured as output 01: Channel 0 is configured as input, IS0 is connected to CI0FE0 10: Channel 0 is configured as input, IS0 is connected to CI1FE0 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.
-----	------------	--

#### **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 CI0 input signal and the length of the digital filter applied to CI0.  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 4channel 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 CH0COMFEN 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 CI3FE3 10: Channel 3 is configured as input, IS3 is connected to CI2FE3 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	<p>Channel 2 output compare clear enable. 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 1: Channel 2 output compare clear enable</p>
6:4	CH2COMCTL[2:0]	<p>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> <p>000: Timing mode. 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 the channel output. O2CPRE signal is forced high when the counter matches the output compare register TIMERx_CH2CV.</p> <p>010: Clear the channel output. 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, O2CPRE is active as long as the counter is smaller than TIMERx_CH2CV else inactive. When counting down, O2CPRE is inactive as long as the counter is larger than TIMERx_CH2CV else active.</p> <p>111: PWM mode1. When counting up, O2CPRE is inactive as long as the counter is smaller than TIMERx_CH2CV else active. When counting down, O2CPRE is active as long as the counter is larger than TIMERx_CH2CV else inactive. When configured in PWM mode, the O2CPRE level changes only when the output compare mode switches from “Timing mode” 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 CH2MS bit-filed is 00(COMPARE MODE).</p>
3	CH2COMSEN	<p>Channel 2 compare output shadow enable 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>

The PWM mode can be used without validating the shadow register only in single 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	CH2COMFEN	Channel 2 output compare fast enable  When this bit is set, the effect of an event on the trigger 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.  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.
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
Reserved	CH3P	CH3EN	CH2NP	CH2NEN	CH2P	CH2EN	CH1NP	CH1NEN	CH1P	CH1EN	CH0NP	CH0NEN	CH0P	CH0EN	

rw      rw

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 This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 or 10 and CH0MS [1:0] bit-filed in TIMERx_CHCTL0 register is 00.
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. 0: CI0 is non-inverted. Input capture is done on a rising edge of CI0. When used as extern trigger, CI0 is non-inverted. 1: CI0 is inverted. Input capture is done on a falling edge of CI0. When used as extern trigger, CI0 is 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)**

Address offset: 0x24

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
CNT[15:0]															

rw

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
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)

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

rw

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
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)

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

Bits	Fields	Descriptions
15:0	CARL[15:0]	<p>Counter auto reload value</p> <p>This bit-filed specifies the auto reload value of the counter.</p> <p>Note: When the timer is configured in input capture mode, this register must be configured a non-zero value (such as 0xFFFF) which is larger than user expected value.</p>

### 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)

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

Bits	Fields	Descriptions
15:8	Reserved	Must be kept at reset value.
7:0	CREP[7:0]	<p>Counter repetition value</p> <p>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.</p>

### 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)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH0VAL[15:0]															
rw															

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>

When channel 0 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)

Address offset: 0x38

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
CH1VAL[15:0]															

rw

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-field indicates the counter value corresponding to the last capture event. And this bit-field is read-only.</p> <p>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.</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)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH2VAL[15:0]															

rw

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-field indicates the counter value corresponding to the last capture event. And this bit-field is read-only.</p> <p>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.</p>

### 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)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH3VAL[15:0]															
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>

### Complementary channel protection register (TIMERx\_CCHP)

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
POEN	OAEN	BRKP	BRKEN	ROS	IOS	PROT[1:0]	DTCFG[7:0]								
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.</p> <p>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.</p> <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	Break polarity

		This bit specifies the polarity of the BRKIN input signal. 0: BRKIN input active low 1; BRKIN input active high
12	BRKEN	<p>Break enable</p> <p>This bit can be set to enable the BRKIN and CKM 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, 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-field in TIMERx_CCHP register is 10 or 11.</p>
9:8	PROT[1:0]	<p>Complementary register protect control</p> <p>This bit-field 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> <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.</p>
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'b10x: DTvalue = (64+DTCFG [5:0]) x t<sub>DT</sub>, t<sub>DT</sub> = t<sub>DTs</sub> x 2.

DTCFG [7:5] =3'b110: DTvalue = (32+DTCFG [4:0]) x t<sub>DT</sub>, t<sub>DT</sub> = t<sub>DTs</sub> x 8.

DTCFG [7:5] =3'b111: DTvalue = (32+DTCFG [4:0]) x t<sub>DT</sub>, t<sub>DT</sub> = t<sub>DTs</sub> x 16.

This bit can be modified only when PROT [1:0] bit-filled 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)

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

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
15:13	Reserved	Must be kept at reset value.
12:8	DMATC [4:0]	<p>DMA transfer count</p> <p>This filed is defined the number of DMA will access(R/W) the register of TIMERx_DMATB</p> <p>5'b0_0000: 1 time transfer</p> <p>5'b0_0001: 2 times transfer</p> <p>...</p> <p>5'b1_0001: 18 times transfer</p>
7:5	Reserved	Must be kept at reset value.
4:0	DMATA [4:0]	<p>DMA transfer access start address</p> <p>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.</p> <p>5'b0_0000: TIMERx_CTL0</p> <p>5'b0_0001: TIMERx_CTL1</p> <p>...</p> <p>5'b1_0010: TIMERx_DMACFG</p> <p>In a word: Start Address = TIMERx_CTL0 + DMATA *4</p>

**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)

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

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>

## 15.2. General level0 timer (TIMERx, x=1, 2, 3, 4)

### 15.2.1. Overview

The general level0 timer module (TIMER1, 2, 3, 4) is a four-channel timer that supports input capture and 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 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.

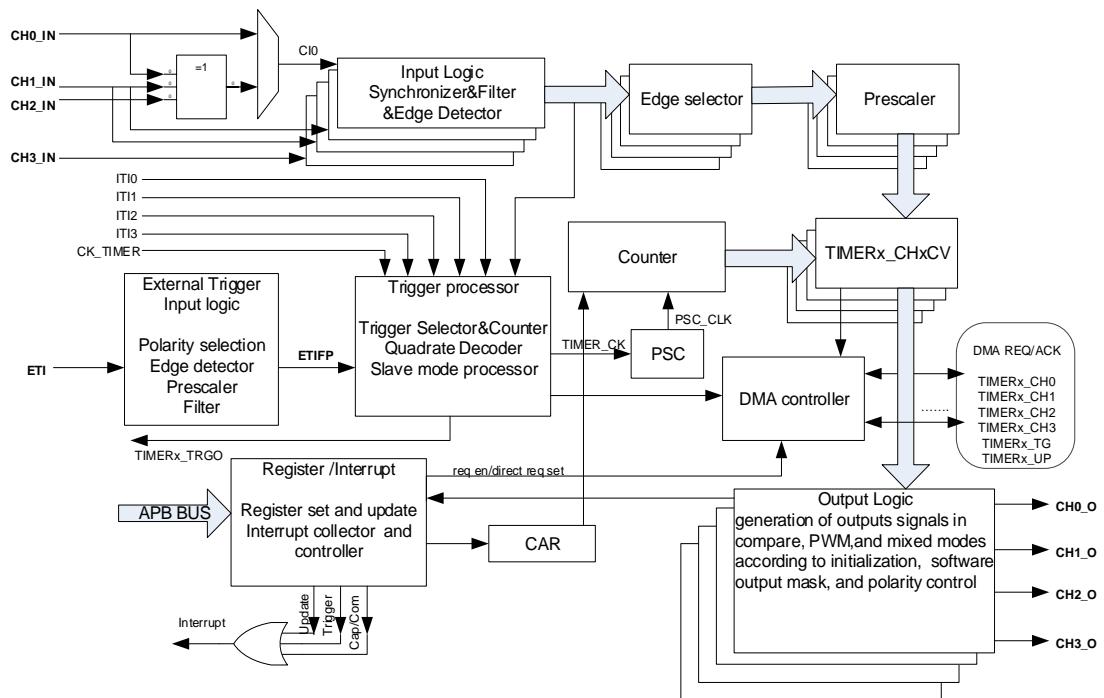
Timers are completely independent with each other, but there may be synchronized to provide a larger timer with their counters value incrementing in unison.

### 15.2.2. Characteristics

- Total channel num: 4.
- Counter width: 16-bit.
- Clock source of timer is selectable: internal clock, internal trigger, external input, external trigger.
- Multiple counter modes: up counting, down counting and center-aligned counting.
- Quadrature decoder: used for motion tracking and determination of both rotation direction and position.
- Hall sensor function: used for 3-phase motor control.
- Programmable prescaler: 16-bit. The factor can be changed ongoing.
- Each channel is user-configurable: input capture mode, output compare mode, programmable PWM mode and single pulse mode
- Auto-reload function.
- Interrupt output or DMA request: update event, 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 15-34. General Level 0 timer block diagram](#) provides details on the internal configuration of the general level0 timer.

**Figure 15-34. General Level 0 timer block diagram**


## 15.2.4. Function overview

### Clock selection

The clock source of the general level0 TIMER can be either the CK\_TIMER or an alternate clock source controlled by SMC bits (TIMERx\_SMCFG bit[2:0]).

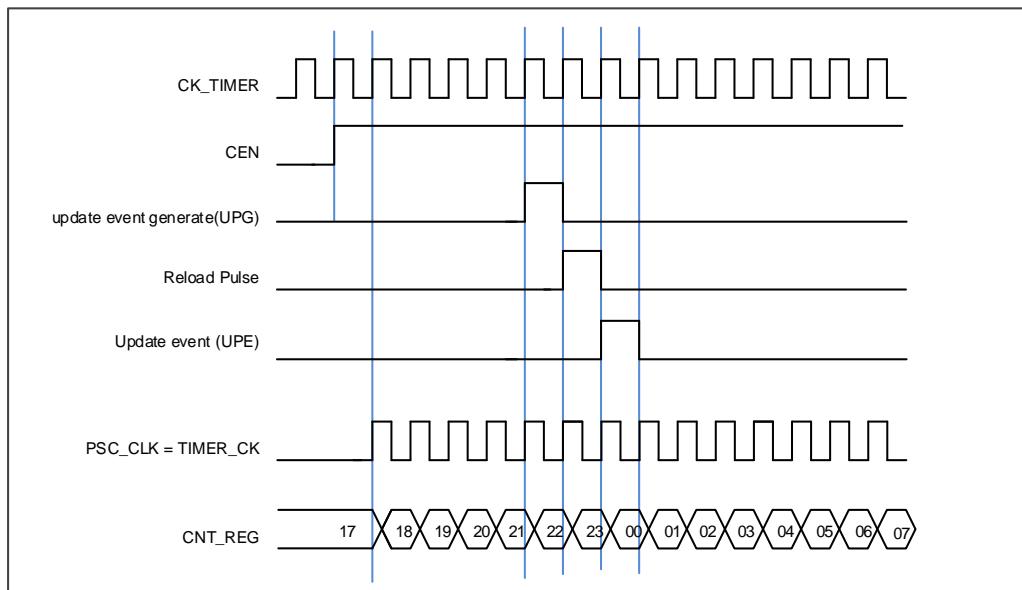
- SMC [2:0] == 3'b000. Internal timer clock CK\_TIMER which is from module RCU.

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 module.

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, more details will be introduced later. When the slave mode control bits SMC [2:0] are set to 0x4, 0x5 or 0x6, the internal clock TIMER\_CK is the counter prescaler driving clock source.

Figure 15-35. 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.

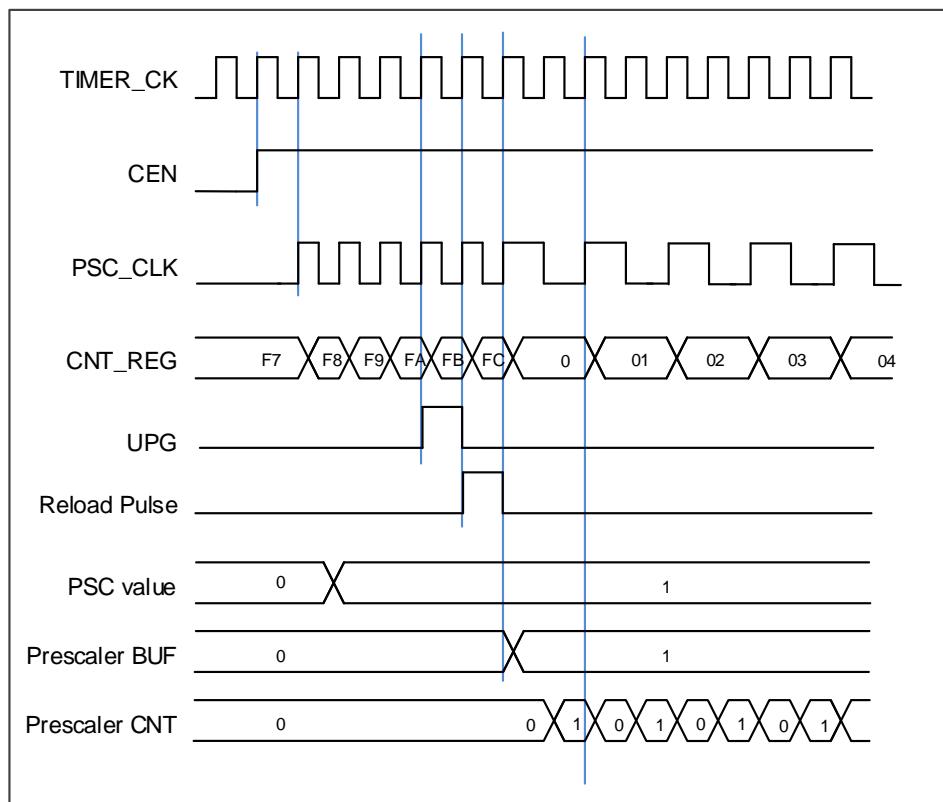
- SMC1== 1'b1 (external clock mode 1). External input ETI is selected as timer clock source

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 ETI. This mode can be selected by setting the SMC1 bit in the TIMERx\_SMCFG register to 1. The other way to select the ETI signal as the clock source is setting the SMC [2:0] to 0x7 and the TRGS [2:0] to 0x7. Note that the ETI signal is derived from the ETI pin sampled by a digital filter. When the ETI signal is selected as the 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 ranging from 1 to 65536. It is controlled by prescaler register (TIMERx\_PSC) which can be changed ongoing, but it is adopted at the next update event.

Figure 15-36. 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 event will be generated after  $(\text{TIMERx_CREP}+1)$  times of overflow. Otherwise the update event is generated each time when counter 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 an update event will be generated.

If the `UPDIS` bit in `TIMERx_CTL0` register is set, the update event is disabled.

When an update event occurs, all the registers (auto reload register, prescaler register) are updated.

[Figure 15-37. Timing chart of up counting mode, PSC=0/1](#) and [Figure 15-38. Timing chart of up counting mode, change TIMERx CAR ongoing](#) show some examples of the counter behavior for different clock prescaler factors when `TIMERx_CAR=0x63`.

Figure 15-37. Timing chart of up counting mode, PSC=0/1

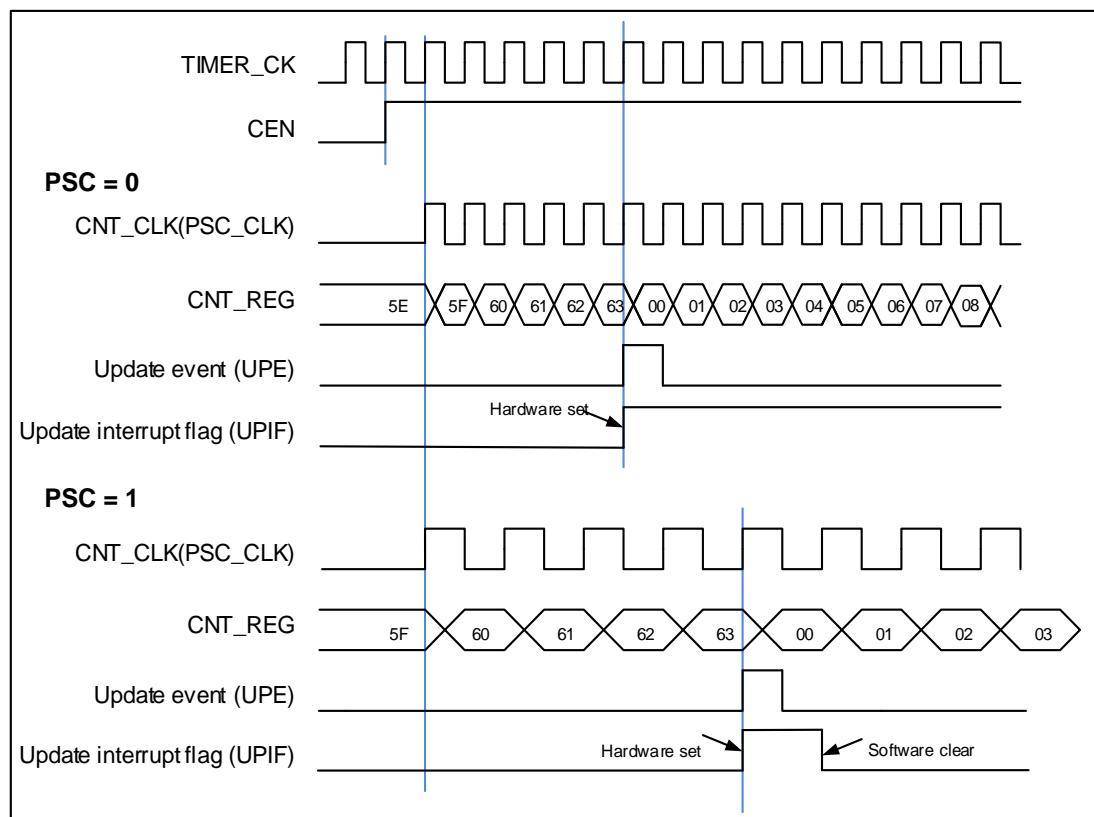
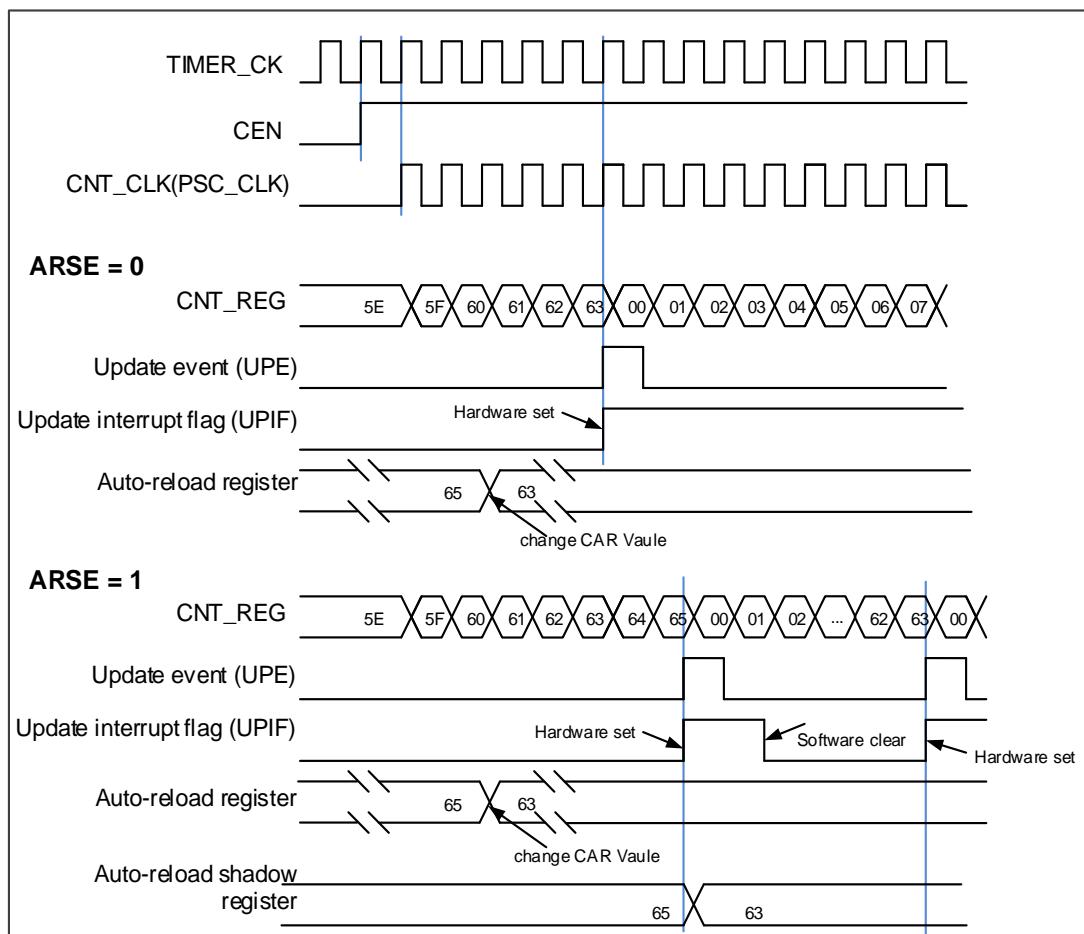


Figure 15-38. Timing chart of up counting mode, change TIMERx\_CAR ongoing



### Down counting mode

In this mode, the counter counts down continuously from the counter reload value, which is defined in the TIMERx\_CAR register, in a count-down direction. Once the counter reaches 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 counter 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 an update event will be generated.

If the UPDIS bit in TIMERx\_CTL0 register is set, the update event is disabled.

When an update event occurs, all the registers (auto reload register, prescaler register) are updated.

[Figure 15-39. Timing chart of down counting mode, PSC=0/1](#) and [Figure 15-40. Timing chart of down counting mode, change TIMERx\\_CAR](#).show some examples of the counter behavior in different clock frequencies when TIMERx\_CAR = 0x63.

Figure 15-39. Timing chart of down counting mode, PSC=0/1

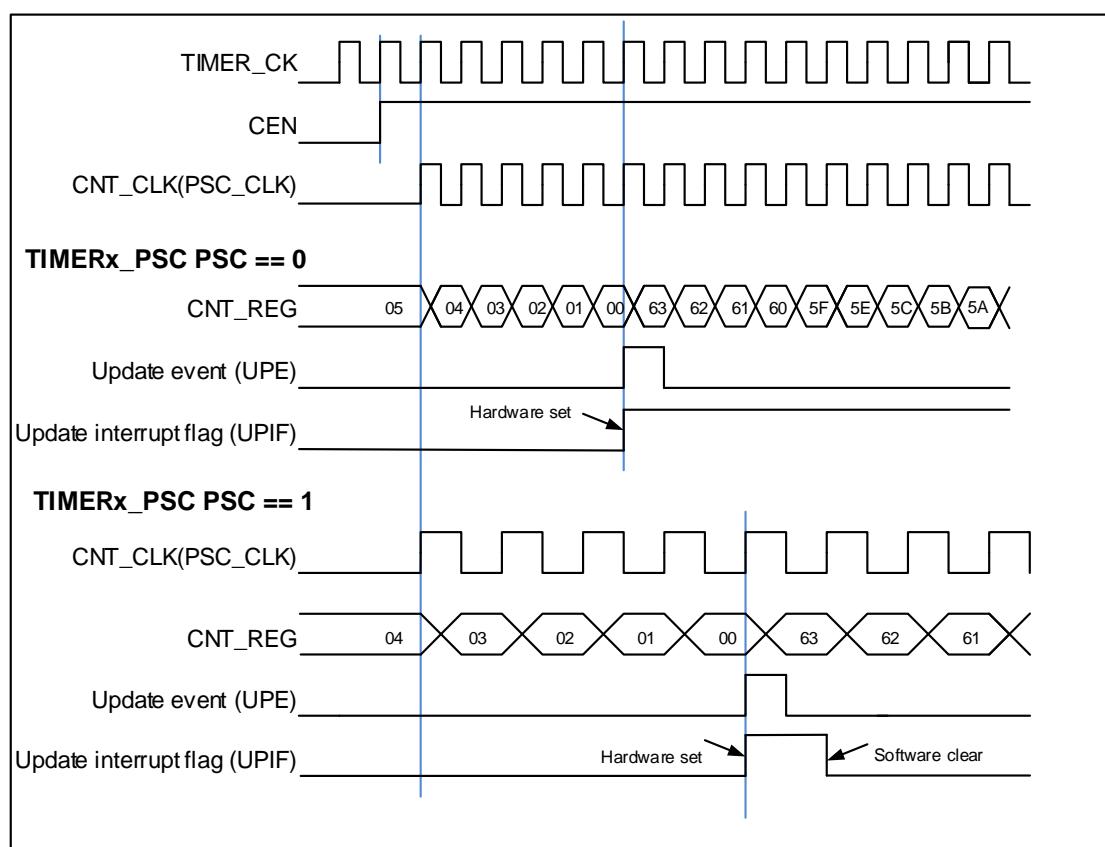
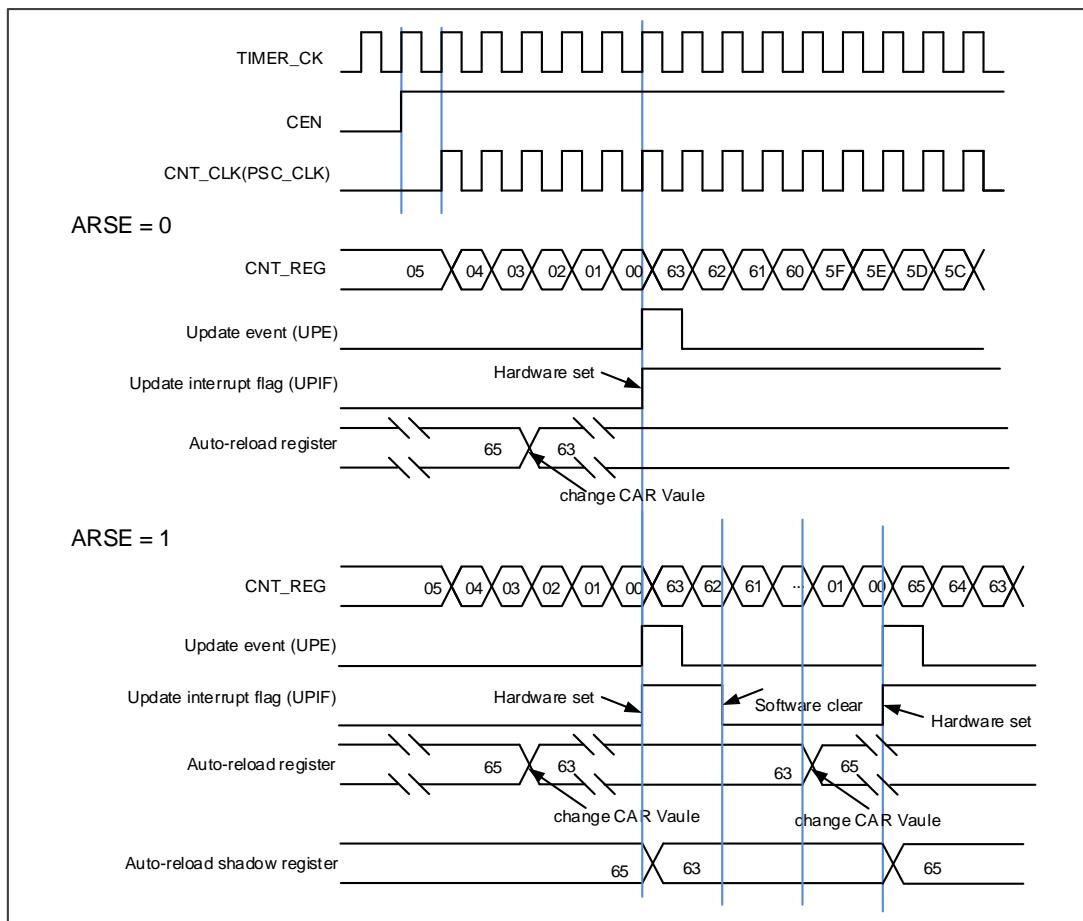


Figure 15-40. Timing chart of down counting mode, change TIMERx\_CAR.



### 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 (TIMERx\_CREP-1) in the count-up direction and generates an underflow event when the counter counts to 1 in the count-down direction. The counting direction bit DIR in the TIMERx\_CTL0 register is read-only and indicates the counting direction when in the center-aligned counting 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 generate an update event irrespective of whether the counter is counting up or down in the center-aligned counting mode.

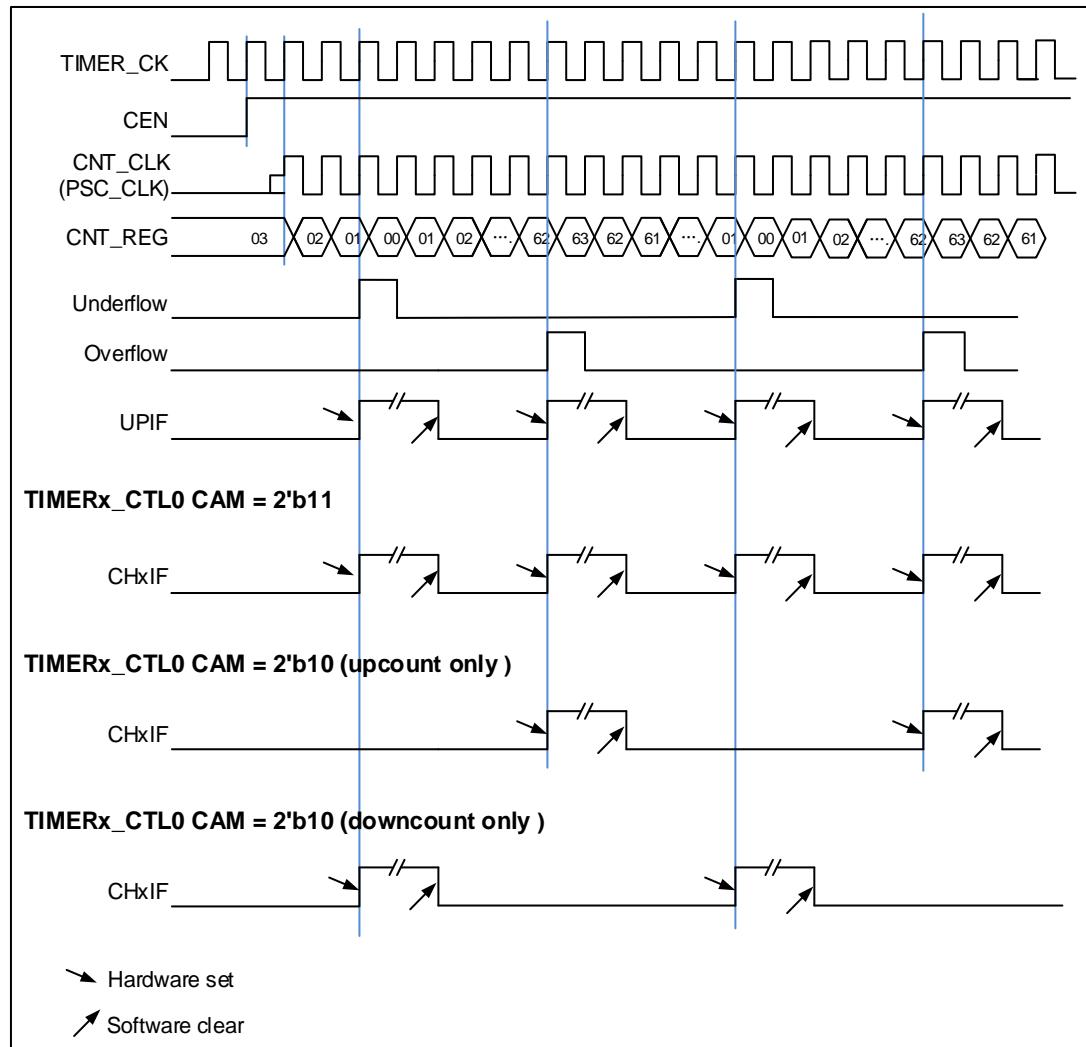
The UPIF bit in the TIMERx\_INTF register will be set to 1 either when an underflow event or an overflow event occurs. While the CHxIF bit is associated with the value of CAM in TIMERx\_CTL0. The details refer to [Figure 15-41. Timing chart of center-aligned counting mode](#).

If the UPDIS bit in the TIMERx\_CTL0 register is set, the update event is disabled.

When an update event occurs, all the registers (auto-reload register, prescaler register) are updated.

**Figure 15-41. Timing chart of center-aligned counting mode** show some examples of the counter behavior when TIMERx\_CAR=0x63. TIMERx\_PSC=0x0

**Figure 15-41. Timing chart of center-aligned counting mode**



### Capture/compare channels

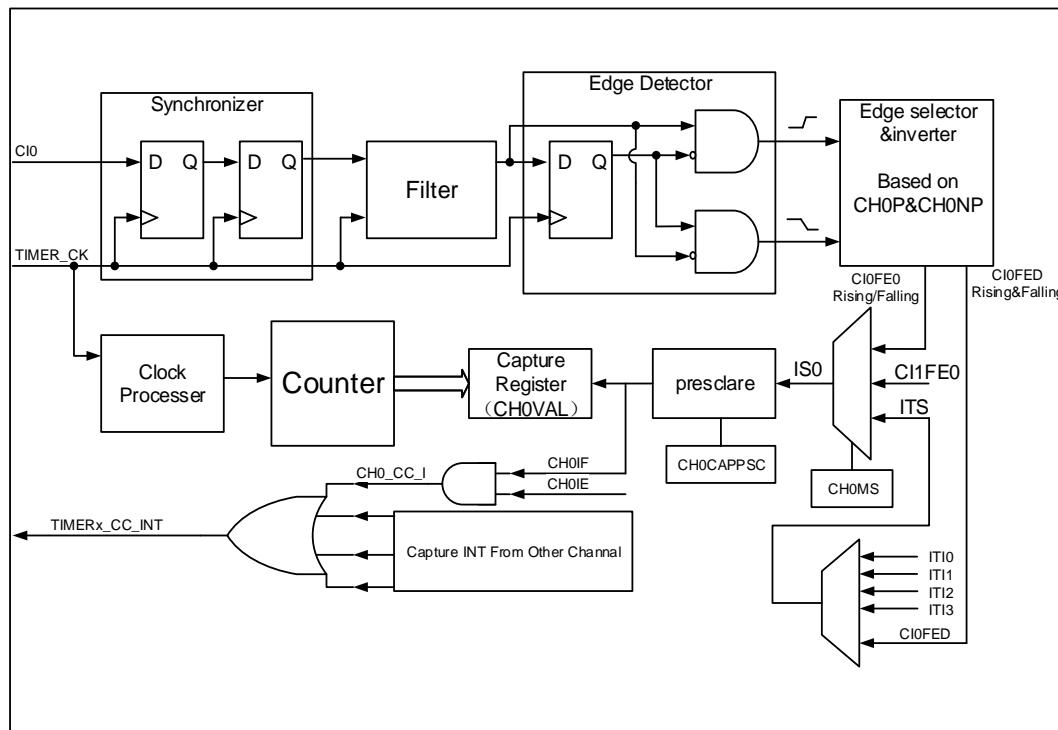
The general level0 Timer has four independent channels which can be used as capture inputs or compare outputs. Each channel is built around a channel capture compare register including an input stage, a channel controller and an output stage.

- Input capture mode

Input 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 it is enabled when CHxIE=1.

**Figure 15-42. Input capture logic**



The input signals of channelx (CIx) can be the TIMERx\_CHx signal or the XOR signal of the TIMERx\_CH0, TIMERx\_CH1 and TIMERx\_CH2 signals. First, the input signal of channel (CIx) is synchronized to TIMER\_CK signal, and then sampled by a digital filter to generate a filtered input signal. Then through the edge detector, the rising or falling edge is detected by configuring CHxP bit. The input capture signal can also be selected from the input signal of other channel or the internal trigger signal by configuring CHxMS bits. The IC prescaler makes several input events generate one effective capture event. On the capture event, TIMERx\_CHxCV will store 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 quality of requested signal, configure compatible CHxCAPFLT.
- Step2:** Edge selection (CHxP/CHxNP in TIMERx\_CHCTL2).  
Rising edge or falling edge, choose one by configuring CHxP/CHxNP bits.
- Step3:** Capture source selection (CHxMS in TIMERx\_CHCTL0).  
As soon as selecting one input capture source by CHxMS, the channel must be set 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 to get the interrupt and DMA request.
- Step5:** Capture enable (CHxEN in TIMERx\_CHCTL2).

**Result:** When the wanted input signal is captured, TIMERx\_CHxCV will be set by counter's value and CHxIF is asserted. If the CHxIF is 1, the CHxOF will also be asserted. The interrupt and DMA request will be asserted or not based on the configuration of CHxIE and CHxDEN in TIMERx\_DMAINTEN.

**Direct generation:** A DMA request or interrupt is generated by setting CHxG directly.

The input capture mode can be also used for pulse width measurement from signals on the TIMERx\_CHx pins. For example, PWM signal connects to CI0 input. Select CI0 as channel 0 capture signals by setting CH0MS to 2'b01 in the channel control register (TIMERx\_CHCTL0) and set capture on rising edge. Select CI0 as channel 1 capture signal by setting CH1MS to 2'b10 in the channel control register (TIMERx\_CHCTL0) and set capture on falling edge. The counter is set to restart mode and is restarted on channel 0 rising edge. Then the TIMERX\_CH0CV can measure the PWM period and the TIMERx\_CH1CV can measure the PWM duty cycle.

■ 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 TIMERx\_CHxCV 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 TIMERx\_CHxCV register, the CHxIF bit will be set and the channel (n) interrupt is generated if CHxIE = 1. And the DMA request will be asserted, if CxDCE=1.

So the process can be divided into 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
- Enable the output by CHxEN

**Step3:** Interrupt/DMA-request enables configuration by CHxIE/CxDEN

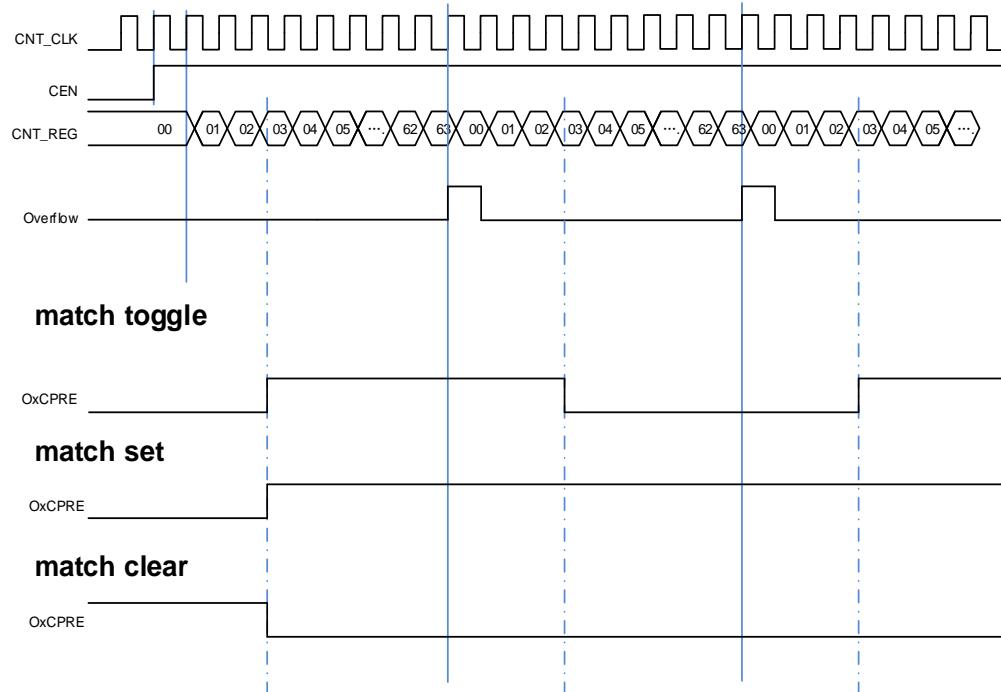
**Step4:** Compare output timing configuration by TIMERx\_CAR and TIMERx\_CHxCV.

About the CHxVAL, you can change it on the go to meet the waveform you expected.

**Step5:** Start the counter by configuring CEN to 1.

[\*\*Figure 15-43. Output-compare in three modes\*\*](#) show the three compare modes toggle/set/clear. CAR=0x63, CHxVAL=0x3

Figure 15-43. Output-compare in three modes



## PWM mode

In the PWM output mode (by setting the CHxCOMCTL bit to 3'b110 (PWM mode 0) or to 3'b111(PWM mode 1)), the channel can generate PWM waveform according to the TIMERx\_CAR registers and TIMERx\_CHxCV registers.

Based on the counter mode, PWM can also be divided into EAPWM (Edge-aligned PWM) and CAPWM (Center-aligned PWM).

The EAPWM's period is determined by TIMERx\_CAR and the duty cycle is determined by TIMERx\_CHxCV. [Figure 15-44. 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 15-45. 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 15-44. EAPWM timechart

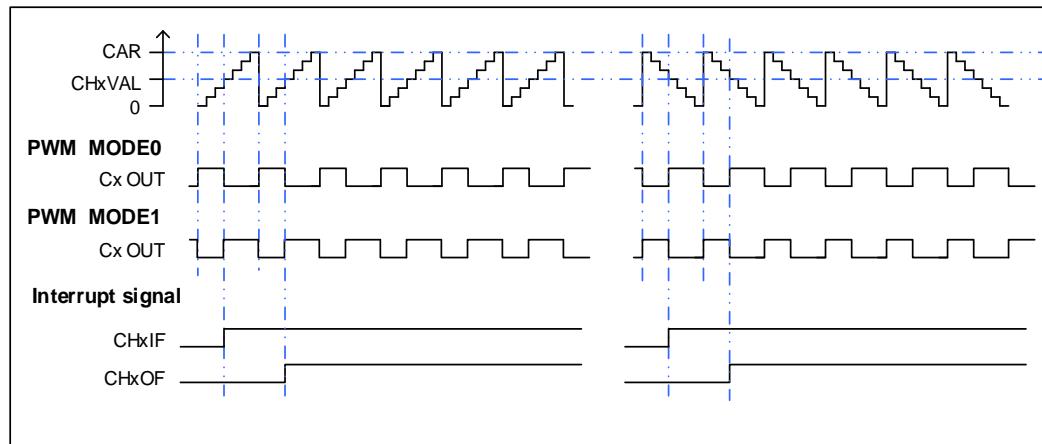
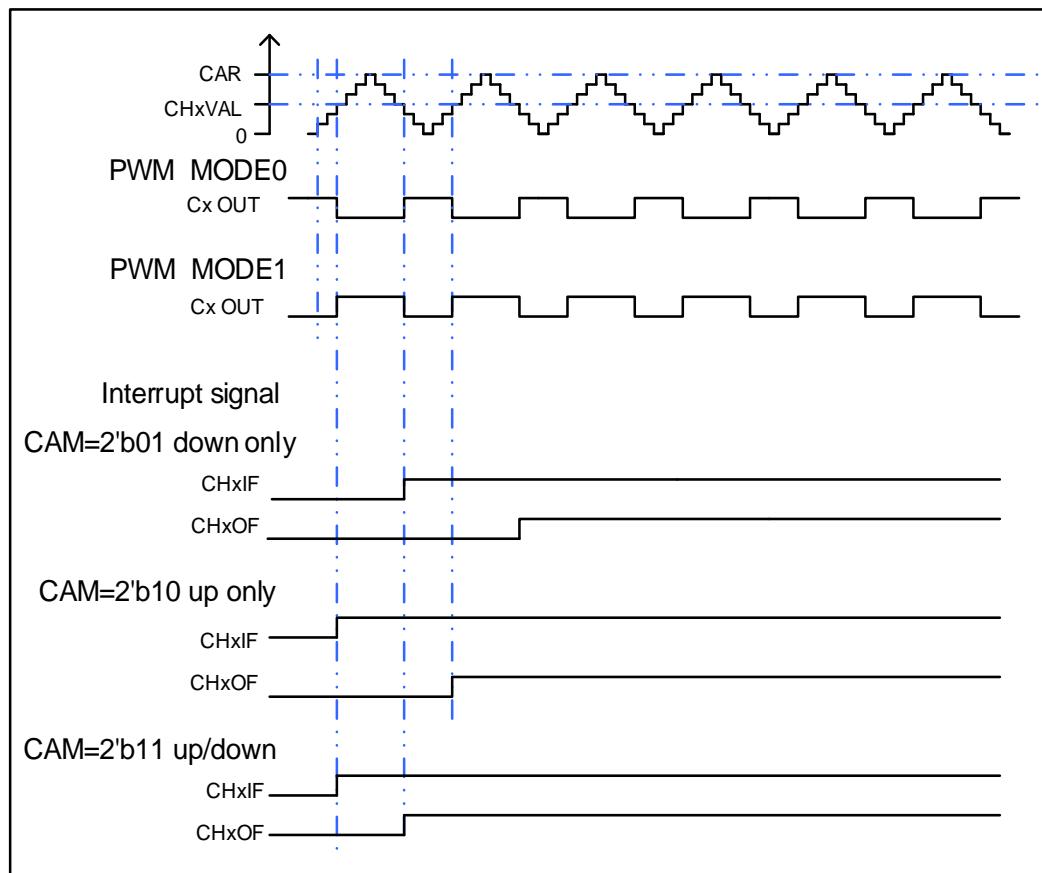


Figure 15-45. CAPWM timechart



### Channel output prepare 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 ETIFP 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 with each other to generate the counter value. Setting SMC=0x01, 0x02, or 0x03 to select that the counting direction of timer is determined only by the CI0, only by the CI1, or by the CI0 and the CI1. The DIR bit is modified by hardware automatically during the voltage level change of each direction selection source. The mechanism of changing the counter direction is shown in [Table 15-5](#).

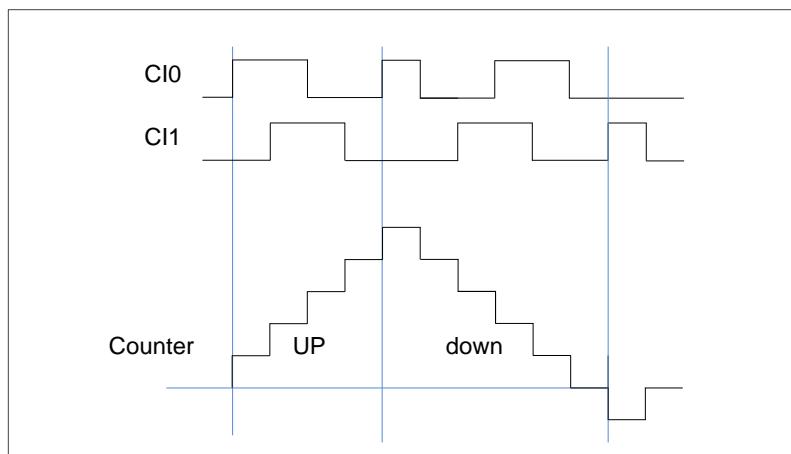
**[Counting direction versus encoder signals](#)**. The quadrature decoder can be regarded as an external clock with a direction selection. This means that the counter counts continuously from 0 to the counter-reload value. Therefore, users must configure the TIMERx\_CAR register before the counter starts to count.

**Table 15-5. Counting direction versus encoder signals**

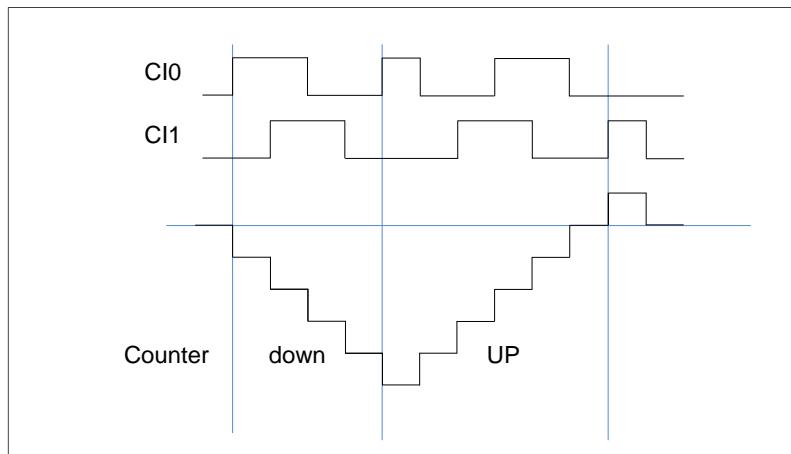
Counting mode	Level	CI0FE0		CI1FE1	
		Rising	Falling	Rising	Falling
CI0 only counting	CI1FE1=High	Down	Up	-	-
	CI1FE1=Low	Up	Down	-	-
CI1 only counting	CI0FE0=High	-	-	Up	Down
	CI0FE0=Low	-	-	Down	Up
CI0 and CI1 counting	CI1FE1=High	Down	Up	X	X
	CI1FE1=Low	Up	Down	X	X
	CI0FE0=High	X	X	Up	Down
	CI0FE0=Low	X	X	Down	Up

**Note:** "-" means "no counting"; "X" means impossible.

**Figure 15-46. Example of counter operation in encoder interface mode**



**Figure 15-47. Example of encoder interface mode with CI0FE0 polarity inverted**



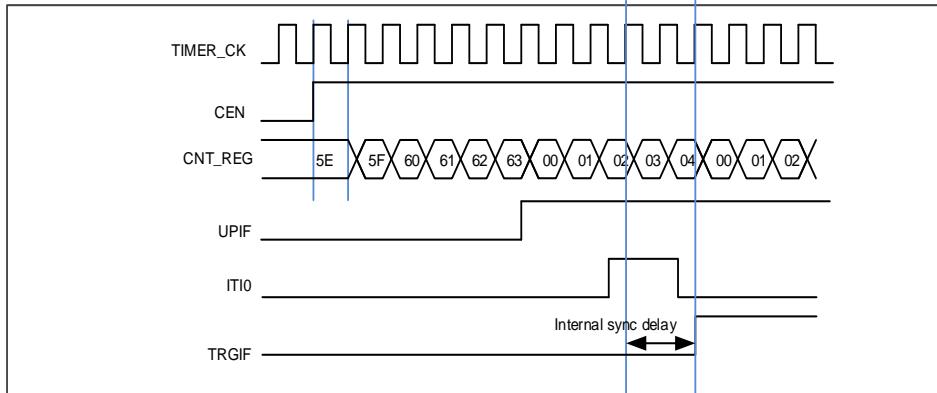
### Hall sensor function

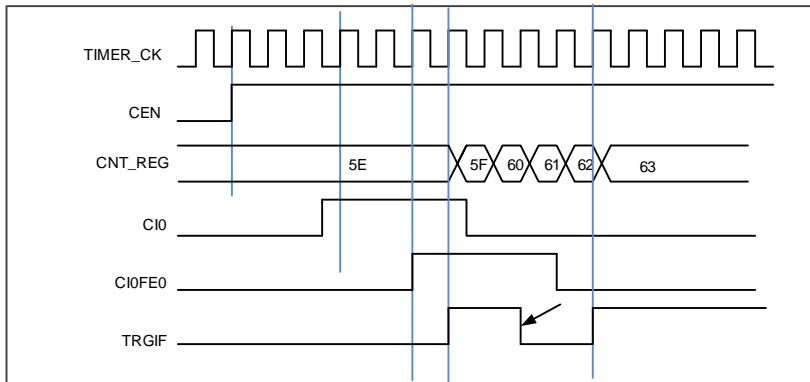
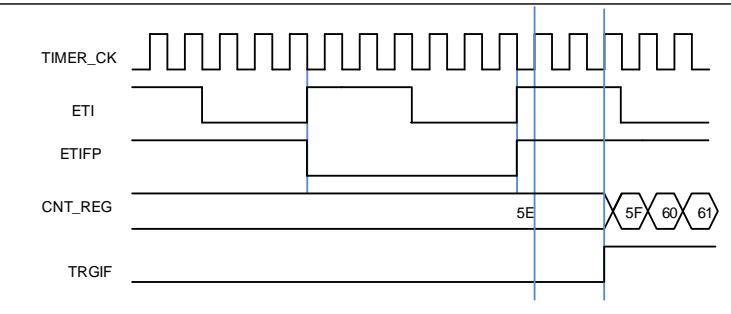
Refer to Advanced timer (TIMERx, x=0).

### Slave controller

The TIMERx can be synchronized with a trigger in several modes including restart mode, pause mode and event mode which is selected by the SMC[2:0] bits in the TIMERx\_SMCFG register. The input trigger of these modes can be selected by the TRGS[2:0] bits in the TIMERx\_SMCFG register.

Table 15-6. Slave controller examples

	Mode Selection	Source Selection	Polarity Selection	Filter and Prescaler
<b>LIST</b>	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: ETIFF	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 Clx, configure Filter by CHxCAPFLT, no prescaler can be used. For the ETIF, configure Filter by ETFC and Prescaler by ETPSC.
<b>Exam1</b>	<b>Restart mode</b> 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 15-48. Restart mode</b>			
				
<b>Exam2</b>	<b>Pause mode</b> 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
	<b>Figure 15-49. Pause mode</b>			
		 <p>The diagram illustrates the timing for Pause mode. The <b>TIMER_CK</b> signal is a square wave. The <b>CEN</b> signal is high during the first four cycles of <b>TIMER_CK</b>. The <b>CNT_REG</b> signal shows a count from 5E to 63. The <b>CI0</b> signal has a pulse at the start of the fourth cycle. The <b>CI0FE0</b> signal has a pulse at the start of the fifth cycle. The <b>TRGIF</b> signal is asserted at the start of the fifth cycle.</p>		
<b>Exam3</b>	<b>Event mode</b> 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
	<b>Figure 15-50. Event mode</b>			
		 <p>The diagram illustrates the timing for Event mode. The <b>TIMER_CK</b> signal is a square wave. The <b>ETI</b> signal has three rising edges. The <b>ETIFP</b> signal has three pulses corresponding to the rising edges of <b>ETI</b>. The <b>CNT_REG</b> signal shows a count from 5E to 61. The <b>TRGIF</b> signal is asserted at the third rising edge of <b>ETI</b>.</p>		

### Single pulse mode

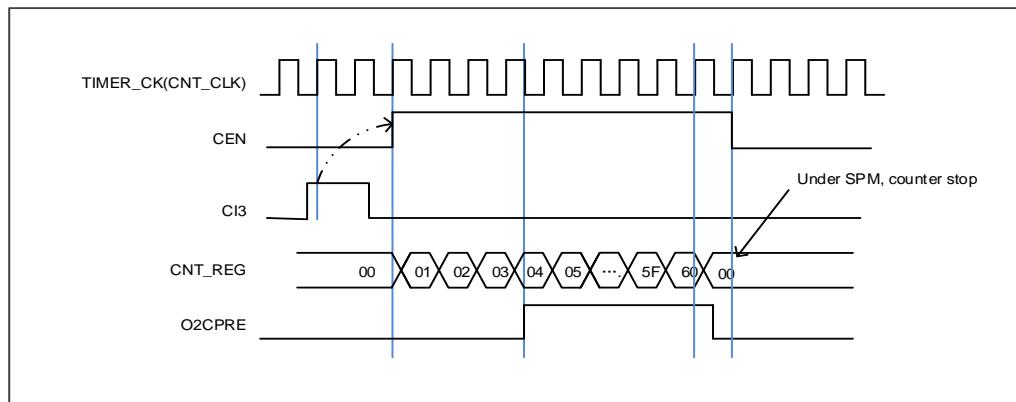
Single pulse mode is enabled by setting SPM in **TIMERx\_CTL0**. If SPM is set, the counter will be cleared and stopped automatically when the next update event occurs. In order to get a pulse waveform, the **TIMERx** is configured to PWM mode or compare mode by **CHxCOMCTL**.

Once the timer is set to the single pulse mode, it is not necessary to configure the timer enable bit **CEN** in the **TIMERx\_CTL0** register to 1 to enable the counter. Setting the **CEN** bit to 1 or a trigger signal 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 by software, the counter will be stopped and its value will be 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 active edge of trigger which sets the **CEN** bit to 1 will enable the counter. However, there exists 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 **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 the PWM mode 0 or PWM mode 1 and the trigger source is derived from the trigger signal.

**Figure 15-51. Single pulse mode TIMERx\_CHxCV = 0x04 TIMERx\_CAR=0x60**



## Timers interconnection

Refer to [Advanced timer \(TIMERx, x=0\)](#).

## Timer DMA mode

Timer DMA mode is the function that configures timer's register by DMA module. The relative registers are TIMERx\_DMACFG and TIMERx\_DMATB. Corresponding DMA request bit should be asserted to enable DMA request for internal interrupt event. TIMERx will send a request to DMA when the interrupt event occurs. DMA is configured to M2P (memory to peripheral) mode and the address of TIMERx\_DMATB is configured to PADDR (peripheral base address), then DMA will access the TIMERx\_DMATB. In fact, TIMERx\_DMATB register 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), the timer sends only one DMA request. 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 and DMATA+0xC at the next 3 accesses to TIMERx\_DMATB. In a word, one-time DMA internal interrupt event asserts, (DMATC+1) times request will be sent by TIMERx.

If one more DMA request event occurs, TIMERx will repeat the process above.

## Timer debug mode

When the RISC-V core halted, and the TIMERx\_HOLD configuration bit in DBG\_CTL register set to 1, the TIMERx counter stops.

### 15.2.5. TIMERx registers(x=1,2,3,4)

TIMER1 base address: 0x40000000

TIMER2 base address: 0x40000400

TIMER3 base address: 0x40000800

TIMER4 base address: 0x40000C00

#### 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                        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: $f_{DTS}=f_{\text{TIMER\_CK}}$ 01: $f_{DTS}= f_{\text{TIMER\_CK}}/2$ 10: $f_{DTS}= f_{\text{TIMER\_CK}}/4$ 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: 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)

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

Bits	Fields	Descriptions
15:8	Reserved	Must be kept at reset value
7	TIOS	<p>Channel 0 trigger input selection</p> <p>0: The TIMERx_CH0 pin input is selected as channel 0 trigger input.</p> <p>1: The result of combinational XOR of TIMERx_CH0, CH1 and CH2 pins is selected as channel 0 trigger input.</p>
6:4	MMC[2:0]	<p>Master mode control</p> <p>These bits control the selection of TRGO signal, which is sent in master mode to slave timers for synchronization function.</p> <p>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.</p> <p>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.</p> <p>010: Update. In this mode the master mode controller selects the update event as TRGO.</p> <p>011: Capture/compare pulse. In this mode the master mode controller generates a TRGO pulse when a capture or a compare match occurred.</p> <p>100: Compare. In this mode the master mode controller selects the O0CPRE signal is used as TRGO</p> <p>101: Compare. In this mode the master mode controller selects the O1CPRE signal is used as TRGO</p> <p>110: Compare. In this mode the master mode controller selects the O2CPRE signal is used as TRGO</p> <p>111: Compare. In this mode the master mode controller selects the O3CPRE signal is used as TRGO</p>
3	DMAS	<p>DMA request source selection</p> <p>0: DMA request of channel x is sent when channel x event occurs.</p> <p>1: DMA request of channel x is sent when update event occurs.</p>
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]	Reserved	SMC[2:0]							

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 ETIFP signal.</p> <p>0: External clock mode 1 disabled</p> <p>1: External clock mode 1 enabled.</p> <p>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.</p> <p>The external clock input will be ETIFP if external clock mode 0 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-field.</p>
13:12	ETPSC[1:0]	<p>External trigger prescaler</p> <p>The frequency of external trigger signal ETIFP 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 ETIFP frequency.</p> <p>00: Prescaler disable</p> <p>01: ETIFP frequency will be divided by 2</p> <p>10: ETIFP frequency will be divided by 4</p> <p>11: ETIFP 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 ETIFP signal and the length of the digital filter applied to ETIFP.</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.

7	MSM	Master-slave mode  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.  0: Master-slave mode disable 1: Master-slave mode enable
6:4	TRGS[2:0]	Trigger selection  This bit-field specifies which signal is selected as the trigger input, which is used to synchronize the counter.  000: Internal trigger input 0 (ITI0) 001: Internal trigger input 1 (ITI1) 010: Internal trigger input 2 (ITI2) 011: Internal trigger input 3 (ITI3) 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)  These bits must not be changed when slave mode is enabled.
3	Reserved	Must be kept at reset value.
2:0	SMC[2:0]	Slave mode control  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.

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	CHOIE	UPIE

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      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	
								w	w	w	w	w	w	w	w

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</p> <p>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</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 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</p> <p>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
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

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. O0CPRE is active high, while CH0_O active level depends on CH0P bits. 000: Timing mode. 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 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 “Timing mode” 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 pulse mode (SPM bit in TIMERx_CTL0 register is set).</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.</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 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.</p> <p>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</p>

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 CI0 input signal and the length of the digital filter applied to CI0. 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. O2CPRE is active high, while CH2_O active level depends on CH2P bits. 000: Timing mode. The O2CPRE signal keeps stable, independent of the

comparison between the output compare register TIMERx\_CH2CV and the counter TIMERx\_CNT.

001: Set the channel output. O2CPRE signal is forced high when the counter matches the output compare register TIMERx\_CH2CV.

010: Clear the channel output. O2CPRE signal is forced low when the counter matches the output compare register TIMERx\_CH2CV.

011: Toggle on match. O2CPRE toggles when the counter matches the output compare register TIMERx\_CH2CV.

100: Force low. O2CPRE is forced low level.

101: Force high. O2CPRE is forced high level.

110: PWM mode 0. When counting up, O2CPRE is active as long as the counter is smaller than TIMERx\_CH2CV else inactive. When counting down, O2CPRE is inactive as long as the counter is larger than TIMERx\_CH2CV else active.

111: PWM mode 1. When counting up, O2CPRE is inactive as long as the counter is smaller than TIMERx\_CH2CV else active. When counting down, O2CPRE is active as long as the counter is larger than TIMERx\_CH2CV else inactive.

When configured in PWM mode, the O2CPRE level changes only when the output compare mode switches from “Timing mode” 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 CH2MS bit-filed is 00(COMPARE MODE).

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 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.</p>

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
Reserved	CH3P	CH3EN	Reserved	CH2P	CH2EN	Reserved	CH1P	CH1EN	Reserved	CH0P	CH0EN	Reserved	CH0P	CH0EN	rw

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
11:10	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
7:6	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
3:2	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.

0: CI0 is non-inverted. Input capture is done on a rising edge of CI0. When used as extern trigger, CI0 is non-inverted.

1: CI0 is inverted. Input capture is done on a falling edge of CI0. When used as extern trigger, CI0 is inverted.

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)

Address offset: 0x24

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
CNT[15:0]															
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)

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

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)

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

Bits	Fields	Descriptions
15:0	CARL[15:0]	<p>Counter auto reload value</p> <p>This bit-field specifies the auto reload value of the counter.</p> <p>Note: When the timer is configured in input capture mode, this register must be configured a non-zero value (such as 0xFFFF) which is larger than user expected value.</p>

### 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)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH0VAL[15:0]															
rw															

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-field indicates the counter value corresponding to the last capture event. And this bit-field is read-only.</p> <p>When channel 0 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.</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)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH1VAL[15:0]															

rw

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)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH2VAL[15:0]															

rw

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)

Address offset: 0x40

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
CH3VAL[15:0]															

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</p>

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)

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

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 5'b0_0000: 1 time transfer 5'b0_0001: 2 times transfer ... 5'b1_0001: 18 times transfer
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 ... 5'b1_0010: TIMERx_DMACFG 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)

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

rw

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
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>

## 15.3. Basic timer (TIMERx, x=5, 6)

### 15.3.1. Overview

The basic timer module (Timer5, 6) 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.

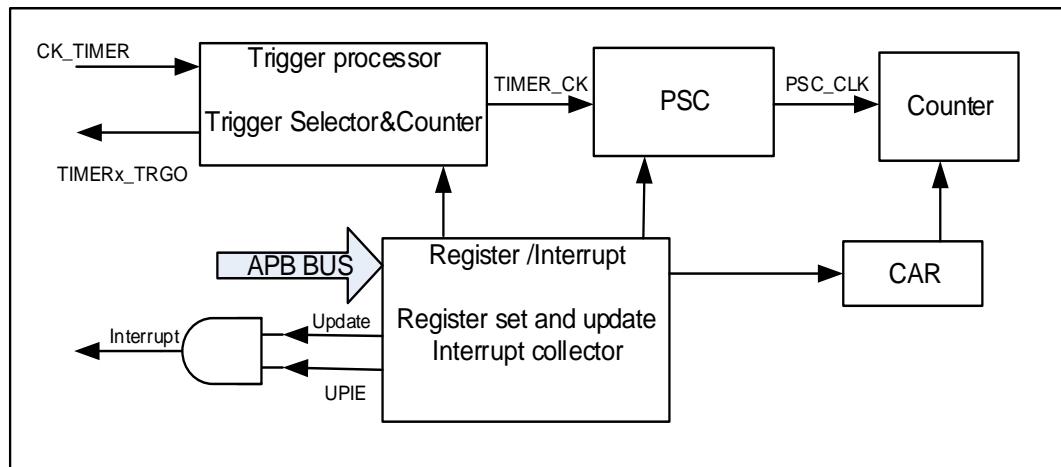
### 15.3.2. Characteristics

- Counter width: 16-bit.
- Source of count clock is internal clock only.
- Multiple counter modes: count up.
- Programmable prescaler: 16-bit. The factor can be changed ongoing.
- Single pulse mode is supported.
- Auto-reload function.
- Interrupt output or DMA request on update event.

### 15.3.3. Block diagram

[Figure 15-52. Basic timer block diagram](#) provides details on the internal configuration of the basic timer.

**Figure 15-52. Basic timer block diagram**



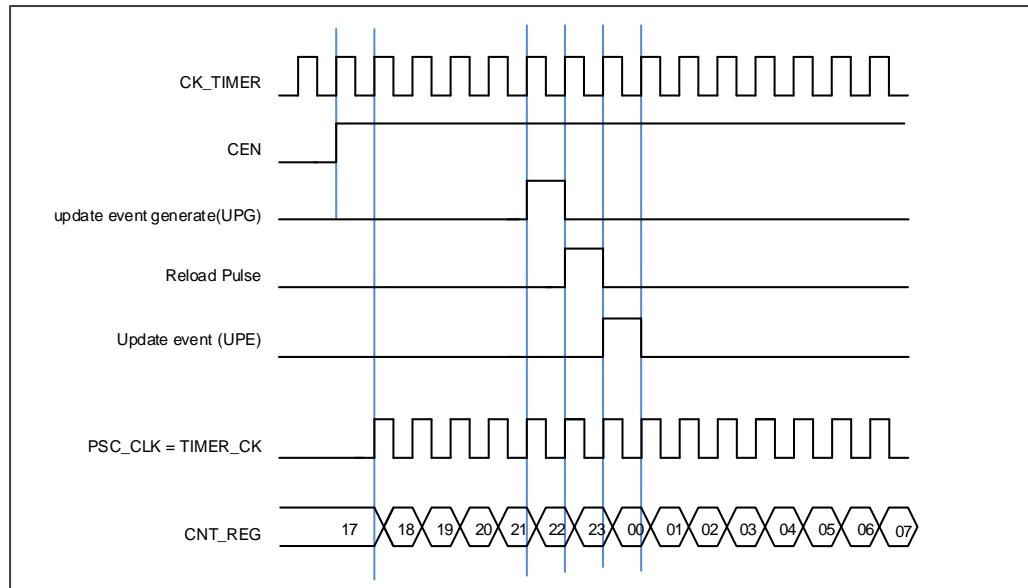
### 15.3.4. Function overview

#### 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.

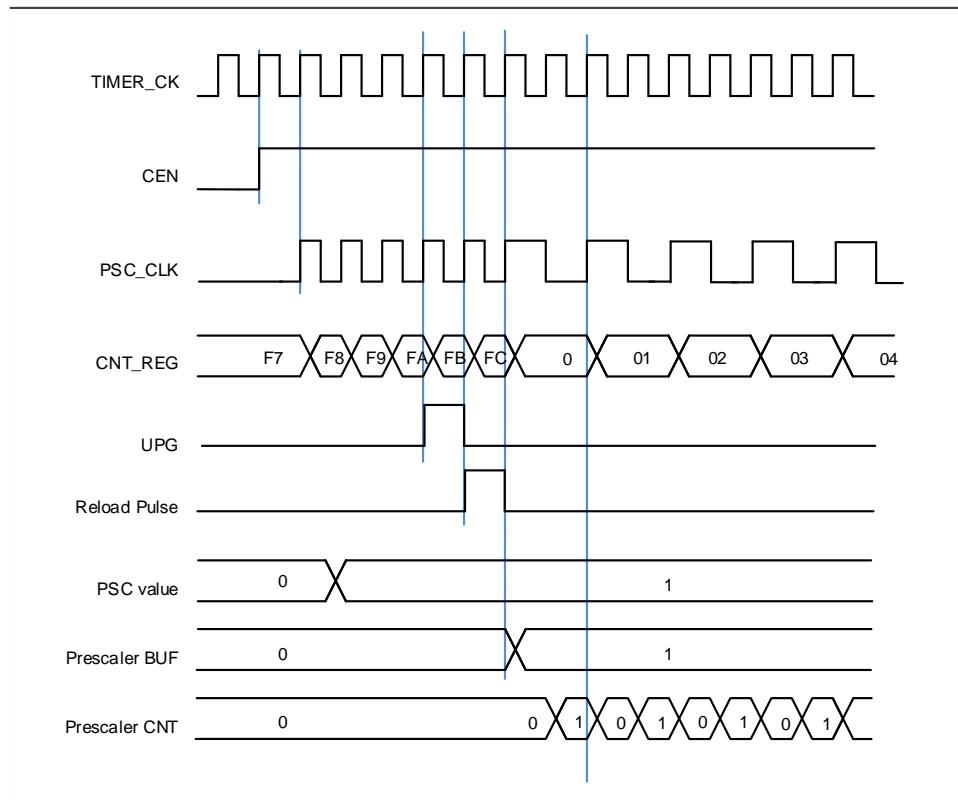
**Figure 15-53. Normal mode, internal clock divided by 1**



### Prescaler

The prescaler can divide the timer clock (TIMER\_CK) to a counter clock (PSC\_CLK) by any factor ranging from 1 to 65536. It is controlled by prescaler register (TIMERx\_PSC) which can be changed ongoing, but it is adopted at the next update event.

Figure 15-54. 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.

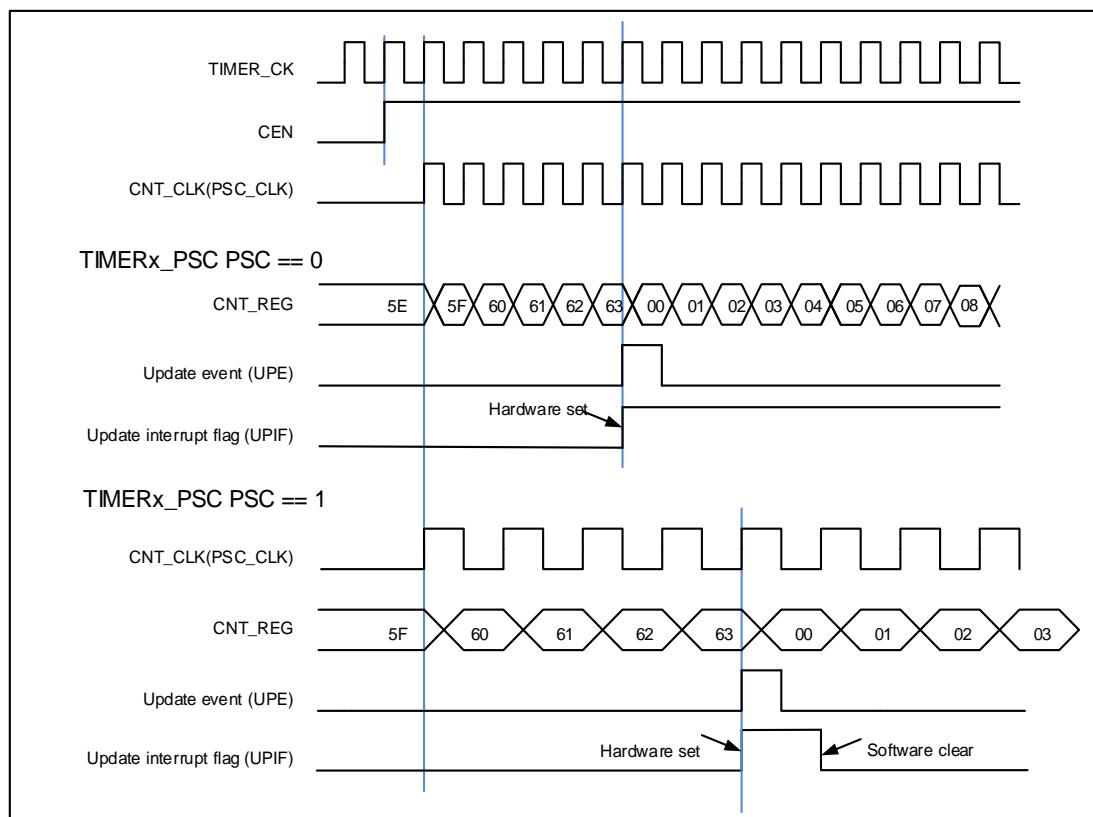
When the update event is set by the `UPG` bit in the `TIMERx_SWEVG` register, the counter value will be initialized to 0 and generates an update event.

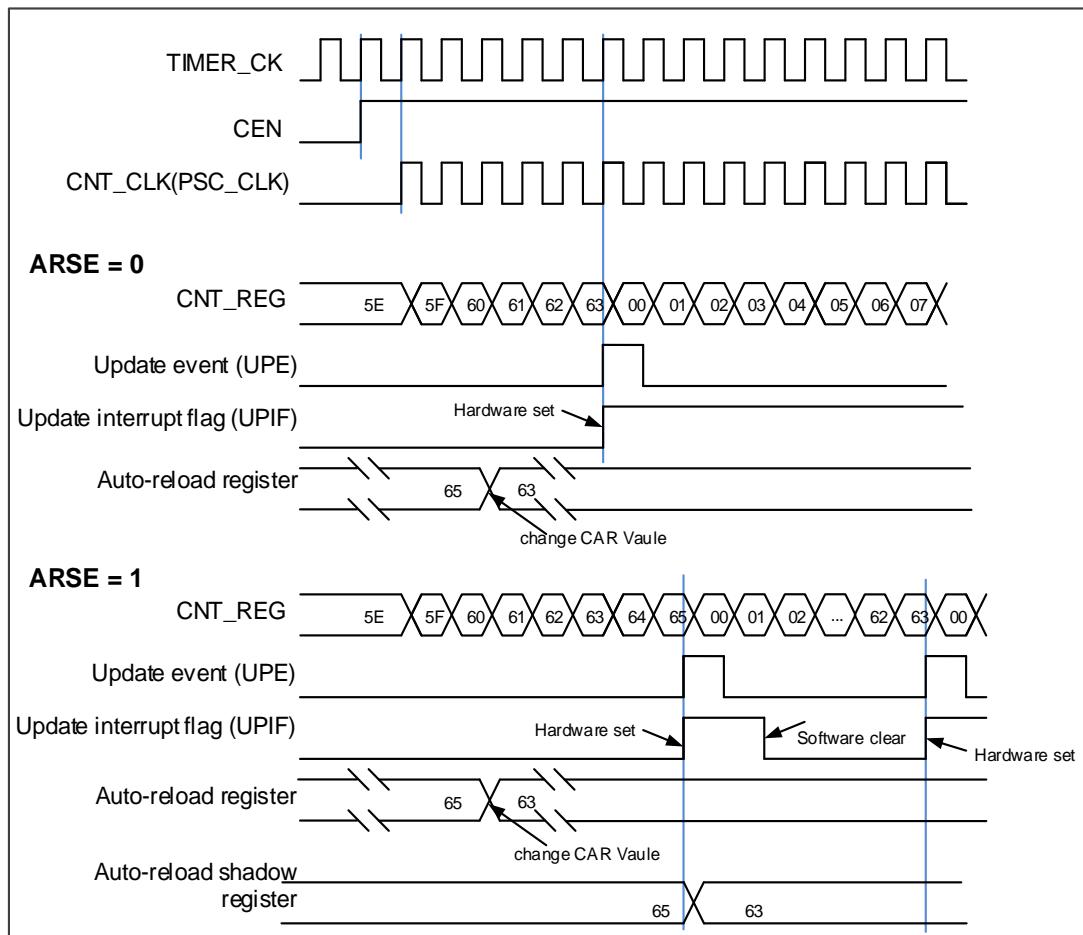
If set the `UPDIS` bit in `TIMERx_CTL0` register, the update event is disabled.

When an update event occurs, all the registers (repetition counter, auto reload register, prescaler register) are updated.

[Figure 15-55. Timing chart of up counting mode, PSC=0/1](#) and [Figure 15-56. Timing chart of up counting mode, change TIMERx\\_CAR ongoing](#) show some examples of the counter behavior for different clock prescaler factor when `TIMERx_CAR=0x63`.

Figure 15-55. Timing chart of up counting mode, PSC=0/1



**Figure 15-56. Timing chart of up counting mode, change TIMERx\_CAR ongoing**


### Timer debug mode

When the RISC-V core halted, and the TIMERx\_HOLD configuration bit in DBG\_CTL register set to 1, the TIMERx counter stops.

### **15.3.5. TIMERx registers(x=5,6)**

TIMER5 base address: 0x40001000

TIMER6 base address: 0x40001400

## 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				ARSE	Reserved			SPM	UPS	UPDIS	CEN	DW	DW	DW	DW

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: Single pulse mode disable. Counter continues after update event. 1: Single pulse mode enable. 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: The UPG bit is set The counter generates an overflow or underflow event The slave mode controller generates an update event. 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: 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						MMC[2:0]			Reserved						
rw															

Bits	Fields	Descriptions
15:7	Reserved	Must be kept at reset value
6:4	MMC[2:0]	<p>Master mode control</p> <p>These bits control the selection of TRGO signal, which is sent in master mode to slave timers for synchronization function.</p> <p>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.</p> <p>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.</p> <p>010: Update. In this mode the master mode controller selects the update event as TRGO.</p>
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)

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

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)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														UPIF	
rc_w0															

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)

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

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</p> <p>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)

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

Bits	Fields	Descriptions
15:0	CNT[15:0]	This bit-field 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)

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

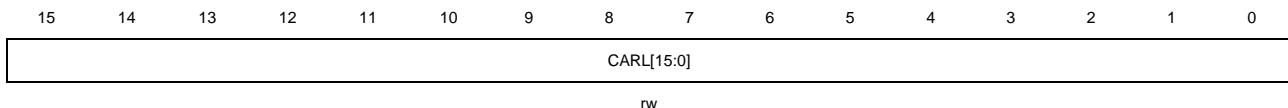
Bits	Fields	Descriptions
15:0	PSC[15:0]	<p>Prescaler value of the counter clock</p> <p>The PSC clock is divided by (PSC+1) to generate the counter clock. The value of this bit-filed will be loaded to the corresponding shadow register at every update event.</p>

### 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)



Bits	Fields	Descriptions
15:0	CARL[15:0]	<p>Counter auto reload value</p> <p>This bit-field specifies the auto reload value of the counter.</p> <p>Note: When the timer is configured in input capture mode, this register must be configured a non-zero value (such as 0xFFFF) which is larger than user expected value.</p>

## 16. Universal synchronous/asynchronous receiver/transmitter (USART)

### 16.1. 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 divides the peripheral clock (PCLK1 or PCLK2) to produce a dedicated baud rate lock 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, half-duplex mode and 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 USART supports DMA function for high-speed data communication, except UART4.

### 16.2. Characteristics

- NRZ standard format
- Asynchronous, full duplex communication
- Half duplex single wire communication
- Programmable baud-rate generator
  - Divided from the peripheral clocks, PCLK2 for USART0, PCLK1 for USART1/2 and USART3/4.
  - Oversampling by 16
  - Maximum speed up to 6.75 MBits/s (PCLK2 108M and oversampling by 16)
- Fully programmable serial interface characteristics:
  - Even, odd or no-parity bit generation/detection
  - A data word length can be 8 or 9 bits
  - 0.5, 1, 1.5 or 2 stop bit generation
- Transmitter and Receiver can be enabled separately
- Hardware flow control protocol (CTS/RTS)
- DMA request for data buffer access
- 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)

- Multiprocessor communication
  - Enter into mute mode if address match does not occur
  - Wake up from mute mode by idle frame or address match detection
- 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)
  - Interrupt occurs at these events when the corresponding interrupt enable bits are set

While USART0/1/2 is fully implemented, UART3/4 is only partially implemented with the following features not supported.

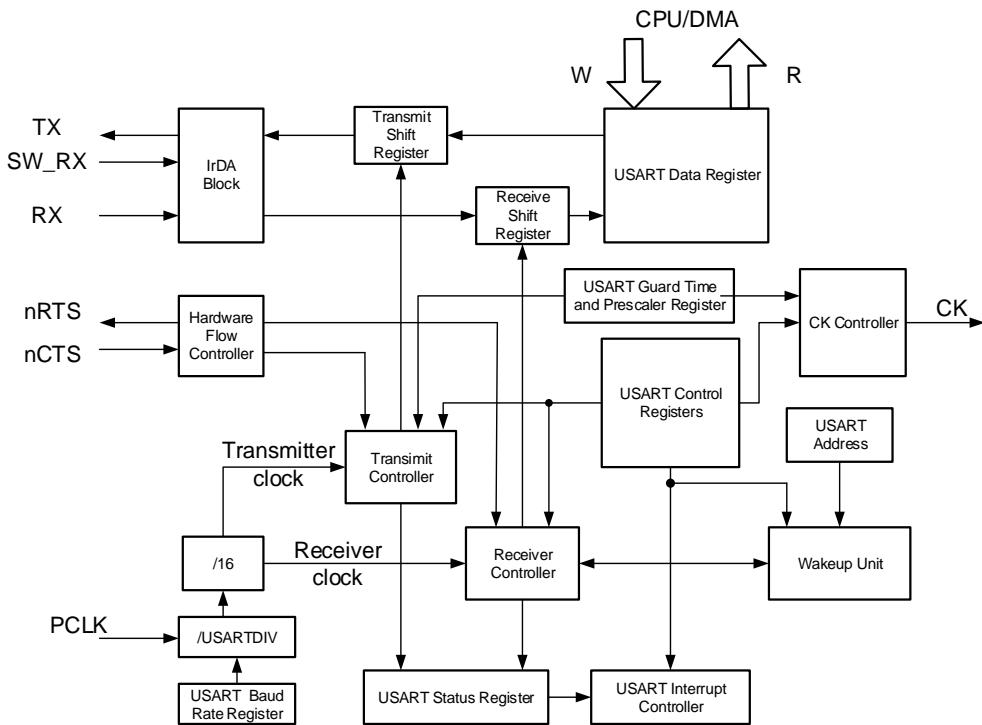
- Smartcard mode
- Synchronous mode
- Hardware flow control protocol (CTS/RTS)

### 16.3. Function overview

The interface is externally connected to another device by the main pins listed in [\*\*Table 16-1. Description of USART important pins.\*\*](#)

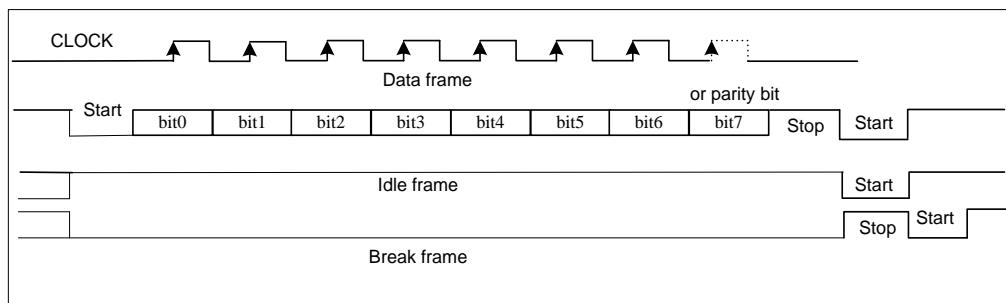
**Table 16-1. Description of USART important pins**

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 16-1. USART module block diagram**


### 16.3.1. 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 of in USART\_CTL0 register. When the WL bit is reset, the parity bit is the 7th bit. When the WL bit is set, the parity bit is the 8th bit. The method of calculating the parity bit is selected by the PM bit in USART\_CTL0 register.

**Figure 16-2. USART character frame (8 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 16-2. Stop bits configuration**

STB[1:0]	stop bit length (bit)	usage description
00	1	default value
01	0.5	Smartcard mode for receiving

STB[1:0]	stop bit length (bit)	usage description
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.

### 16.3.2. Baud rate generation

The baud-rate divider 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.

When oversampled by 16, the baud-rate divider (USARTDIV) has the following relationship with the peripheral clock:

$$\text{USARTDIV} = \frac{\text{PCLK}}{16 \times \text{Baud Rate}} \quad (16-1)$$

The peripheral clock is PCLK2 for USART0 and PCLK1 for USART1/2 and UART3/4. The peripheral clock must be enabled through the clock control unit before enabling the USART.

1. Get USARTDIV by calculating the value of USART\_BUAD:

If USART\_BUAD=0x21D, then INTDIV=33 (0x21), FRADIV=13 (0xD).

USARTDIV=33+13/16=33.81.

2. Get the value of USART\_BUAD by calculating the value of USARTDIV:

If USARTDIV=30.37, then INTDIV=30 (0x1E).

$16 \times 0.37 = 5.92$ , the nearest integer is 6, so FRADIV=6 (0x6).

USART\_BUAD=0x1E6.

**Note:** If the roundness of FRADIV is 16 (overflow), the carry must be added to the integer part.

### 16.3.3. 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. Clock pulses can output through the CK pin.

After the TEN bit is set, an idle frame will be sent. The TEN bit should not be cleared while

the transmission is ongoing.

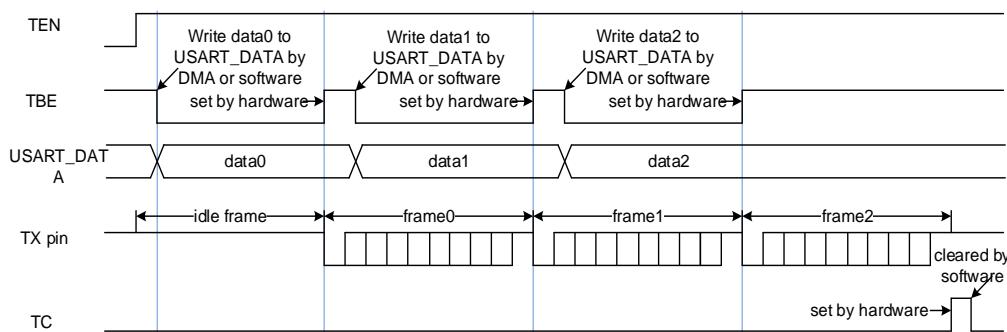
After power on, the TBE bit is high by default. Data can be written to the USART\_DATA when the TBE bit of the USART\_STAT register is asserted. The TBE bit is cleared by writing to the USART\_DATA register and it is set by hardware after the data is put into the transmit shift register. If a data is written to the USART\_DATA 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\_DATA 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 will be generated if the corresponding interrupt enable bit (TCIE) is set in the USART\_CTL0 register.

The USART transmit procedure is shown in [Figure 16-3. USART transmit procedure](#). The software operating process is as follows:

1. Write the WL bit in USART\_CTL0 to set the data bits length.
2. Set the STB[1:0] bits in USART\_CTL1 to configure the number of stop bits.
3. Enable DMA (DENT bit) in USART\_CTL2 if multibuffer communication is selected.
4. Set the baud rate in USART\_BAUD.
5. Set the TEN bit in USART\_CTL0.
6. Set the UEN bit in USART\_CTL0 to enable the USART
7. Wait for the TBE to be asserted.
8. Write the data to the USART\_DATA register.
9. Repeat step7-8 for each data, if DMA is not enabled.
10. Wait until TC=1 to finish.

**Figure 16-3. USART transmit procedure**



It is necessary to wait for the TC bit to be asserted before disabling the USART or entering the power saving mode. This bit can be cleared by a software sequence: reading the USART\_STAT register and then writing the USART\_DATA register. If the multibuffer communication is selected (DENT=1), this bit can also be cleared by writing 0 directly.

### 16.3.4. USART receiver

After power on, the USART receiver can be enabled by the follow procedure:

1. Write the WL bit in USART\_CTL0 to set the data bits length.
2. Set the STB[1:0] bits in USART\_CTL1.
3. Enable DMA (DENR bit) in USART\_CTL2 if multibuffer communication is selected.
4. Set the baud rate in USART\_BAUD.
5. Set the REN bit in USART\_CTL0.
6. Set the UEN bit in USART\_CTL0 to enable the USART.

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 is performed during the reception of a frame.

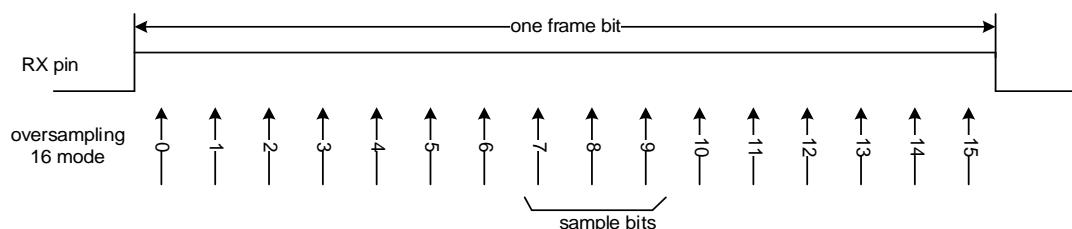
When a frame is received, the RBNE bit in USART\_STAT is asserted, an interrupt is generated if the corresponding interrupt enable bit (RBNEIE) is set in the USART\_CTL0 register. The status of the reception are stored in the USART\_STAT register.

The software can get the received data by reading the USART\_DATA register directly, or through DMA. The RBNE bit is cleared by a read operation on the USART\_DATA 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. While in the oversampling 16 mode, the 7th, 8th, and 9th samples are used. If two or more samples of a frame bit is 0, the frame bit is confirmed as a 0, else 1. If the value of the three samples of any bit are not the same, whatever it is a start bit, data bit, parity bit or stop bit, a noisy error (NERR) will be generated for the frame. An interrupt will be generated, if the receive DMA is enabled and the ERRIE bit in USART\_CTL2 register is set.

**Figure 16-4. Receiving a frame bit by oversampling method**



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.

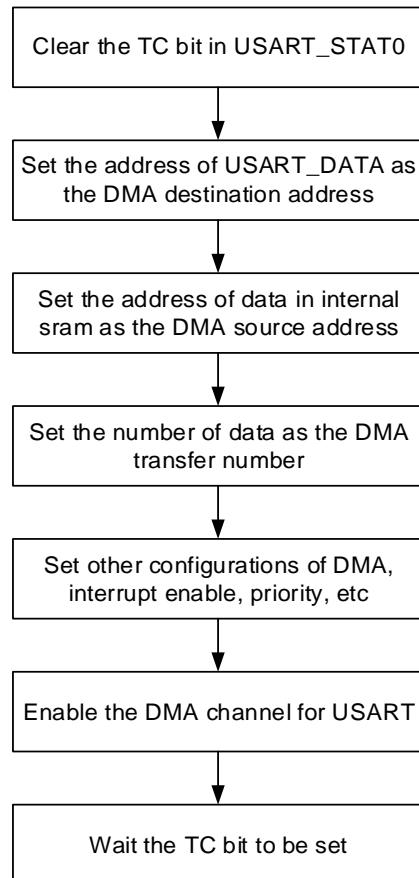
The RBNE, NERR, PERR, FERR and ORERR flags are always set at the same time in a reception. If the receive DMA is not enabled, software can check NERR, PERR, FERR and ORERR flags when serving the RBNE interrupt.

### 16.3.5. 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 steps are shown in [Figure 16-5. Configuration steps when using DMA for USART transmission](#).

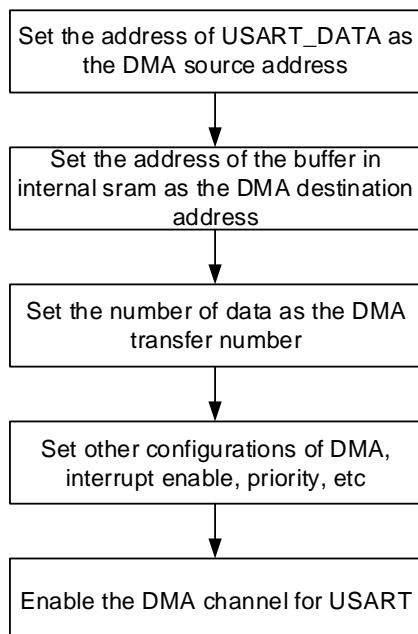
**Figure 16-5. Configuration steps when using DMA for USART transmission**



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 steps are shown in [Figure 16-6. Configuration steps 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.

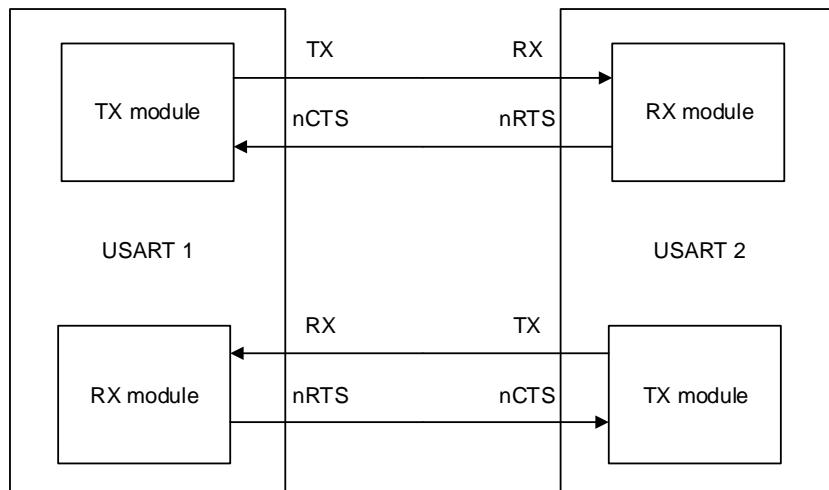
**Figure 16-6. Configuration steps when using DMA for USART reception**



When the number of the data received by USART reaches the DMA transfer number, an end of transfer interrupt will be generated in the DMA module.

### 16.3.6. 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.

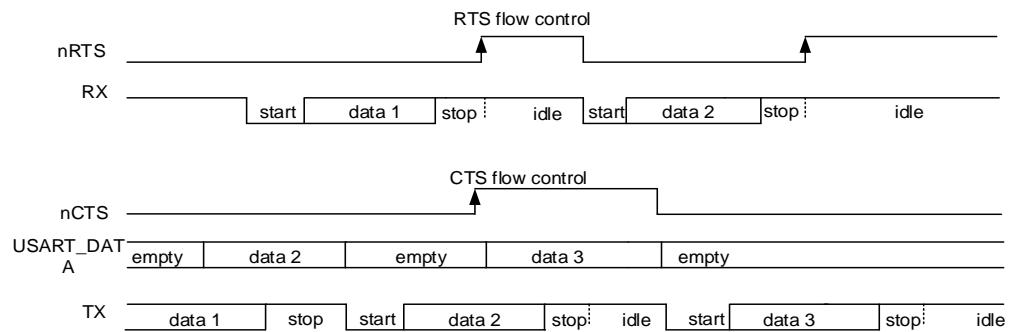
**Figure 16-7. Hardware flow control between two USARTs**


### RTS flow control

The USART receiver outputs the nRTS, which reflects the status of the receive buffer. When data frame is received, the nRTS signal goes high to prevent the transmitter from sending next frame. The nRTS signal keeps high when the receive buffer is full, and can be cleared by reading the USART\_DATA register.

### CTS flow control

The USART transmitter monitors the nCTS input pin to decide whether a data frame can be transmitted. 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 16-8. Hardware flow control**


If the CTS flow control is enabled, the CTSF bit in USART\_STAT is set when the nCTS pin toggles. An interrupt is generated if the CTSIE bit in USART\_CTL2 is set.

### 16.3.7. 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 setting the RWU bit in USART\_CTL0 register.

If a USART is in mute mode, all of the receive status bits cannot be set. Software can wake up the USART by clearing the RWU bit.

The USART can also be woken 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 it is woken up by an idle frame, the IDLEF bit in USART\_STAT will not be 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 bits of an address frame are the same as the ADDR[3:0] bits in the USART\_CTL1 register, the hardware will clear the RWU bit and exits the mute mode. The RBNE bit will be set when the frame that wakes up the USART. The status bits are available in the USART\_STAT register. If the LSB 4 bits of an address frame differ from the ADDR[3:0] 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 address match method is selected, the receiver does not check the parity value of an address frame by default. 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 bit.

### 16.3.8. LIN mode

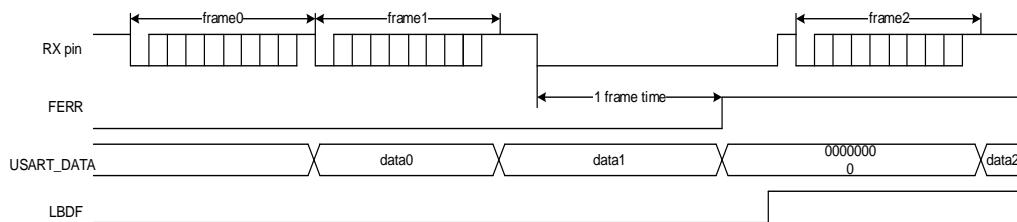
The local interconnection network mode is enabled by setting the LMEN bit in USART\_CTL1. The CKEN, WL, STB[1:0] bits in USART\_CTL1 and the SCEN, HDEN, IREN bits in USART\_CTL2 should be cleared in LIN mode.

When transmitting a normal data frame, the transmission procedure is the same as the normal USART mode. When the SBKCMD bit in USART\_CTL0 is set, the USART transmits 13 '0' bits continuously, followed by 1 stop bit.

The break detection function is totally independent of 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 configuring LBLEN bit 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.

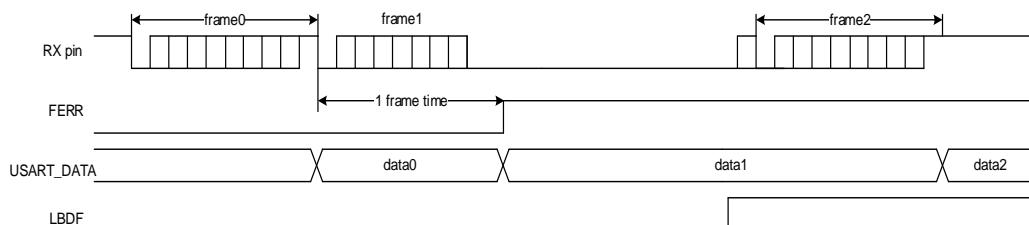
As shown in [\*\*Figure 16-9. Break frame occurs during idle state\*\*](#), if a break frame occurs during the idle state on the RX pin, the USART receiver will receive an all '0' frame, with an asserted FERR status.

**Figure 16-9. Break frame occurs during idle state**



As shown in [Figure 16-10. 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 16-10. Break frame occurs during a frame**

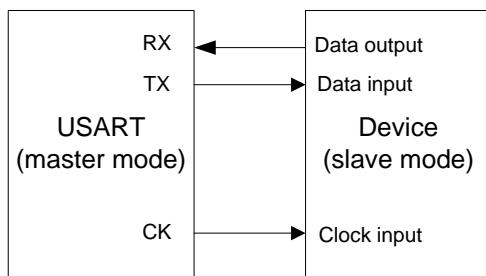
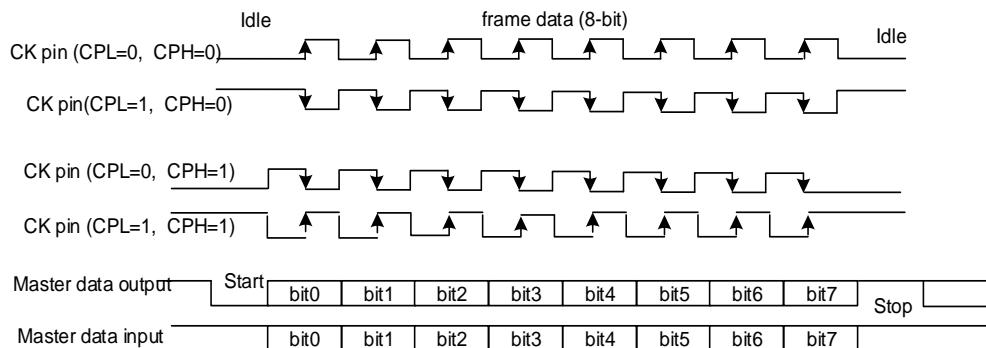


### 16.3.9. Synchronous mode

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 cleared in synchronous mode. The CK pin is the clock output of the synchronous USART transmitter, and can be only activated when the TEN bit is enabled. No clock pulse will be sent through the CK pin during the transmission of the start bit and stop bit. The CLEN bit in USART\_CTL1 can be used to determine whether the clock is output or not during the last (address flag) bit transmission. 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 idle state.

The CPL, CPH and CLEN bits in USART\_CTL1 determine the waveform on the CK pin. Software can only change them when the USART is disabled (UEN=0).

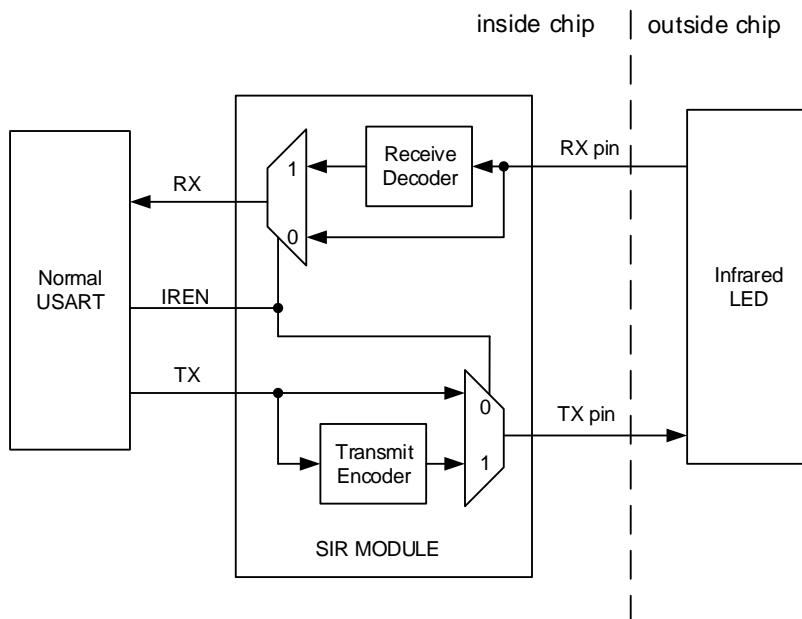
If the REN bit in USART\_CTL0 is set, the receiver works differently from the normal USART reception method. The receiver samples the data on the capture edge of the CK pin without any oversampling.

**Figure 16-11. Example of USART in synchronous mode**

**Figure 16-12. 8-bit format USART synchronous waveform (CLEN=1)**


### 16.3.10. IrDA SIR ENDEC mode

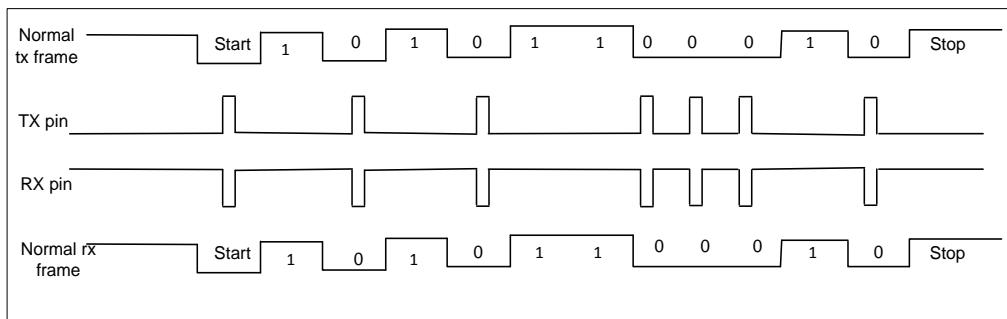
The IrDA mode is enabled by setting the IREN bit in USART\_CTL2. The LMEN, STB[1:0], CKEN bits in USART\_CTL1 and HDEN, SCEN bits in USART\_CTL2 should be cleared in IrDA mode.

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 16-13. IrDA SIR ENDEC module**


In IrDA mode, the polarity of the TX pin and RX pin 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 of 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 16-14. IrDA data modulation**


The SIR sub module 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 division 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 period. The receiver decoder works in the same manner as the normal IrDA mode.

### 16.3.11. 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 cleared in half-duplex communication mode.

In the half-duplex mode the receive line is internally connected to the TX pin, and the RX pin is no longer used. The TX pin should be configured as output open drain mode. The software should make sure that the transmission and reception process never conflict with each other.

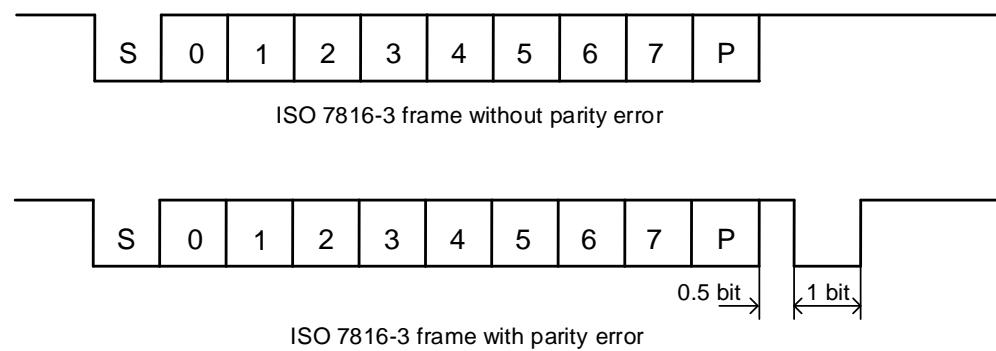
### 16.3.12. Smartcard (ISO7816-3) mode

The smartcard mode is an asynchronous mode, which is designed to support the ISO7816-3 protocol. The smartcard mode 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 external smart card through the CK pin after the CKEN bit is set. The clock is divided from the PCLK. The divide ratio is configured by the PSC[4:0] bits in USART\_GP register. The CK pin only provides a clock source to the smartcard.

The smartcard mode is a half-duplex communication protocol. When connected to a smartcard, the TX pin must be configured as open drain, and an external pull-up resistor will be needed, which drives a bidirectional line that is also driven by the smartcard. The data frame consists of 1 start bit, 9 data bits (1 parity bit included) and 1.5 stop bits. The 0.5 stop bit may be configured for a receiver.

**Figure 16-15. ISO7816-3 frame format**



#### Character (T=0) mode

Compared to the timing in normal operation, the transmission time from transmit shift register to the TX pin is delayed by half baud clock, and the TC flag assertion time is delayed by a guard time that is configured by the GUAT[7:0] bits in USART\_GP. In Smartcard mode, the internal guard time counter starts counting up after the stop bits of the last data frame, and the GUAT[7:0] bits should be configured as the character guard time (CGT) in ISO7816-3

protocol minus 12. The TC status is forced reset while the guard time counter is counting up. When the counter reaches the programmed value TC is asserted high.

During USART transmission, if a parity error event is detected, the smartcard may NACK the current frame by pulling down the TX pin during the last 1 bit time of the stop bits. The USART can automatically resend data according to the protocol. The USART will not take the NACK signal as the start bit.

During USART reception, if the parity error is detected in the current frame, the TX pin is pulled low during the last 1 bit time of the stop bits. This signal is the NACK signal to smart card. Then a frame error occurs in smart card side. The RBNE/receive DMA request is not activated if the received character is erroneous. According to the protocol, the smart card can resend the data. The NACK signal is enabled by setting the NKEN bit in USART\_CTL2.

The idle frame and break frame are not applied for the Smartcard mode.

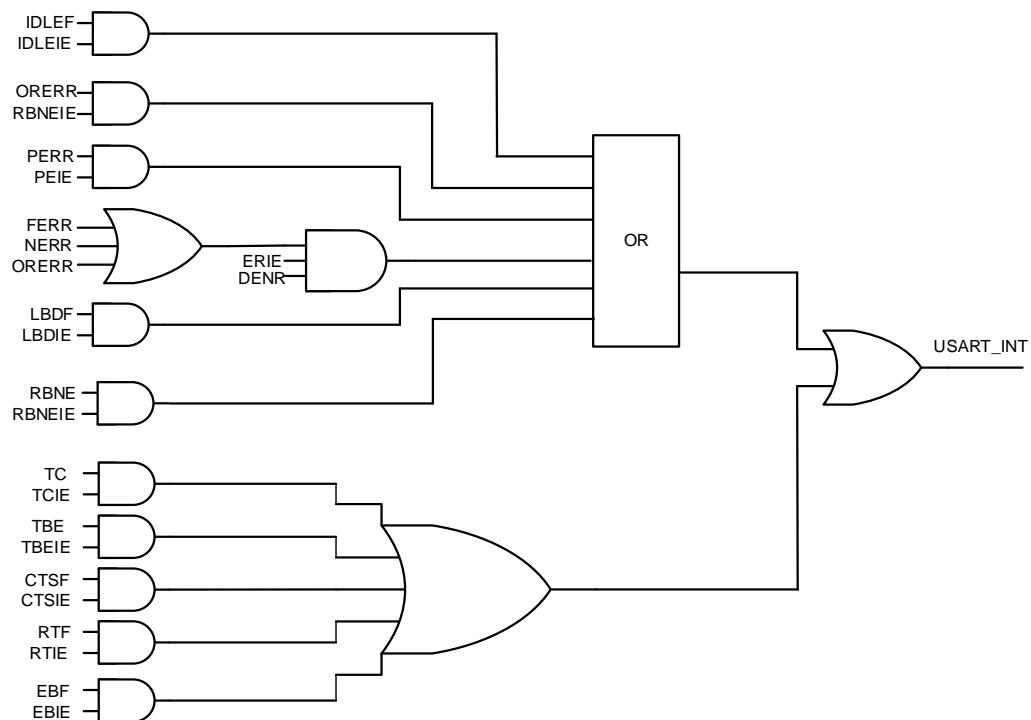
### 16.3.13. USART interrupts

The USART interrupt events and flags are listed in [Table 16-3. USART interrupt requests](#).

**Table 16-3. USART interrupt requests**

Interrupt event	Event flag	Control register	Enable Control bit
Transmit data buffer empty	TBE	USART_CTL0	TBEIE
CTS toggled flag	CTSF	USART_CTL2	CTSIE
Transmission complete	TC	USART_CTL0	TCIE
Received buff not empty	RBNE	USART_CTL0	RBNEIE
Overrun error	ORERR		
Idle frame	IDLEF	USART_CTL0	IDLEIE
Parity error	PERR	USART_CTL0	PERRIE
Break detected flag in LIN mode	LBDF	USART_CTL1	LBDIE
Reception Errors (Noise flag, overrun error, framing error) in DMA reception	NERR or ORERR or FERR	USART_CTL2	ERRIE

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 16-16. USART interrupt mapping diagram**


## 16.4. Register definition

USART0 base address: 0x4001 3800

USART1 base address: 0x4000 4400

USART2 base address: 0x4000 4800

UART3 base address: 0x4000 4C00

UART4 base address: 0x4000 5000

### 16.4.1. Status register (USART\_STAT)

Address offset: 0x00

Reset value: 0x0000 00C0

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

Bits	Fields	Descriptions
31:10	Reserved	Must be kept the reset value
9	CTSF	<p>CTS change flag</p> <p>If CTSEN bit in USART_CTL2 is set, this bit is set by hardware when the nCTS input toggles. An interrupt occurs if the CTSIE bit in USART_CTL2 is set.</p> <p>Software can clear this bit by writing 0 to it.</p> <p>0: The status of the nCTS line does not change</p> <p>1: The status of the nCTS line has changed</p> <p>This bit is reserved for UART3/4.</p>
8	LBDF	<p>LIN break detected flag</p> <p>This bit is set when LIN break is detected. An interrupt occurs if the LBDIE bit in USART_CTL1 is set.</p> <p>Software can clear this bit by writing 0 to it.</p> <p>0: The USART does not detect a LIN break</p> <p>1: The USART has detected a LIN break</p>
7	TBE	<p>Transmit data buffer empty</p> <p>This bit is set after power on or when the transmit data has been transferred to the transmit shift register. An interrupt occurs if the TBEIE bit in USART_CTL0 is set.</p> <p>This bit is cleared when the software writes transmit data to the USART_DATA register.</p> <p>0: Transmit data buffer is not empty</p> <p>1: Transmit data buffer is empty</p>
6	TC	<p>Transmission complete</p> <p>This bit is set after power on. If the TBE bit has been set, this bit is set when the transmission of current data is complete. An interrupt occurs if the TCIE bit in USART_CTL0 is set.</p>

		Software can clear this bit by writing 0 to it. 0: Transmission of current data is not complete 1: Transmission of current data is complete
5	RBNE	<p>Read data buffer not empty</p> <p>This bit is set when the read data buffer is filled with a data frame, which has been received through the receive shift register. An interrupt occurs if the RBNEIE bit in USART_CTL0 is set.</p> <p>Software can clear this bit by writing 0 to it or by reading the USART_DATA register.</p> <p>0: Read data buffer is empty 1: Read data buffer is not empty</p>
4	IDLEF	<p>IDLE frame detected flag</p> <p>This bit is set when the RX pin has been detected in idle state for a frame time. An interrupt occurs if the IDLEIE bit in USART_CTL0 is set.</p> <p>Software can clear this bit by reading the USART_STAT and USART_DATA registers one by one.</p> <p>0: The USART module does not detect an IDLE frame 1: The USART module has detected an IDLE frame</p>
3	ORERR	<p>Overrun error flag</p> <p>This bit is set if the RBNE is not cleared and a new data frame is received through the receive shift register. An interrupt occurs if RBNEIE bit in USART_CTL0 is set.</p> <p>In multi-processor communication or DMA mode, an interrupt occurs if ERRIE bit in USART_CTL2 is set.</p> <p>Software can clear this bit by reading the USART_STAT and USART_DATA registers one by one.</p> <p>0: The USART does not detect a overrun error 1: The USART has detected a overrun error</p>
2	NERR	<p>Noise error flag</p> <p>This bit is set if the USART detects noise on the RX pin when receiving a frame. An interrupt occurs if the ERRIE bit in USART_CTL2 is set.</p> <p>Software can clear this bit by reading the USART_STAT and USART_DATA registers one by one.</p> <p>0: The USART does not detect a noise error 1: The USART has detected a noise error</p>
1	FERR	<p>Frame error flag</p> <p>This bit is set when the RX pin is detected low during the stop bits of a receive frame. An interrupt occurs if the ERRIE bit in USART_CTL2 is set.</p> <p>Software can clear this bit by reading the USART_STAT and USART_DATA registers one by one.</p> <p>0: The USART does not detect a framing error 1: The USART has detected a framing error</p>
0	PERR	Parity error flag

This bit is set when the parity bit of a receive frame does not match the expected parity value. An interrupt occurs if the PERRIE bit in USART\_CTL0 is set.

Software can clear this bit in the sequence: read the USART\_STAT register, and then read or write the USART\_DATA register.

0: The USART does not detect a parity error

1: The USART has detected a parity error

### 16.4.2. Data register (USART\_DATA)

Offset: 0x04

Reset value: Undefined

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								DATA[8:0]							
rw															

Bits	Fields	Descriptions
31:9	Reserved	Must be kept the reset value
8:0	DATA[8:0]	<p>Transmitted or received data value</p> <p>Software can write these bits to update the transmitted data or read these bits to get the received data.</p> <p>If the parity check function is enabled, when transmitted data is written to this register, the MSB bit (bit 7 or bit 8 depending on the WL bit in USART_CTL0) will be replaced by the parity bit.</p>

### 16.4.3. Baud rate register (USART\_BAUD)

Address offset: 0x08

Reset value: 0x0000 0000

The software must not write this register when the USART is enabled (UEN=1).

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
INTDIV [11:0]								FRADIV[3:0]							
rw															

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:16	Reserved	Must be kept the reset value
15:4	INTDIV[11:0]	Integer part of baud-rate divider
3:0	FRADIV[3:0]	Fraction part of baud-rate divider

#### 16.4.4. Control register 0 (USART\_CTL0)

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															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	UEN	WL	WM	PCEN	PM	PERRIE	TBEIE	TCIE	RBNEIE	IDLEIE	TEN	REN	RWU	SBKCMD	rw

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:14	Reserved	Must be kept the reset value
13	UEN	USART enable 0: USART disabled 1: USART enabled
12	WL	Word length 0: 8 data bits 1: 9 data bits
11	WM	Wakeup method in mute mode 0: Wake up by idle frame 1: Wake up by address match
10	PCEN	Parity check function enable 0: Parity check function disabled 1: Parity check function enabled
9	PM	Parity mode 0: Even parity 1: Odd parity
8	PERRIE	Parity error interrupt enable If this bit is set, an interrupt occurs when the PERR bit in USART_STAT is set. 0: Parity error interrupt is disabled 1: Parity error interrupt is enabled

7	TBEIE	<p>Transmitter buffer empty interrupt enable</p> <p>If this bit is set, an interrupt occurs when the TBE bit in USART_STAT is set.</p> <p>0: Transmitter buffer empty interrupt is disabled</p> <p>1: Transmitter buffer empty interrupt is enabled</p>
6	TCIE	<p>Transmission complete interrupt enable</p> <p>If this bit is set, an interrupt occurs when the TC bit in USART_STAT is set.</p> <p>0: Transmission complete interrupt is disabled</p> <p>1: Transmission complete interrupt is enabled</p>
5	RBNEIE	<p>Read data buffer not empty interrupt and overrun error interrupt enable</p> <p>If this bit is set, an interrupt occurs when the RBNE bit or the ORERR bit in USART_STAT is set.</p> <p>0: Read data register not empty interrupt and overrun error interrupt disabled</p> <p>1: Read data register not empty interrupt and overrun error interrupt enabled</p>
4	IDLEIE	<p>IDLE line detected interrupt enable</p> <p>If this bit is set, an interrupt occurs when the IDLEF bit in USART_STAT is set.</p> <p>0: IDLE line detected interrupt disabled</p> <p>1: IDLE line detected interrupt enabled</p>
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</p>
1	RWU	<p>Receiver wakes up from mute mode.</p> <p>Software can set this bit to make the USART work in mute mode and clear this bit to wake up the USART.</p> <p>If it is configured to wake up by idle frame (WM=0), this bit can be cleared by hardware when an idle frame has been detected. If it is configured to wake up by address matching (WM=1), this bit can be cleared by hardware when receiving an address match frame or set by hardware when receiving an address mismatch frame.</p> <p>0: Receiver in active mode</p> <p>1: Receiver in mute mode</p>
0	SBKCMD	<p>Send break command</p> <p>Software can set this bit to send a break frame.</p> <p>Hardware clears this bit automatically when the break frame has been transmitted.</p> <p>0: Do not transmit a break frame</p> <p>1: Transmit a break frame</p>

### 16.4.5. Control register 1 (USART\_CTL1)

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															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	LMEN	STB[1:0]	CKEN	CPL	CPH	CLEN	Reserved	LBDIE	LBLEN	Reserved					ADDR[3:0]

rw                         rw

Bits	Fields	Descriptions
31:15	Reserved	Must be kept the reset value
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).
13:12	STB[1:0]	Stop bits length 00: 1 stop bit 01: 0.5 stop bit 10: 2 stop bits 11: 1.5 stop bits This bit field cannot be written when the USART is enabled (UEN=1). Only 1 stop bit and 2 stop bits are available for UART3/4.
11	CKEN	CK pin enable 0: CK pin disabled 1: CK pin enabled This bit field cannot be written when the USART is enabled (UEN=1). This bit is reserved for UART3/4.
10	CPL	CK polarity This bit specifies the polarity of the CK pin in synchronous mode. 0: The CK pin is in low state when the USART is in idle state 1: The CK pin is in high state when the USART is in idle state This bit field cannot be written when the USART is enabled (UEN=1). This bit is reserved for UART3/4.
9	CPH	CK phase This bit specifies the phase of the CK pin in synchronous mode. 0: The capture edge of the LSB bit is the first edge of CK pin 1: The capture edge of the LSB bit is the second edge of CK pin

This bit field cannot be written when the USART is enabled (UEN=1).

This bit is reserved for UART3/4.

8	CLEN	CK length
		This bit specifies the length of the CK signal in synchronous mode.
	0:	There are 7 CK pulses for an 8 bit frame and 8 CK pulses for a 9 bit frame
	1:	There are 8 CK pulses for an 8 bit frame and 9 CK pulses for a 9 bit frame
		This bit field cannot be written when the USART is enabled (UEN=1).
		This bit is reserved for UART3/4.
7	Reserved	Must be kept the reset value
6	LBDIE	LIN break detected interrupt enable
		If this bit is set, an interrupt occurs when the LBDF bit in USART_STAT is set.
	0:	LIN break detected interrupt is disabled
	1:	LIN break detected interrupt is enabled
5	LBLEN	LIN break frame length
		This bit specifies the length of a LIN break frame.
	0:	10 bits
	1:	11 bits
		This bit field cannot be written when the USART is enabled (UEN=1).
4	Reserved	Must be kept the reset value
3:0	ADDR[3:0]	Address of the USART
		If it is configured toIn wake up by address matching (WM=1), the USART enters mute mode when the LSB 4 bits of a received frame do not equal the ADDR[3:0] bits, and wakes up when the LSB 4 bits of a received frame equal the ADDR[3:0] bits.

#### 16.4.6. Control register 2 (USART\_CTL2)

Address offset: 0x14

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					CTSIE	CTSEN	RTSEN	DENT	DENR	SCEN	NKEN	HDEN	IRLP	IREN	ERRIE
					rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:11	Reserved	Must be kept the reset value
10	CTSIE	<p>CTS interrupt enable</p> <p>If this bit is set, an interrupt occurs when the CTSF bit in USART_STAT is set.</p> <p>0: CTS interrupt disabled</p> <p>1: CTS interrupt enabled</p> <p>This bit is reserved for UART3/4.</p>
9	CTSEN	<p>CTS enable</p> <p>This bit enables the CTS hardware flow control function.</p> <p>0: CTS hardware flow control disabled</p> <p>1: CTS hardware flow control enabled</p> <p>This bit field cannot be written when the USART is enabled (UEN=1).</p> <p>This bit is reserved for UART3/4.</p>
8	RTSEN	<p>RTS enable</p> <p>This bit enables the RTS hardware flow control function.</p> <p>0: RTS hardware flow control disabled</p> <p>1: RTS hardware flow control enabled</p> <p>This bit field cannot be written when the USART is enabled (UEN=1).</p> <p>This bit is reserved for UART3/4.</p>
7	DENT	<p>DMA request enable for transmission</p> <p>0: DMA request is disabled for transmission</p> <p>1: DMA request is enabled for transmission</p>
6	DENR	<p>DMA request enable for reception</p> <p>0: DMA request is disabled for reception</p> <p>1: DMA request is enabled for reception</p>
5	SCEN	<p>Smartcard mode enable</p> <p>This bit enables the smartcard work mode.</p> <p>0: Smartcard mode disabled</p> <p>1: Smartcard mode enabled</p> <p>This bit field cannot be written when the USART is enabled (UEN=1).</p> <p>This bit is reserved for UART3/4.</p>
4	NKEN	<p>NACK enable in Smartcard mode</p> <p>This bit enables the NACK transmission when parity error occurs in smartcard mode.</p> <p>0: Disable NACK transmission</p> <p>1: Enable NACK transmission</p> <p>This bit field cannot be written when the USART is enabled (UEN=1).</p> <p>This bit is reserved for UART3/4.</p>
3	HDEN	<p>Half-duplex enable</p> <p>This bit enables the half-duplex USART mode.</p>

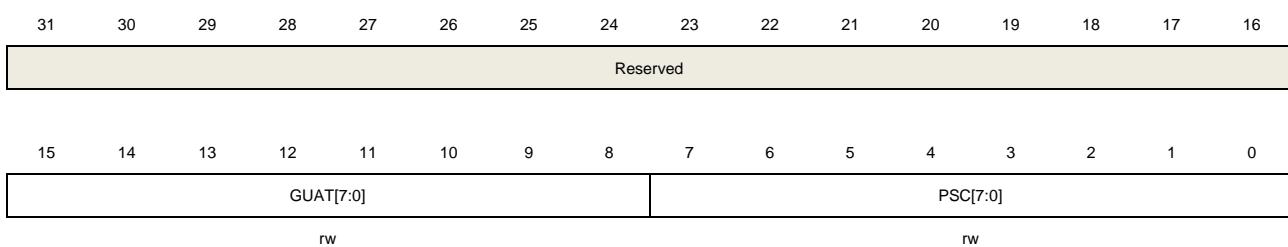
		0: Half duplex mode is disabled 1: Half duplex mode is enabled This bit field cannot be written when the USART is enabled (UEN=1).
2	IRLP	IrDA low-power This bit selects low-power mode of IrDA mode. 0: Normal mode 1: Low-power mode This bit field cannot be written when the USART is enabled (UEN=1).
1	IREN	IrDA mode enable This bit enables the IrDA mode of USART. 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 When DMA request for reception is enabled (DENR=1), if this bit is set, an interrupt occurs when any one of the FERR, ORERR and NERR bits in USART_STAT is set. 0: Error interrupt disabled 1: Error interrupt enabled

#### 16.4.7. Guard time and prescaler register (USART\_GP)

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 the reset value.
15:8	GUAT[7:0]	Guard time value in Smartcard mode TC flag assertion time is delayed by GUAT[7:0] baud clock cycles. This bit field cannot be written when the USART is enabled (UEN=1). These bits are reserved for UART3/4.
7:0	PSC[7:0]	When the USART IrDA low-power mode is enabled, these bits specify the division factor that is used to divide the peripheral clock (PCLK1/PCLK2) to generate the

low-power frequency.

00000000: Reserved - never program this value

00000001: Divided by 1

00000010: Divided by 2

...

11111111: Divided by 255

When the USART works in IrDA normal mode, these bits must be set to 00000001.

When the USART smartcard mode is enabled, the PSC [4:0] bits specify the division factor that is used to divide the peripheral clock (APB1/APB2) to generate the smartcard clock (CK). The actual division factor is twice as the PSC [4:0] value.

00000: Reserved - never program this value

00001: Divided by 2

00010: Divided by 4

...

11111: Divided by 62

The PSC [7:5] bits are reserved in smartcard mode.

This bit field cannot be written when the USART is enabled (UEN=1).

## 17. Inter-integrated circuit interface (I2C)

### 17.1. 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.

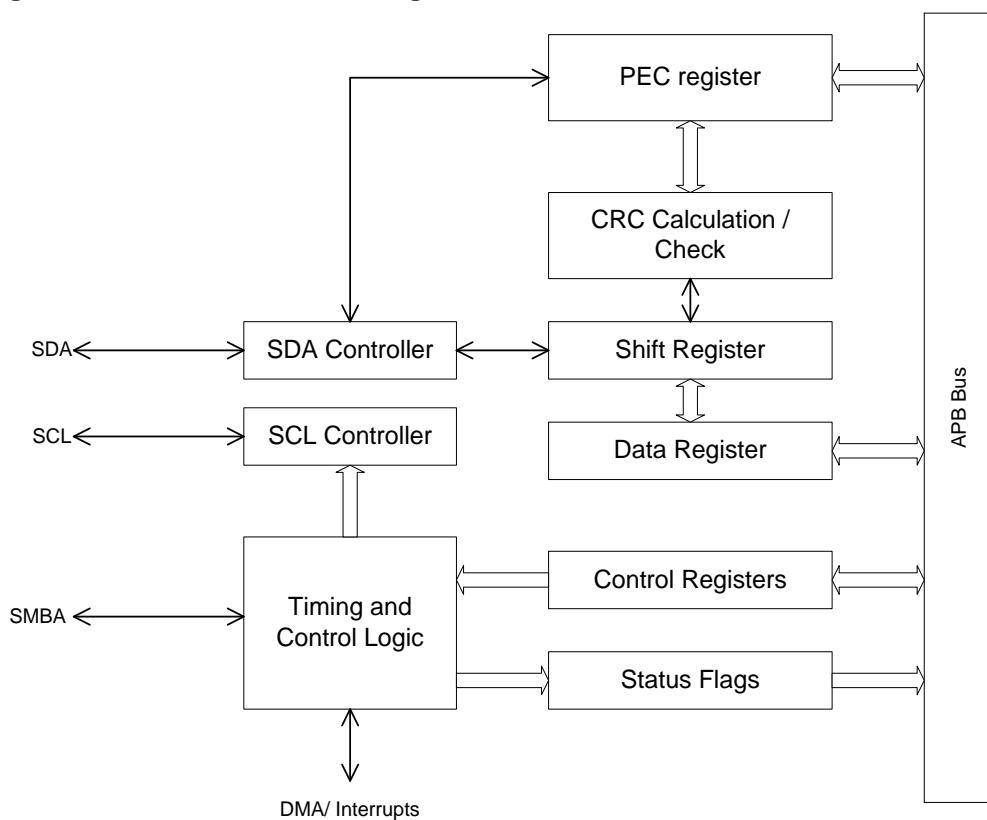
The I2C interface implements standard I2C protocol with standard-mode and fast-mode as well as CRC calculation and checking, SMBus (system management bus) and PMBus (power management bus). It also supports multi-master I2C bus. The I2C interface provides DMA mode for users to reduce CPU overload.

### 17.2. 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.

### 17.3. Function overview

[Figure 17-1. I2C module block diagram](#) below provides details on the internal configuration of the I2C interface.

**Figure 17-1. I2C module block diagram**

**Table 17-1. 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 signal 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 signal 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

### 17.3.1. 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.

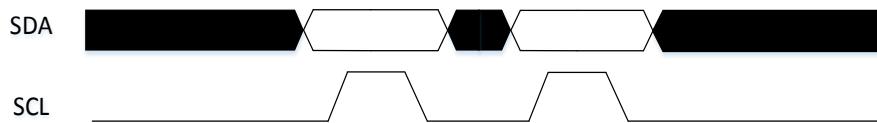
Both SDA and SCL are bidirectional lines, connected to a positive supply voltage via current-source or pull-up resistor. When the bus is free, both lines are HIGH. The output stages of

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<sub>DD</sub>.

### 17.3.2. 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 17-2. Data validation](#)). One clock pulse is generated for each data bit transferred.

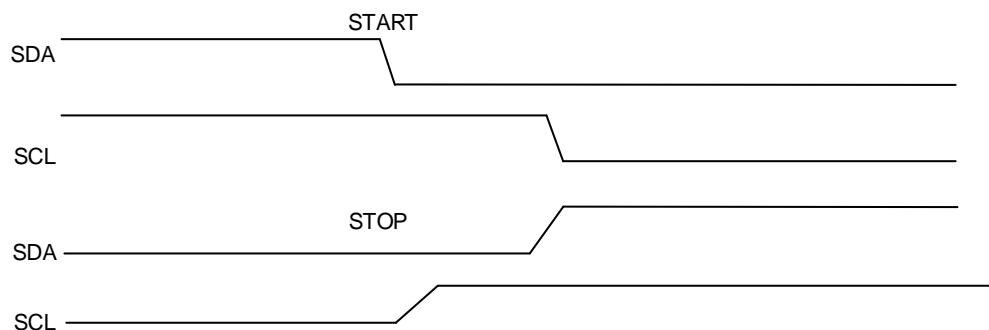
**Figure 17-2. Data validation**



### 17.3.3. START and STOP condition

All transactions begin with a START (S) and are terminated by a STOP (P) (see [Figure 17-3. START and STOP condition](#)). 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 17-3. START and STOP condition**

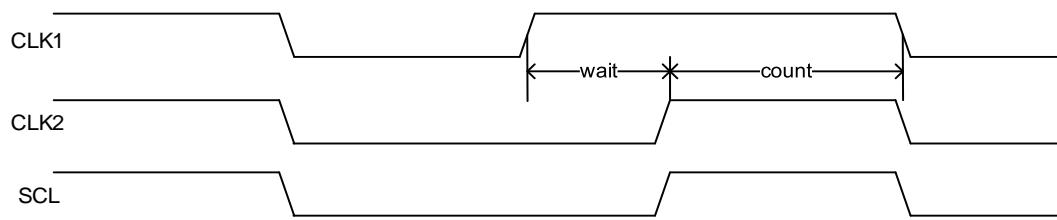


### 17.3.4. Clock synchronization

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 17-4. 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 17-4. Clock synchronization**



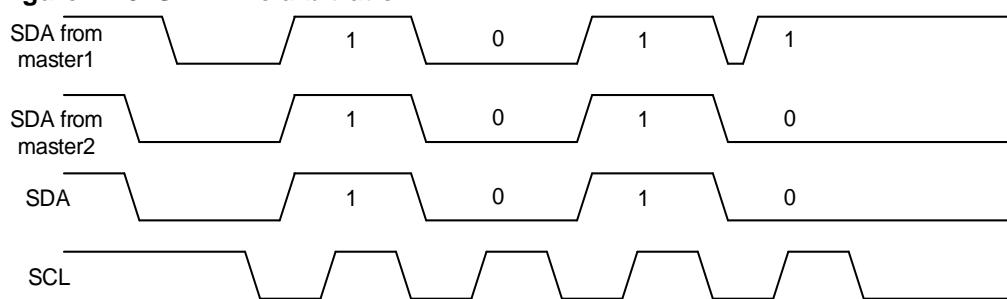
### 17.3.5. Arbitration

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 17-5. SDA Line arbitration**



### 17.3.6. I2C communication flow

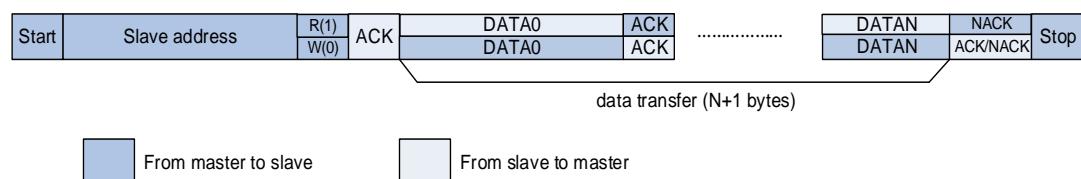
Each I2C device is recognized by a unique address (whether it is a microcontroller, LCD driver,

memory or keyboard interface) and can operate as either a transmitter or receiver, depending on the function of the device.

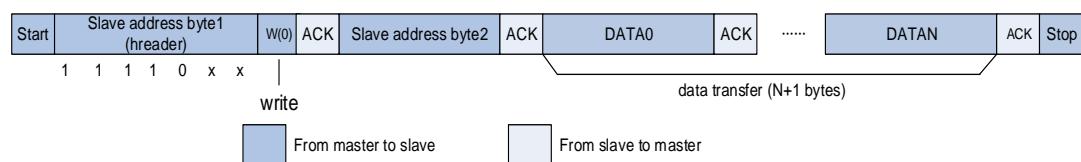
An I2C slave will continue to detect addresses after a START condition on I2C bus and compare the detected address with its slave address which is programmable by software. Once the two addresses match, the I2C slave will send an ACK to the I2C bus and responses to the following command on I2C bus: transmitting 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.

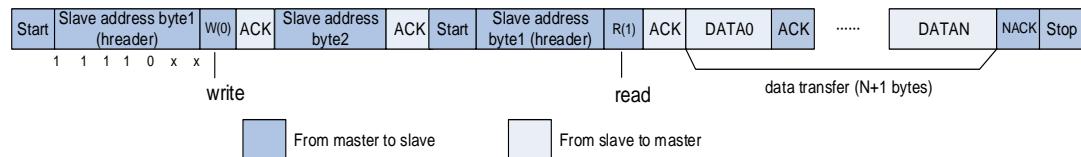
**Figure 17-6. I2C communication flow with 7-bit address**



**Figure 17-7. I2C communication flow with 10-bit address (Master Transmit)**



**Figure 17-8. I2C communication flow with 10-bit address (Master Receive)**



### 17.3.7. 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 signal to permit that transfer. At that time, any device addressed is considered as 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.

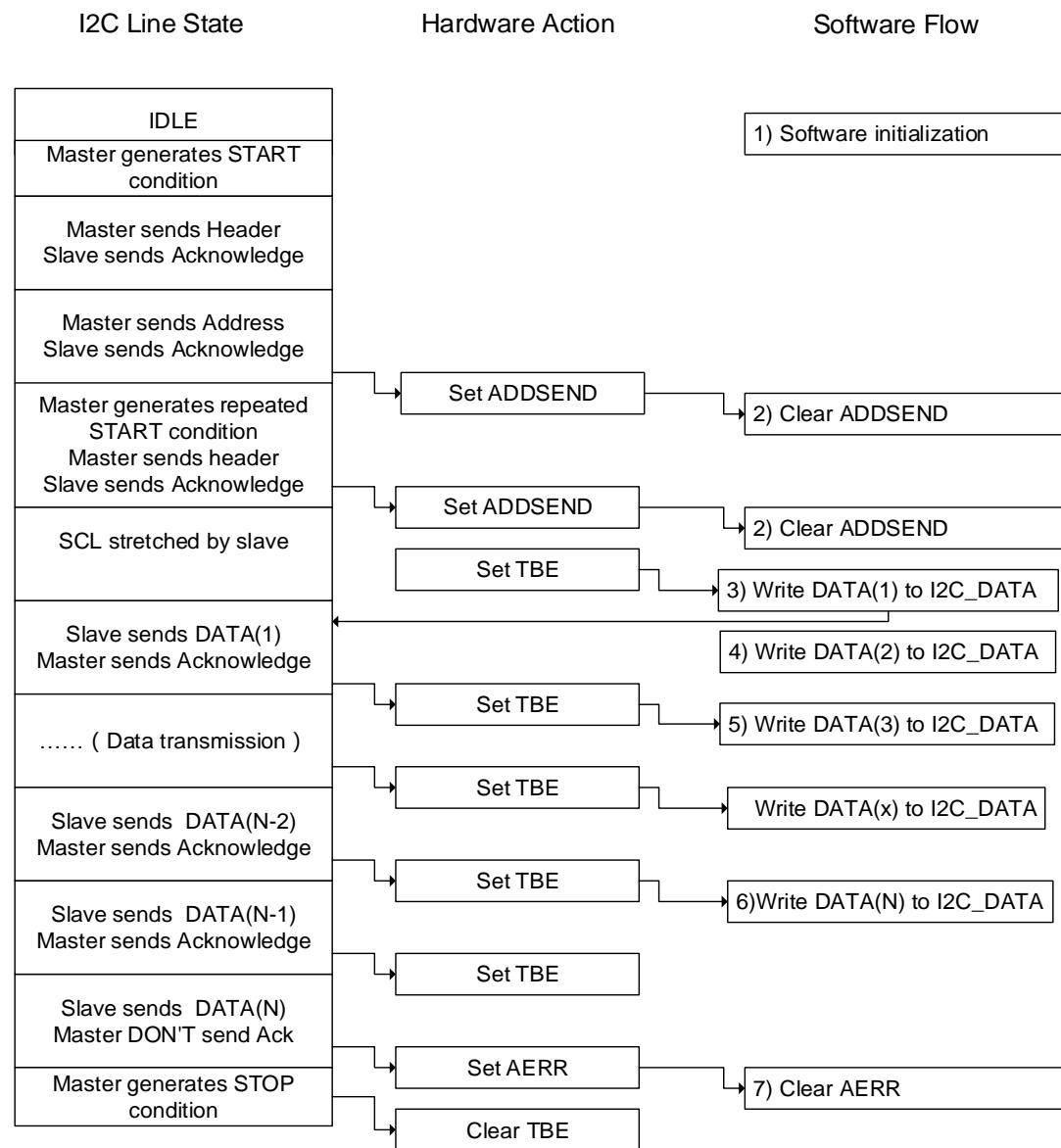
I2C block supports all of the four I2C modes. After system reset, it works in slave mode. If it's programmed by software and finished sending a START condition on I2C bus, it changes into master mode. The I2C changes back to slave mode after it's programmed by software and finished sending a STOP condition on I2C bus.

### Programming model in slave transmitting mode

As is shown in [Figure 17-9. Programming model for slave transmitting\(10-bit address mode\)](#), 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 17-9. Programming model for slave transmitting(10-bit address mode)



### Programming model in slave receiving mode

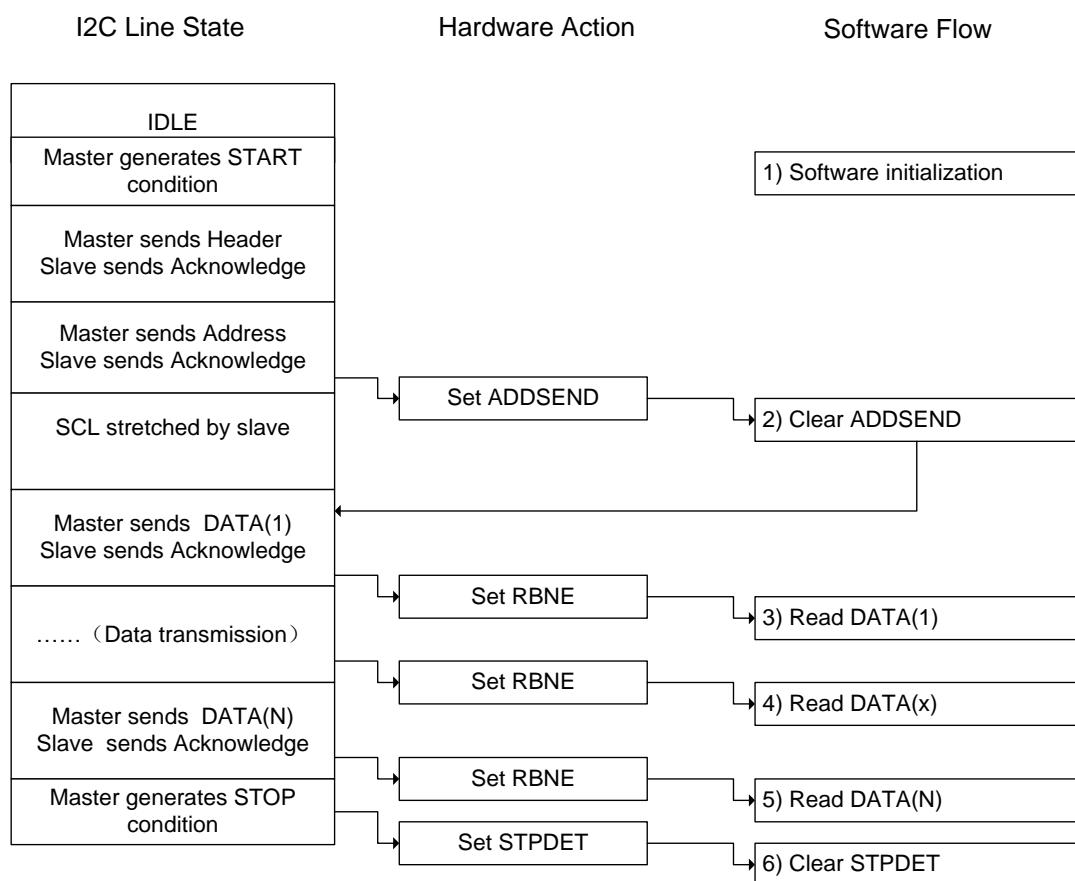
As is shown in [Figure 17-10. Programming model for slave receiving\(10-bit address mode\)](#), the following software procedure should be followed if users wish to make reception in slave receiver mode:

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 17-10. Programming model for slave receiving(10-bit address mode)**

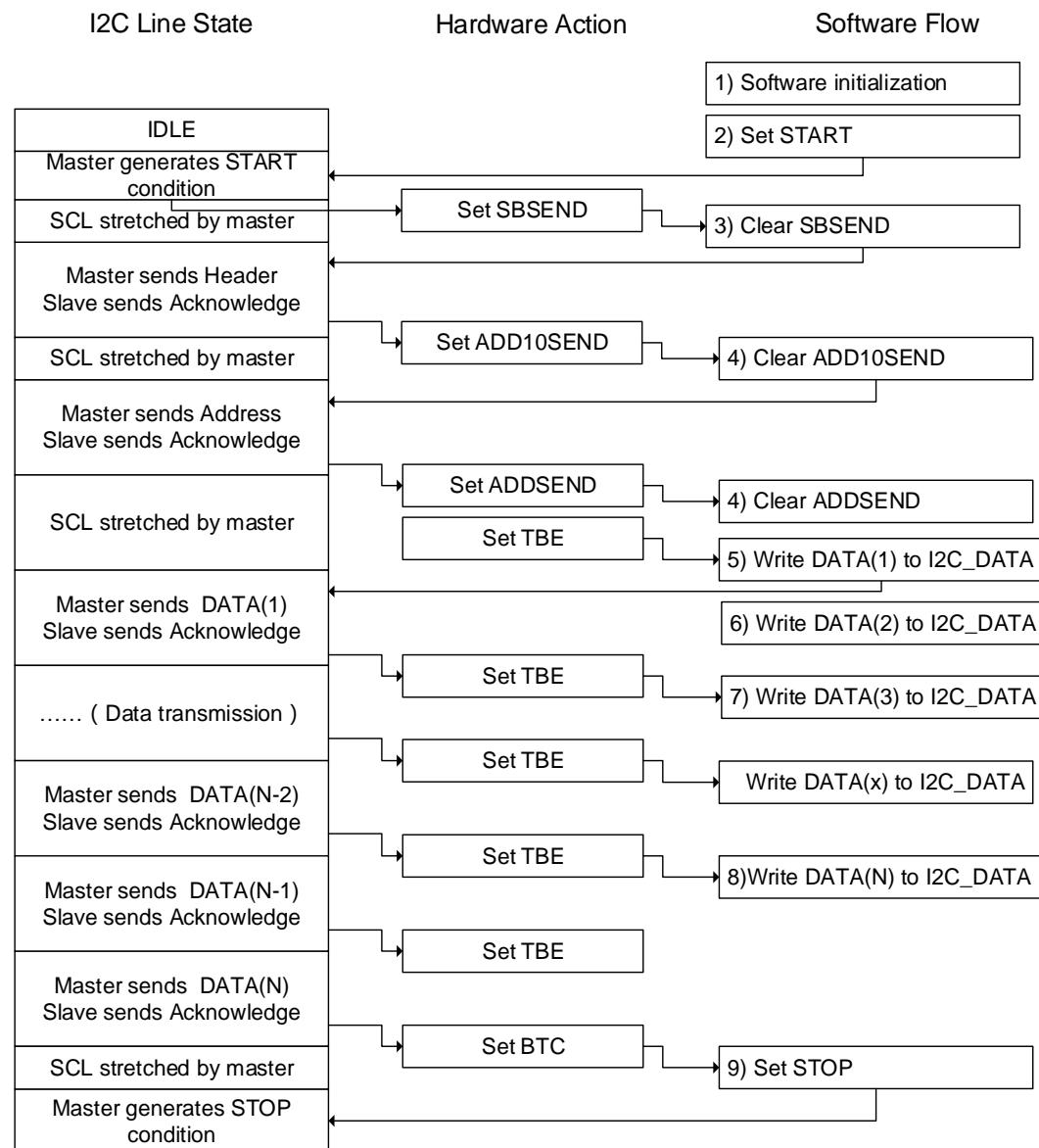


### Programming model in master transmitting mode

As it shows in [\*\*Figure 17-11. Programming model for master transmitting\(10-bit address mode\)\*\*](#), 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 ADD10S END 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 17-11. Programming model for master transmitting(10-bit address mode)**


### **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.

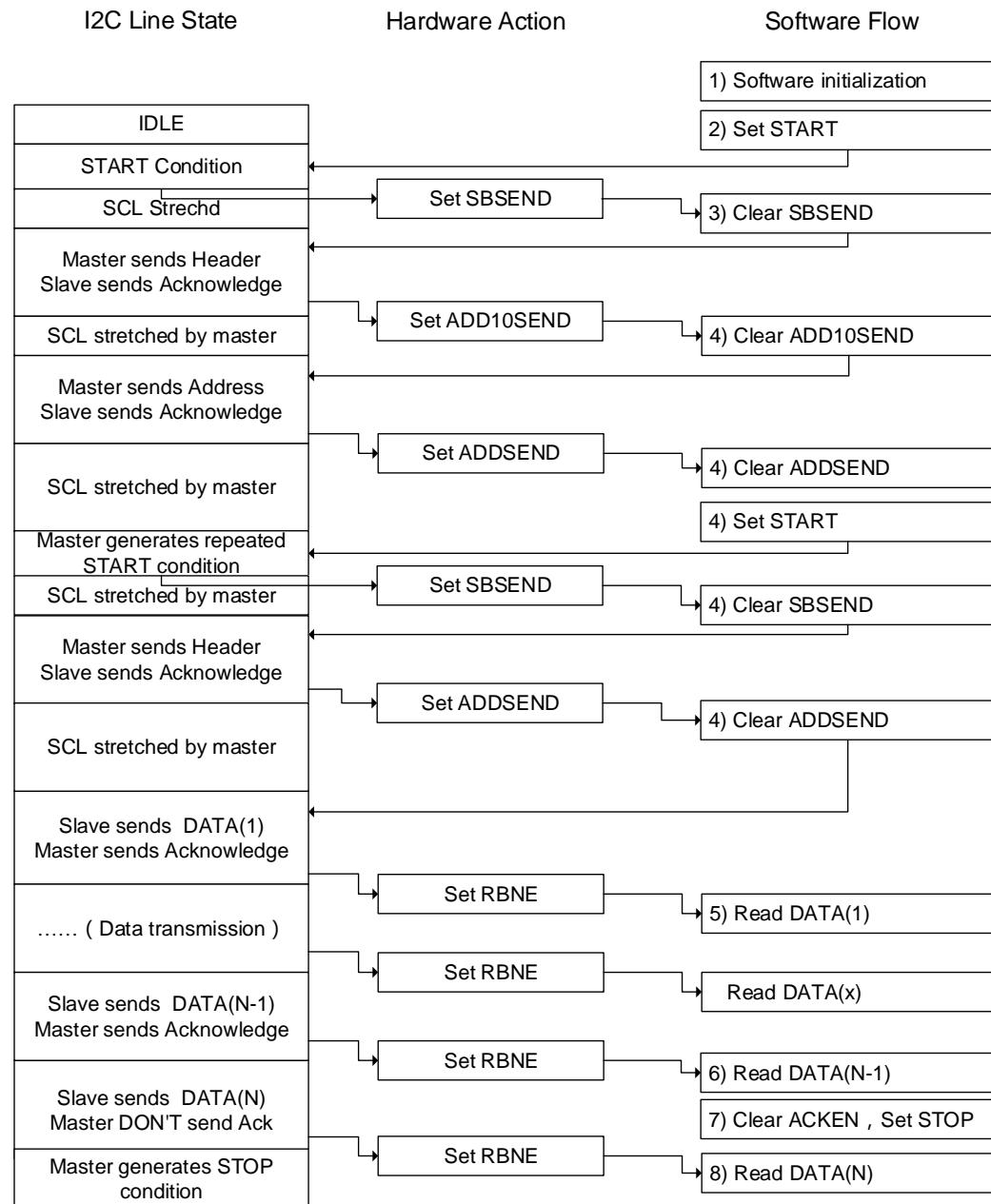
#### **Solution A**

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 17-12. Programming model for master receiving using Solution A(10-bit address mode)**



### **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 17-13. Programming model for master receiving using solution B\(10-bit address mode\)](#), 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:

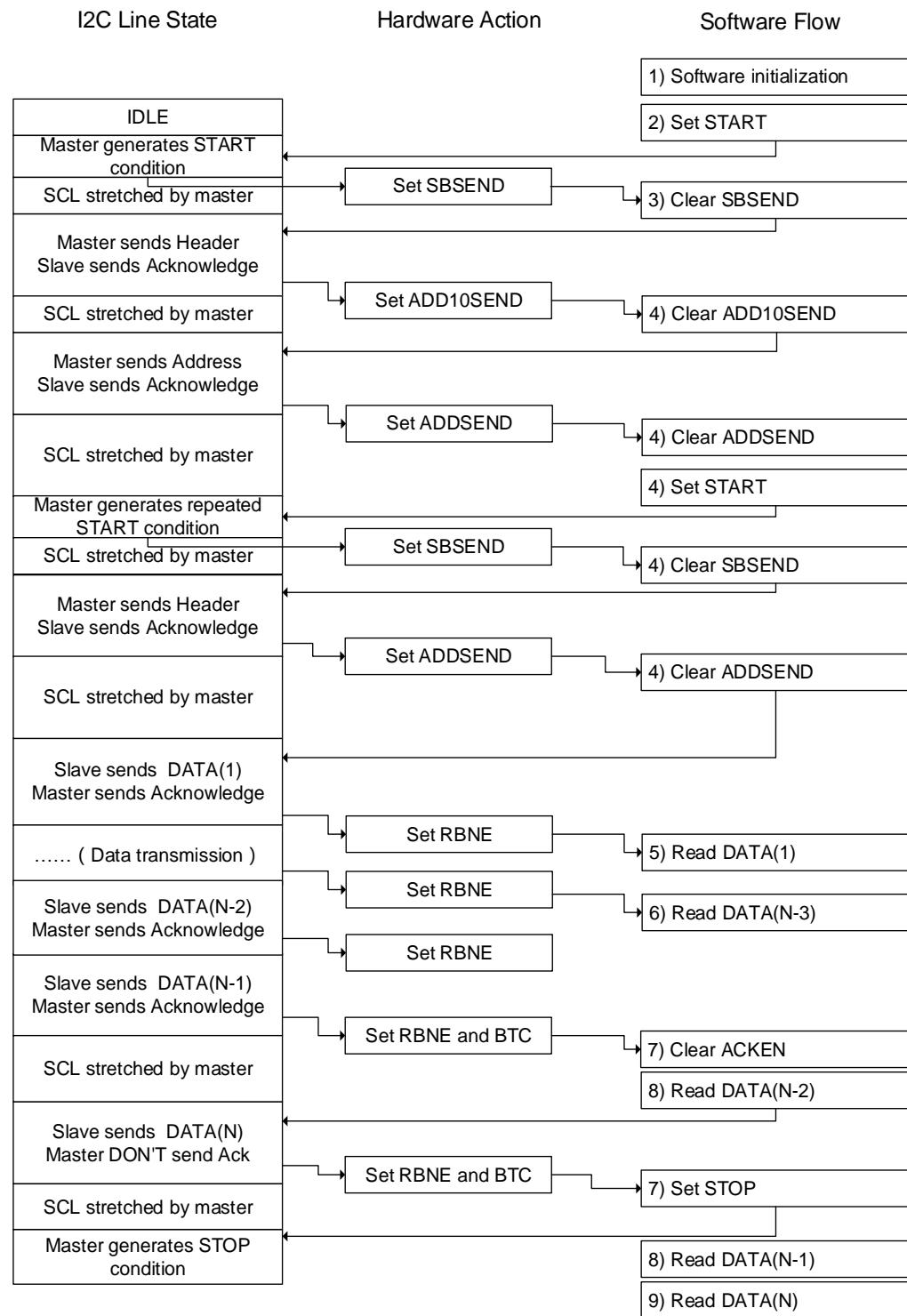
#### N=1

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 17-13. Programming model for master receiving using solution B(10-bit address mode)**



### 17.3.8. 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 SS 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.

### 17.3.9. 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 after 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.

### 17.3.10. 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 and PECTRANS bit is set.

### 17.3.11. 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 (SMBSEL =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.

#### 17.3.12. 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).

**Table17-2. 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

**Table17-3. 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
SMBTO	Bus timeout in SMBus mode
SMBALT	SMBus Alert

## 17.4. Register definition

I2C0 base address: 0x4000 5400

I2C1 base address: 0x4000 5800

### 17.4.1. 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	PECTRA NS	POAP	ACKEN	STOP	START	SS	GCEN	PECEN	ARPEN	SMBSEL	Reserved	SMBEN	I2CEN

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	Generate a STOP condition on I2C bus  This bit is set and cleared by software and set by hardware when SMBUs timeout and cleared by hardware when STOP condition detected. 0: STOP will not be sent 1: STOP will be sent
8	START	Generate a START condition on I2C bus  This bit is set and cleared by software and cleared by hardware when START condition detected or I2CEN=0 0: START will not be sent 1: START will be sent
7	SS	Whether to stretch SCL low when data is not ready in slave mode.  This bit is set and cleared by software. 0: SCL Stretching is enabled 1: SCL Stretching is disabled
6	GCEN	Whether or not to response to a General Call (0x00)  0: Slave won't response to a General Call 1: Slave will response to a General Call
5	PECEN	PEC Calculation Switch  0: PEC Calculation off 1: PEC Calculation on
4	ARPEN	ARP protocol in SMBus switch  0: ARP is disabled 1: ARP is enabled
3	SMBSEL	SMBusType Selection  0: Device 1: Host
2	Reserved	Must keep the reset value.
1	SMBEN	SMBus/I2C mode switch  0: I2C mode 1: SMBus mode
0	I2CEN	I2C peripheral enable  0: I2C is disabled 1: I2C is enabled

### 17.4.2. 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

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 SBSEND, 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 APB1 clock in MHz which is at least 2. 000000 - 000001: Not allowed 000010 - 110110: 2 MHz~54MHz 110111 - 111111: Not allowed due to the limitation of APB1 clock Note: In I2C standard mode, the frequencies of APB1 must be equal or greater than 2MHz. In I2C fast mode, the frequencies of APB1 must be equal or greater than

8MHz.

#### 17.4.3. 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[0]				

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

#### 17.4.4. 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			

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
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	DUADDEN	Dual-Address mode switch 0: Dual-Address mode disabled 1: Dual-Address mode enabled

### 17.4.5. 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

### 17.4.6. 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 carelessly

		ACKEN bit.
11	OUERR	<p>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.</p> <p>This bit is set by hardware and cleared by writing 0.</p> <p>0: No over-run or under-run occurs 1: Over-run or under-run occurs</p>
10	AERR	<p>Acknowledge Error</p> <p>This bit is set by hardware and cleared by writing 0.</p> <p>0: No Acknowledge Error 1: Acknowledge Error</p>
9	LOSTARB	<p>Arbitration Lost in master mode</p> <p>This bit is set by hardware and cleared by writing 0.</p> <p>0: No Arbitration Lost 1: Arbitration Lost occurs and the I2C block changes back to slave mode.</p>
8	BERR	<p>A bus error occurs indication an unexpected START or STOP condition on I2C bus</p> <p>This bit is set by hardware and cleared by writing 0.</p> <p>0: No bus error 1: A bus error detected</p>
7	TBE	<p>I2C_DATA is Empty during transmitting</p> <p>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).</p> <p>0: I2C_DATA is not empty 1: I2C_DATA is empty, software can write</p>
6	RBNE	<p>I2C_DATA is not Empty during receiving</p> <p>This bit is set by hardware after it moves a byte from shift register to I2C_DATA and cleared by reading I2C_DATA. 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.</p> <p>0: I2C_DATA is empty 1: I2C_DATA is not empty, software can read</p>
5	Reserved	Must be kept the reset value.
4	STPDET	<p>STOP condition detected in slave mode</p> <p>This bit is set by hardware and cleared by reading I2C_STAT0 and then writing I2C_CTL0</p> <p>0: STOP condition not detected in slave mode</p>

		1: STOP condition detected in slave mode
3	ADD10SEND	<p>Header of 10-bit address is sent in master mode</p> <p>This bit is set by hardware and cleared by reading I2C_STAT0 and writing I2C_DATA.</p> <p>0: No header of 10-bit address sent in master mode</p> <p>1: Header of 10-bit address is sent in master mode</p>
2	BTC	<p>Byte transmission completed.</p> <p>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.</p> <p>This bit is set by hardware.</p> <p>This bit can be cleared by 3 ways as follow:</p> <ol style="list-style-type: none"> <li>1. Reading I2C_STAT0 followed by reading or writing I2C_DATA</li> <li>2. Hardware clearing: sending the STOP condition or START condition</li> <li>3. Bit 0 (I2CEN bit) of the I2C_CTL0 is reset.</li> </ol> <p>0: BTC not asserted</p> <p>1: BTC asserted</p>
1	ADDSEND	<p>Address is sent in master mode or received and matches in slave mode.</p> <p>This bit is set by hardware and cleared by reading I2C_STAT0 and reading I2C_STAT1.</p> <p>0: No address sent or received</p> <p>1: Address sent out in master mode or a matched address is received in slave mode</p>
0	SBSEND	<p>START condition sent out in master mode</p> <p>This bit is set by hardware and cleared by reading I2C_STAT0 and writing I2C_DATA</p> <p>0: No START condition sent</p> <p>1: START condition sent</p>

#### 17.4.7. 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
					PECV[7:0]			DUMODF	HSTSMB	DEFSMB	RXGC	Reserved	TR	I2CBSY	MASTER

Bits	Fields	Descriptions
15:8	PECV[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	TR	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 set by hardware when a START condition generates. 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

#### 17.4.8. 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)

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

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]	I2C Clock control in master mode In standard speed mode: $T_{high} = T_{low} = CLKC * T_{PCLK1}$ In fast speed mode, if DTCY=0: $T_{high} = CLKC * T_{PCLK1}$ , $T_{low} = 2 * CLKC * T_{PCLK1}$ In fast speed mode, if DTCY=1: $T_{high} = 9 * CLKC * T_{PCLK1}$ , $T_{low} = 16 * CLKC * T_{PCLK1}$

#### 17.4.9. 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)

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

Bits	Fields	Descriptions
15:6	Reserved	Must be kept the reset value.
5:0	RISETIME[5:0]	Maximum rise time in master mode The RISETIME value should be the maximum SCL rise time incremented by 1.

## 18. Serial peripheral interface/Inter-IC sound (SPI/I2S)

### 18.1. 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.

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.

### 18.2. Characteristics

#### 18.2.1. SPI characteristics

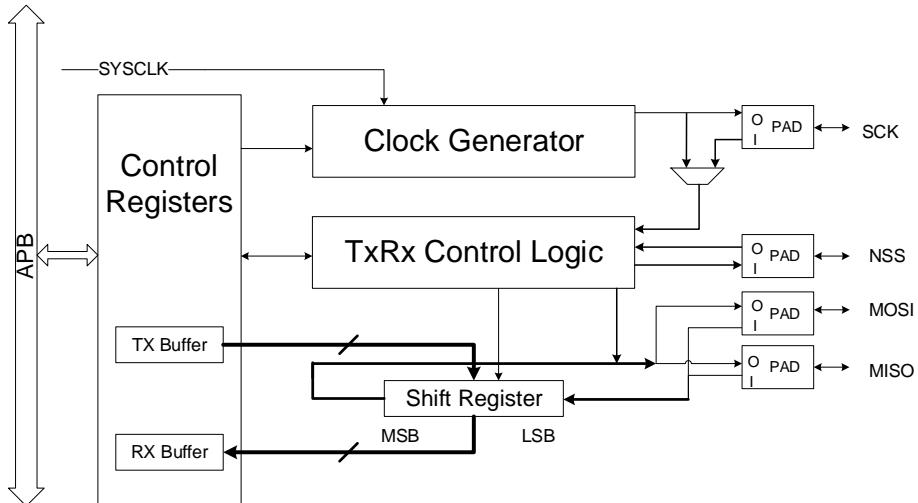
- Master or slave operation with full-duplex or simplex mode.
- Separate transmit and receive buffer, 16 bits wide.
- Data frame size can be 8 or 16 bits.
- Bit order can be LSB or MSB.
- Software and hardware NSS management.
- Hardware CRC calculation, transmission and checking.
- Transmission and reception using DMA.
- SPI TI mode supported.
- SPI NSS pulse mode supported.

#### 18.2.2. I2S characteristics

- Master or slave operation for transmission/reception.
- 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.

## 18.3. SPI block diagram

Figure 18-1. Block diagram of SPI



## 18.4. SPI signal description

### 18.4.1. Normal configuration

Table 18-1. SPI signal description

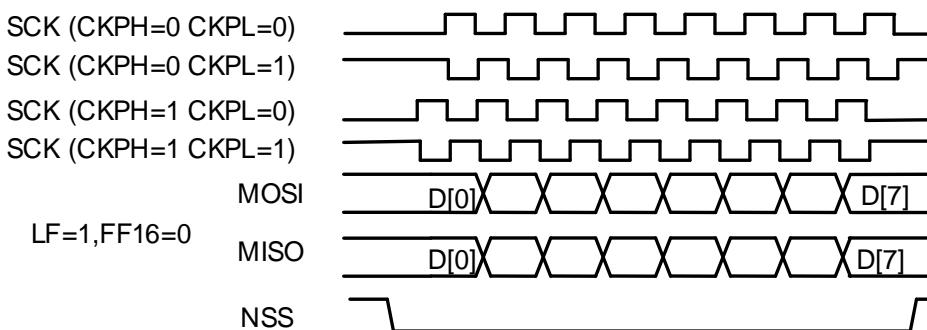
Pin name	Direction	Description
SCK	I/O	Master: SPI clock output Slave: SPI clock input
MISO	I/O	Master: Data reception line Slave: Data transmission line Master with bidirectional mode: Not used Slave with bidirectional mode: Data transmission and reception line.
MOSI	I/O	Master: Data transmission line 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 for single master (NSSDRV=1) or for multi-master (NSSDRV=0) application. Slave in hardware NSS mode: NSS input, as a chip select signal for slave.

## 18.5. SPI function overview

### 18.5.1. 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. These bits take no effect in TI mode.

**Figure 18-2. SPI timing diagram in normal mode**



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.

Data order is configured by the LF bit in SPI\_CTL0 register, and SPI will first send the LSB if LF=1, or the MSB if LF=0. The data order is fixed to MSB first in TI mode.

### 18.5.2. NSS function

#### Slave mode

When slave mode is configured (MSTMOD=0), SPI gets NSS level from NSS pin in hardware NSS mode (SWNSSEN = 0) or from SWNSS bit in software NSS mode (SWNSSEN = 1), and SPI 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 (SWNSSEN=0, NSSDRV=0) or software mode (SWNSSEN=1). Then, once the NSS pin (in hardware NSS mode) or the SWNSS bit (in software NSS mode) goes low, the SPI automatically enters 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 (SWNSSEN=0, NSSDRV=1). NSS stays high after SPI is enabled and goes low when transmission or reception process begins. When SPI is disabled, the NSS goes high.

The application may also use a general purpose IO as NSS pin to realize more flexible NSS.

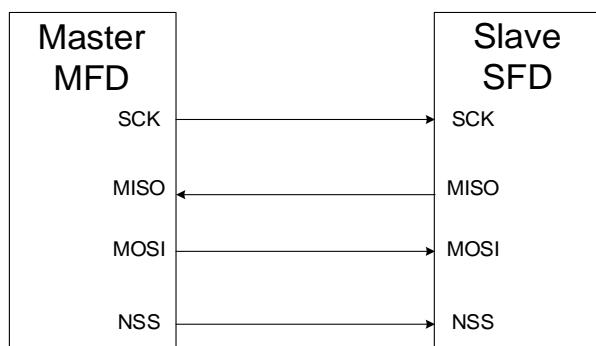
### 18.5.3. SPI operating modes

**Table 18-2. SPI operating modes**

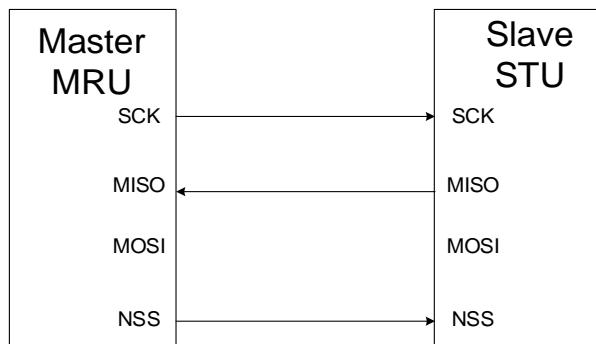
Mode	Description	Register configuration	Data pin usage
MFD	Master full-duplex	MSTMOD = 1 RO = 0 BDEN = 0 BDOEN: Don't care	MOSI: Transmission MISO: Reception
MTU	Master transmission with unidirectional connection	MSTMOD = 1 RO = 0 BDEN = 0 BDOEN: Don't care	MOSI: Transmission MISO: Not used
MRU	Master 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 BDEN = 1 BDOEN = 1	MOSI: Transmission MISO: Not used
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	MSTMOD = 0	MOSI: Not used

<b>Mode</b>	<b>Description</b>	<b>Register configuration</b>	<b>Data pin usage</b>
	bidirectional connection	RO = 0 BDEN = 1 BDOEN = 1	MISO: Transmission
SRB	Slave reception with bidirectional connection	MSTMOD = 0 RO = 0 BDEN = 1 BDOEN = 0	MOSI: Not used MISO: Reception

**Figure 18-3. A typical full-duplex connection**



**Figure 18-4. A typical simplex connection (Master: Receive, Slave: Transmit)**



**Figure 18-5. A typical simplex connection (Master: Transmit only, Slave: Receive)**

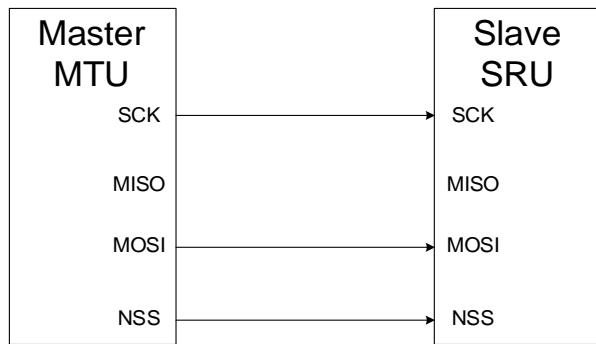
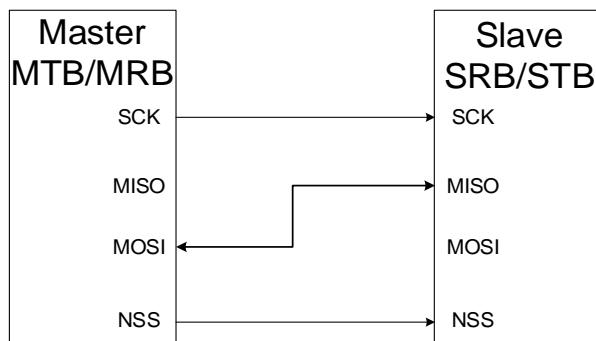


Figure 18-6. A typical bidirectional connection



### SPI initialization sequence

Before transmitting or receiving data, application should follow the SPI initialization sequence described below:

1. If master mode or slave TI mode is used, program the PSC [2:0] bits in SPI\_CTL0 register to generate SCK with desired baud rate, or configure the Td time in TI mode, 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 (SWNSSEN and NSSDRV bits in the SPI\_CTL0 register) according to the application's demand as described above in [NSS function](#) section.
6. If TI mode is used, set TMOD bit in SPI\_CTL1 register, otherwise, ignore this step.
7. Configure MSTMOD, RO, BDEN and BDOEN depending on the operating modes described in [SPI operating modes](#) section.
8. If Quad-SPI mode is used, set the QMOD bit in SPI\_QCTL register. Ignore this step if Quad-SPI mode is not used.
1. 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 at SCK pin begins to toggle 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

After the last valid sample clock, the incoming data will be moved from shift register to the receive buffer and 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 TI mode or NSSP mode)

In full-duplex mode, either MFD or SFD, the RBNE and TBE flags should be monitored and then follow the sequences described above.

The transmission mode (MTU, MTB, STU or STB) is similar to the transmission sequence of full-duplex mode regardless of the RBNE and OVRE bits.

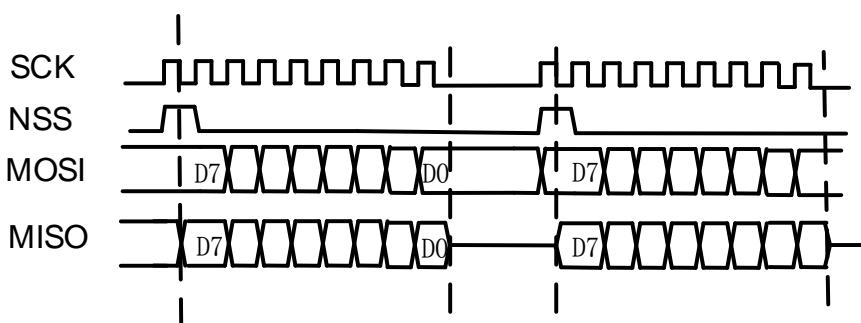
The master reception mode (MRU or MRB) is different from the reception sequence of full-duplex mode. In MRU or MRB mode, after SPI is enabled, the SPI continuously generates SCK until the SPI is disabled. So the application should ignore the TBE flag and read out reception buffer in time after the RBNE flag is set, otherwise a data overrun fault will occur.

The slave reception mode (SRU or SRB) is similar to the reception sequence of full-duplex mode regardless of the TBE flag.

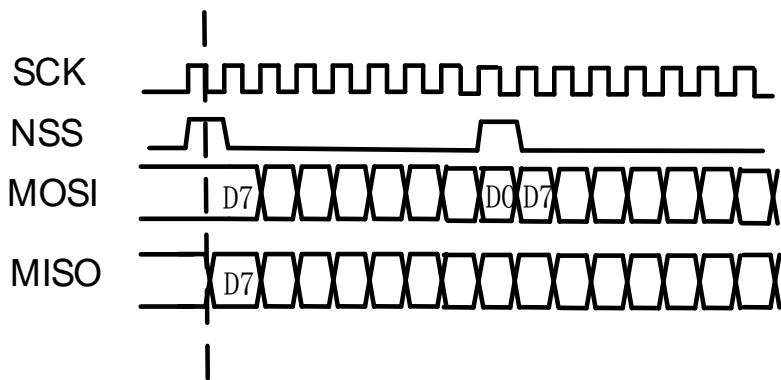
### SPI TI mode

SPI TI mode takes NSS as a special frame header flag signal and its operation sequence is similar to normal mode described above. The modes described above (MFD, MTU, MRU, MTB, MRB, SFD, STU, SRU, STB and SRB) are still supported in TI mode. While, in TI mode the CKPL and CKPH bits in SPI\_CTL0 registers take no effect and the SCK sample edge is falling edge.

**Figure 18-7. Timing diagram of TI master mode with discontinuous transfer**

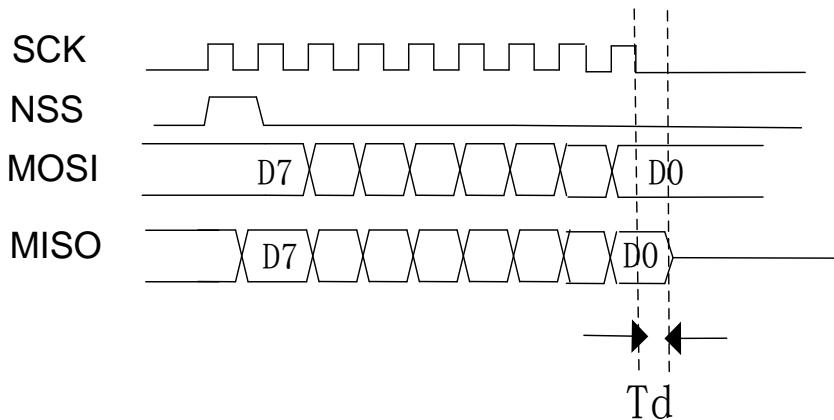


**Figure 18-8. Timing diagram of TI master mode with continuous transfer**



In master TI mode, SPI can perform continuous or non-continuous transfer. If the master writes SPI\_DATA register fast enough, the transfer is continuous, otherwise non-continuous. In non-continuous transfer, there is an extra header clock cycle before each byte. While in continuous transfer, the extra header clock cycle only exists before the first byte and the following bytes' header clock is overlaid at the last bit of previous bytes.

**Figure 18-9. Timing diagram of TI slave mode**



In slave TI mode, after the last rising edge of SCK in transfer, the slave begins to transmit the LSB bit of the last data byte, and after a half-bit time, the master begins to sample the line. To make sure that the master samples the right value, the slave should continue to drive this bit after the falling sample edge of SCK for a period of time before releasing the pin. This time is called  $T_d$ .  $T_d$  is decided by PSC[2:0] bits in SPI\_CTL0 register.

$$T_d = \frac{T_{bit}}{2} + 5 * T_{pclk} \quad (18-1)$$

For example, if PSC[2:0] = 010,  $T_d$  is  $9 * T_{pclk}$ .

In slave mode, the slave also monitors the NSS signal and sets an error flag FERR if it detects an incorrect NSS behavior, for example, toggles at the middle bit of a byte.

## NSS pulse mode operation sequence

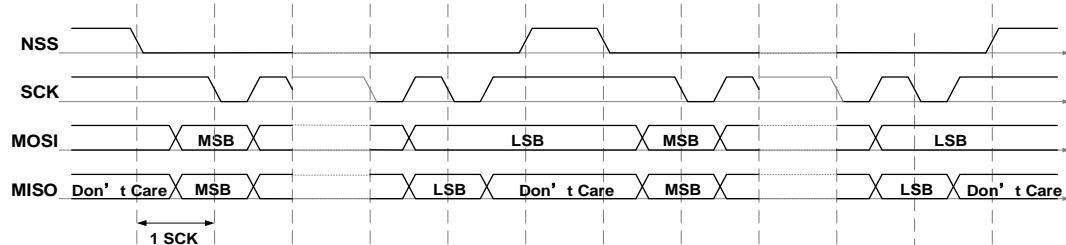
This function is controlled by NSSP bit in SPI\_CTL1 register. In order to implement this function, several additional conditions must be met: configure the device to master mode, frame format should follow the normal SPI protocol, select the first clock transition as the data capture edge.

In summary, MSTMOD = 1, NSSP = 1, CKPH = 0.

When NSS pulse mode is enabled

, a pulse duration of at least 1 SCK clock period is inserted between two successive data frames depending on the status of internal data transmit buffer. Multiple SCK clock cycle intervals are possible if the transfer buffer stays empty. This function is designed for single master-slave configuration for the slave to latch data. The following diagram depicts its timing diagram.

**Figure 18-10. Timing diagram of NSS pulse with continuous transmission**



## 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 ongoing transfer completes.

### TI mode

The disabling sequence of TI mode is the same as the sequences described above.

### NSS pulse mode

The disabling sequence of NSSP mode is the same as the sequences described above.

## 18.5.4. DMA function

The DMA 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 when 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 when RBNE=1, then DMA will acknowledge to this request and read data from the SPI\_DATA register automatically.

## 18.5.5. CRC function

There are two CRC calculators in SPI: one for transmission and the other for reception. The CRC calculation uses the polynomial defined in SPI\_CRCPOLY register.

Application can enable 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 registers.

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), when the SPI transmits a CRC and prepares to check the received CRC value, the SPI treats the incoming data as a 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 configure CRCNT bit and hardware will automatically process the CRC transmitting and checking.

## 18.6. SPI interrupts

### 18.6.1. 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 ongoing flag (TRANS)

TRANS is a status flag to indicate whether the transfer is ongoing or not. It is set and cleared by hardware and not controlled by software. This flag doesn't generate any interrupt.

### 18.6.2. Error flags

- Configuration fault error (CONFERR)

CONFERR is an error flag in master mode. In NSS hardware mode and if 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 bits are write protected 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.

- Format error (FERR)

In slave TI mode, the slave also monitors the NSS signal and set an error flag if it detects an incorrect NSS behavior, for example: toggles at the middle bit of a byte.

- 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.

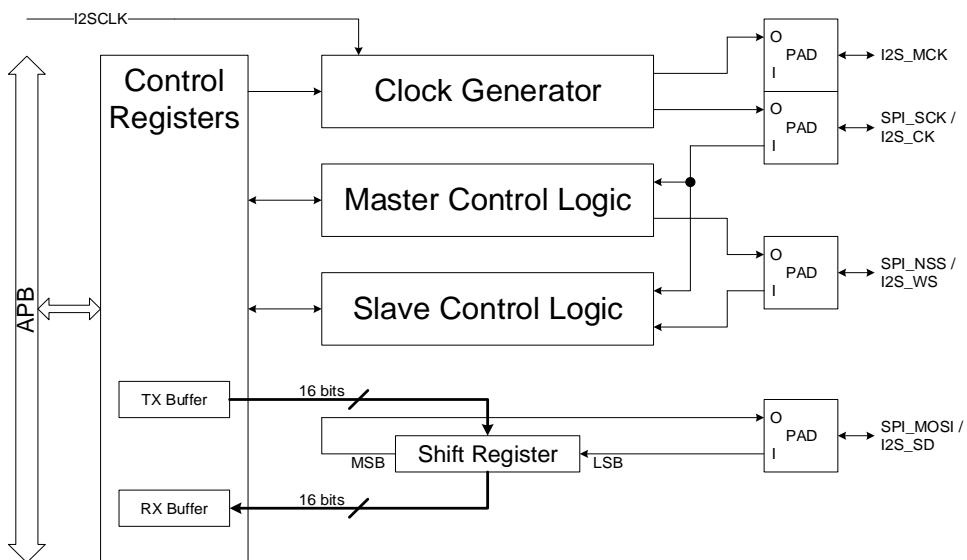
**Table 18-3. SPI interrupt requests**

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

Flag	Description	Clear method	Interrupt enable bit
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	
FERR	TI mode format error	Write 0 to FERR bit	

## 18.7. I2S block diagram

**Figure 18-11. 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 I2S\_CK and I2S\_WS. The shift register handles the serial data transmission and reception on I2S\_SD.

## 18.8. 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. It produces a frequency rate equals to  $256 \times F_s$ , and  $F_s$  is the audio sampling frequency.

## 18.9. I2S function overview

### 18.9.1. 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.

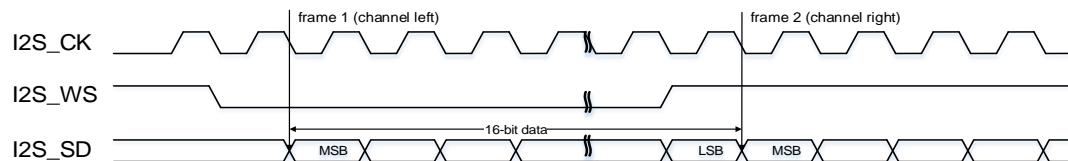
The data length and the channel length are configured by the DTLEN bits and CHLEN bit in the SPI\_I2SCTL register. Since the channel length must be greater than or equal to the data length, four packet types are available. They are 16-bit data packed in 16-bit frame, 16-bit data packed in 32-bit frame, 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 the transmission of 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 the transmission of 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.

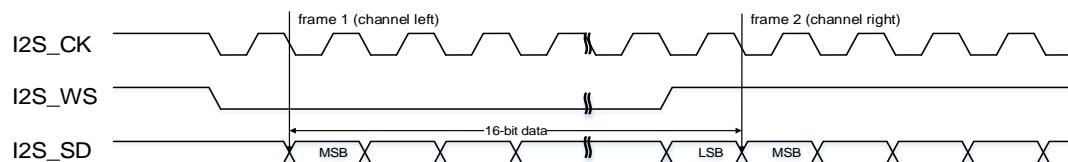
#### I2S Phillips standard

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 18-12. I2S Phillips standard timing diagram (DTLEN=00, CHLEN=0, CKPL=0)**



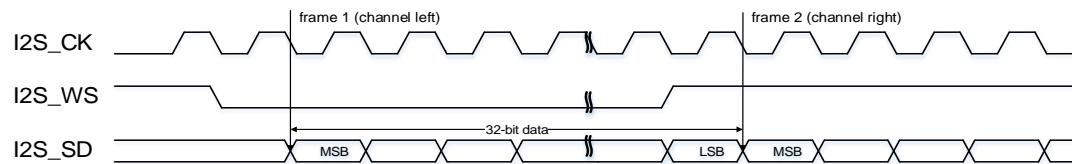
**Figure 18-13. I2S Phillips standard timing diagram (DTLEN=00, CHLEN=0, CKPL=1)**



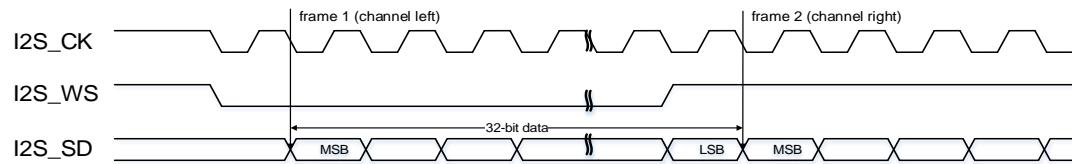
When the packet type is 16-bit data packed in 16-bit frame, only one write or read operation

to or from the SPI\_DATA register is needed to complete the transmission of a frame.

**Figure 18-14. I2S Phillips standard timing diagram (DTLEN=10, CHLEN=1, CKPL=0)**

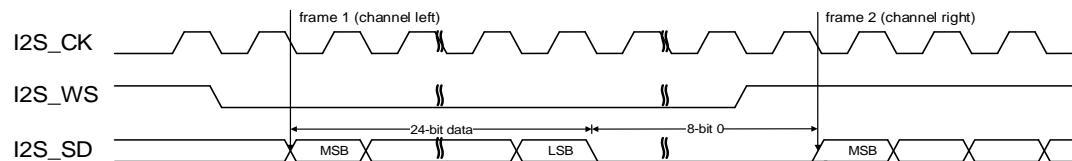


**Figure 18-15. I2S Phillips standard timing diagram (DTLEN=10, CHLEN=1, CKPL=1)**

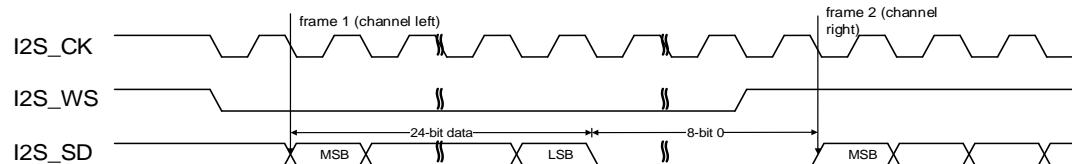


When the packet type is 32-bit data packed in 32-bit frame, two write or read operations to or from the SPI\_DATA register are needed to complete the transmission of a frame. In transmission mode, if a 32-bit data is going to be sent, the first data written to the SPI\_DATA register should be the higher 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 the higher 16 bits, and the second one should be the lower 16 bits.

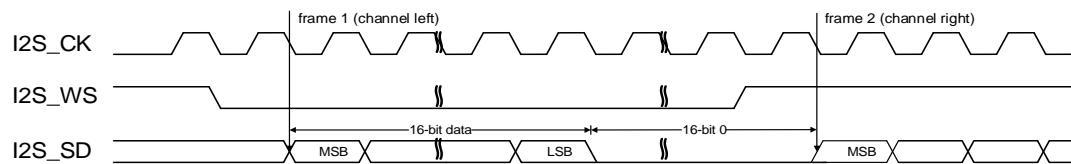
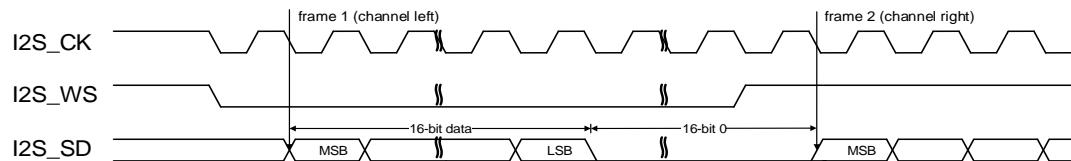
**Figure 18-16. I2S Phillips standard timing diagram (DTLEN=01, CHLEN=1, CKPL=0)**



**Figure 18-17. I2S Phillips standard timing diagram (DTLEN=01, CHLEN=1, CKPL=1)**



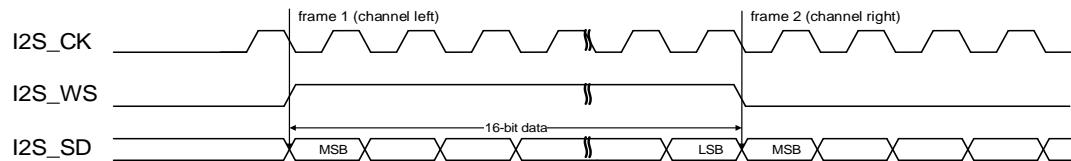
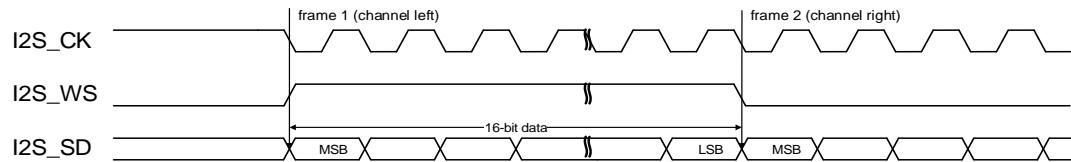
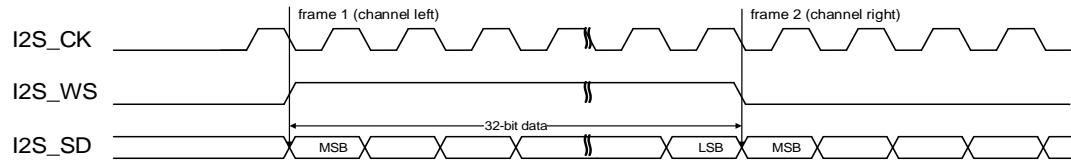
When the packet type is 24-bit data packed in 32-bit frame, two write or read operations to or from the SPI\_DATA register are needed to complete the transmission of a frame. In transmission mode, if a 24-bit data D[23:0] is going to be sent, the first data written to the SPI\_DATA register should be 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.

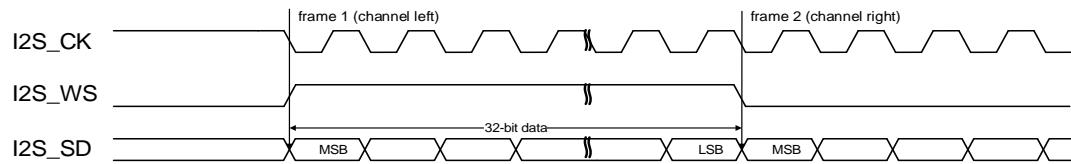
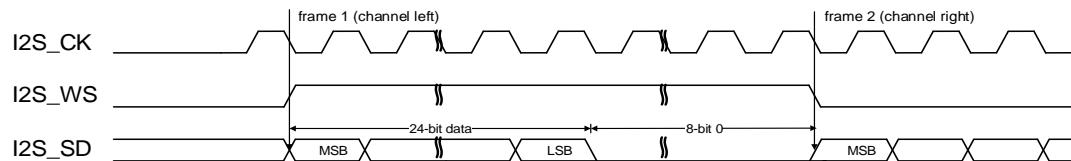
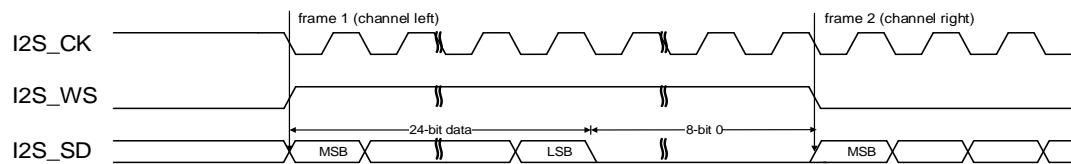
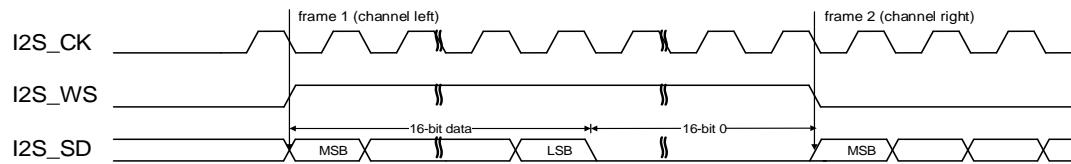
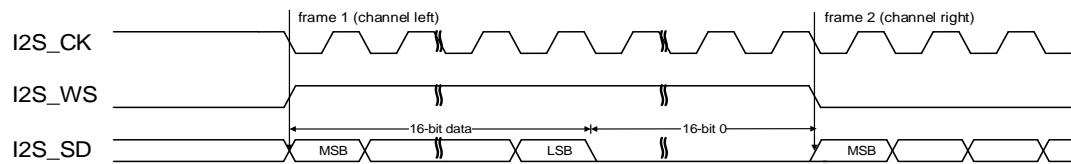
**Figure 18-18. I2S Phillips standard timing diagram (DTLEN=00, CHLEN=1, CKPL=0)**

**Figure 18-19. 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 the transmission of a frame. The remaining 16 bits are forced by hardware to 0x0000 to extend the data to 32-bit format.

### MSB justified standard

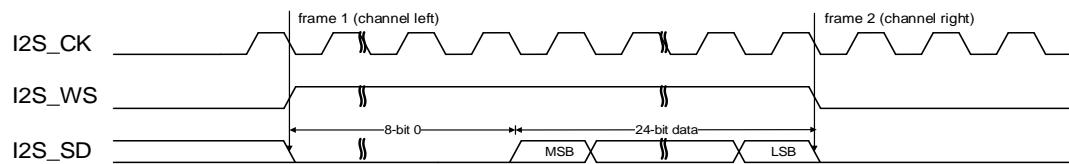
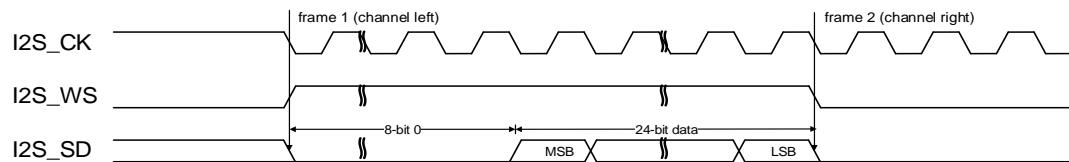
For MSB justified standard, I2S\_WS and I2S\_SD are updated on the falling edge of I2S\_CK. The SPI\_DATA register is handled in the exactly same way as that for I2S Phillips standard. The timing diagrams for each configuration are shown below.

**Figure 18-20. MSB justified standard timing diagram (DTLEN=00, CHLEN=0, CKPL=0)**

**Figure 18-21. MSB justified standard timing diagram (DTLEN=00, CHLEN=0, CKPL=1)**

**Figure 18-22. MSB justified standard timing diagram (DTLEN=10, CHLEN=1, CKPL=0)**


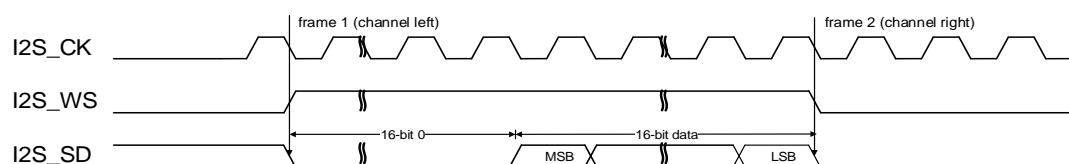
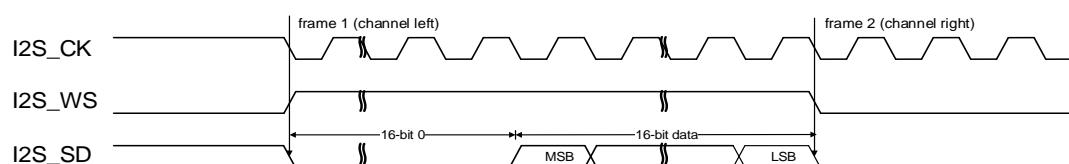
**Figure 18-23. MSB justified standard timing diagram (DTLEN=10, CHLEN=1, CKPL=1)**

**Figure 18-24. MSB justified standard timing diagram (DTLEN=01, CHLEN=1, CKPL=0)**

**Figure 18-25. MSB justified standard timing diagram (DTLEN=01, CHLEN=1, CKPL=1)**

**Figure 18-26. MSB justified standard timing diagram (DTLEN=00, CHLEN=1, CKPL=0)**

**Figure 18-27. 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 18-28. LSB justified standard timing diagram (DTLEN=01, CHLEN=1, CKPL=0)**

**Figure 18-29. 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 the transmission of 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 18-30. LSB justified standard timing diagram (DTLEN=00, CHLEN=1, CKPL=0)**

**Figure 18-31. LSB justified standard timing diagram (DTLEN=00, CHLEN=1, CKPL=1)**


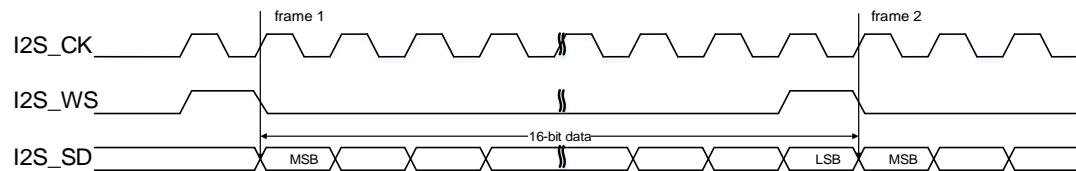
When the packet type is 16-bit data packed in 32-bit frame, only one write or read operation to or from the SPI\_DATA register is needed to complete the transmission of a frame. The remaining 16 bits are forced by hardware to 0x0000 to extend the data to 32-bit format.

### **PCM standard**

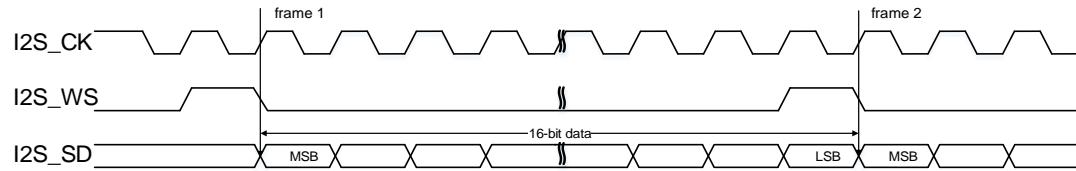
For PCM standard, I2S\_WS and I2S\_SD are updated on the rising edge of I2S\_CK, and the I2S\_WS signal indicates frame synchronization information. Both the short frame synchronization mode and the long frame synchronization mode are available and

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.

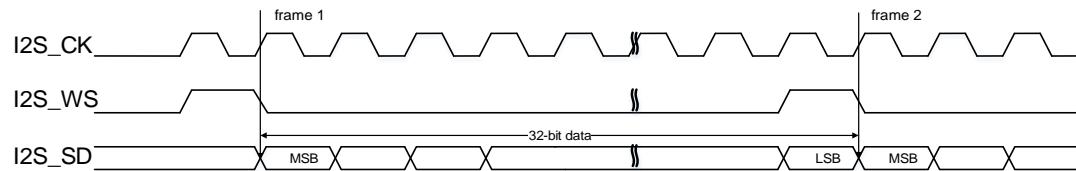
**Figure 18-32. PCM standard short frame synchronization mode timing diagram (DTLEN=00, CHLEN=0, CKPL=0)**



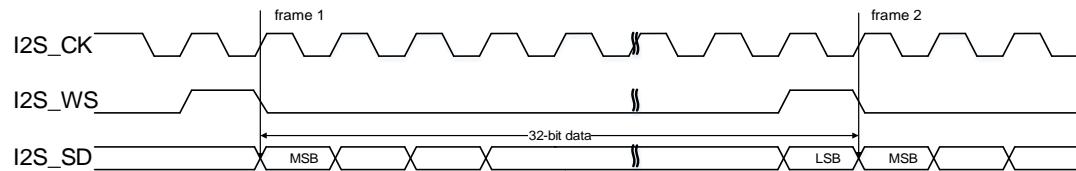
**Figure 18-33. PCM standard short frame synchronization mode timing diagram (DTLEN=00, CHLEN=0, CKPL=1)**



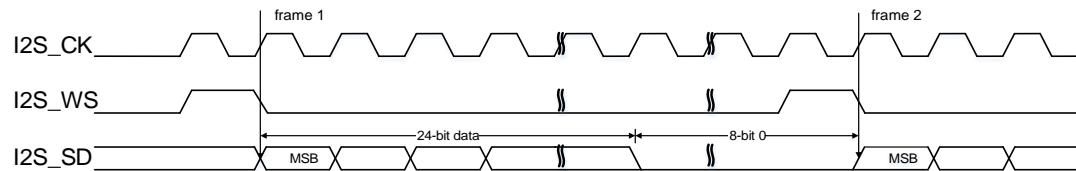
**Figure 18-34. PCM standard short frame synchronization mode timing diagram (DTLEN=10, CHLEN=1, CKPL=0)**



**Figure 18-35. PCM standard short frame synchronization mode timing diagram (DTLEN=10, CHLEN=1, CKPL=1)**

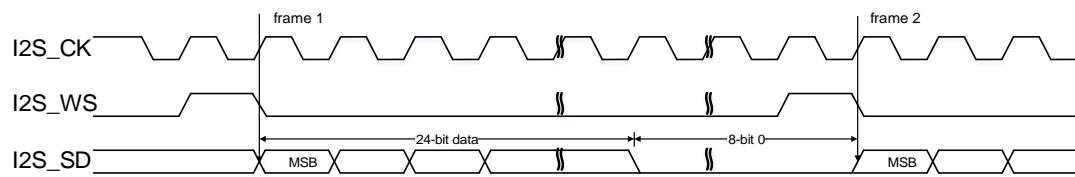


**Figure 18-36. PCM standard short frame synchronization mode timing diagram (DTLEN=01, CHLEN=1, CKPL=0)**

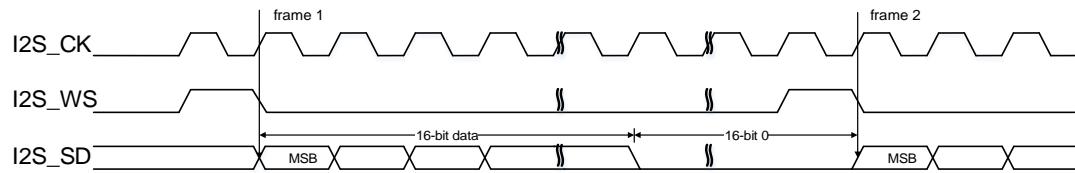


**Figure 18-37. PCM standard short frame synchronization mode timing diagram**

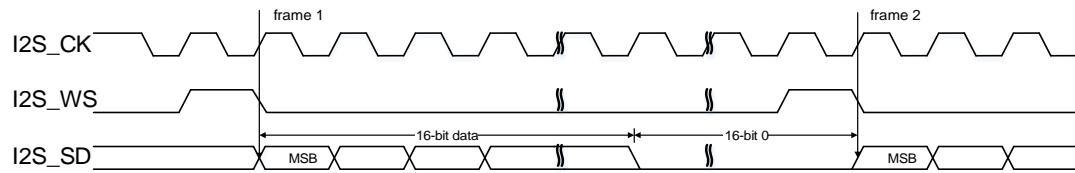
(DTLEN=01, CHLEN=1, CKPL=1)



**Figure 18-38. PCM standard short frame synchronization mode timing diagram (DTLEN=00, CHLEN=1, CKPL=0)**

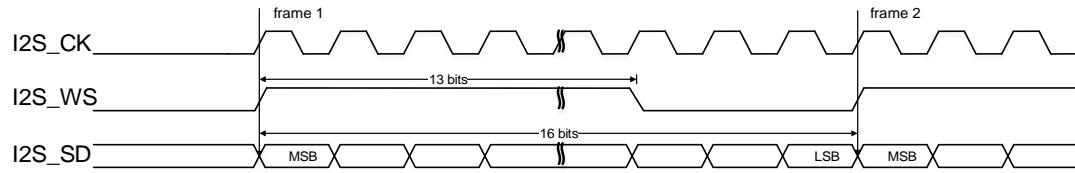


**Figure 18-39. PCM standard short frame synchronization mode timing diagram (DTLEN=00, CHLEN=1, CKPL=1)**

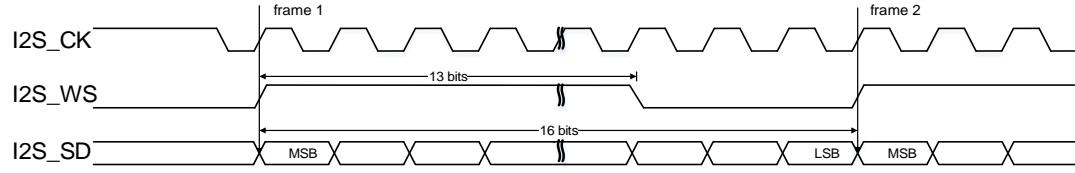


The timing diagrams for each configuration of the long frame synchronization mode are shown below.

**Figure 18-40. PCM standard long frame synchronization mode timing diagram (DTLEN=00, CHLEN=0, CKPL=0)**

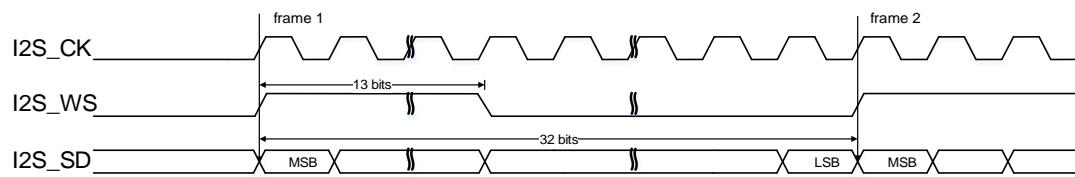


**Figure 18-41. PCM standard long frame synchronization mode timing diagram (DTLEN=00, CHLEN=0, CKPL=1)**

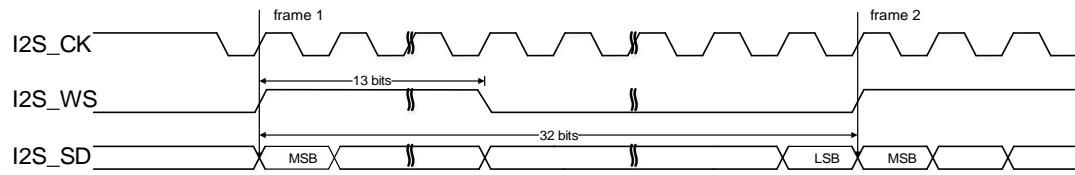


**Figure 18-42. PCM standard long frame synchronization mode timing diagram**

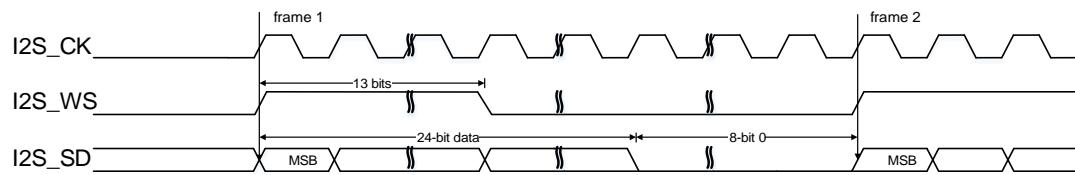
**(DTLEN=10, CHLEN=1, CKPL=0)**



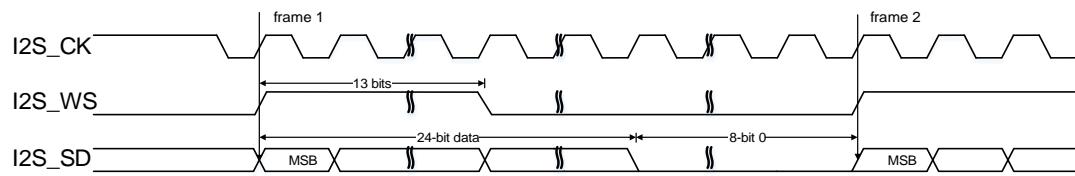
**Figure 18-43. PCM standard long frame synchronization mode timing diagram (DTLEN=10, CHLEN=1, CKPL=1)**



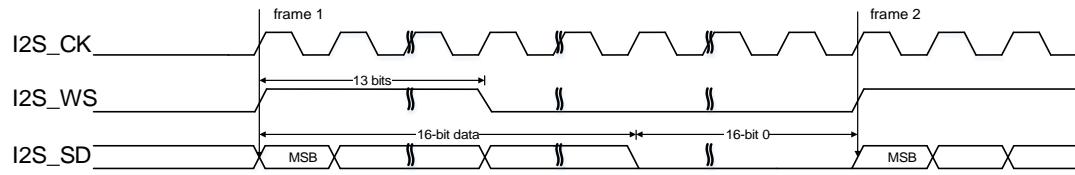
**Figure 18-44. PCM standard long frame synchronization mode timing diagram (DTLEN=01, CHLEN=1, CKPL=0)**



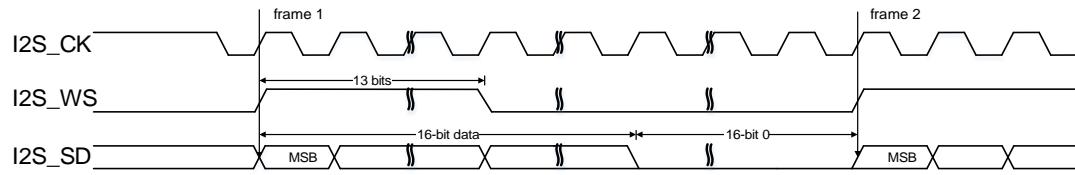
**Figure 18-45. PCM standard long frame synchronization mode timing diagram (DTLEN=01, CHLEN=1, CKPL=1)**



**Figure 18-46. PCM standard long frame synchronization mode timing diagram (DTLEN=00, CHLEN=1, CKPL=0)**

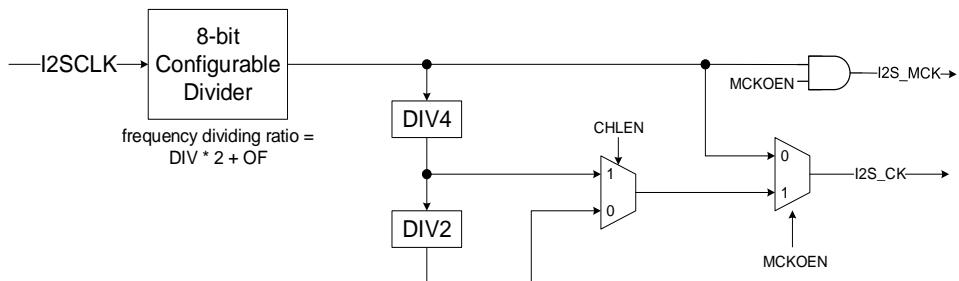


**Figure 18-47. PCM standard long frame synchronization mode timing diagram (DTLEN=00, CHLEN=1, CKPL=1)**



### 18.9.2. I2S clock

**Figure 18-48. Block diagram of I2S clock generator**



The block diagram of I2S clock generator is shown as [Figure 18-48. 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 18-4. I2S bitrate calculation formulas](#).

**Table 18-4. 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:

$$Fs = \text{I2S bitrate} / (\text{number of bits per channel} * \text{number of channels})$$

So, in order to get the desired audio sampling frequency, the clock generator needs to be configured according to the formulas listed in [Table 18-5. Audio sampling frequency calculation formulas](#).

**Table 18-5. 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))

### 18.9.3. 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 18-6. Direction of I2S interface signals for each operation mode.](#)

**Table 18-6. 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 to master mode, all the five steps should be done. In order to initialize I2S to 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 to define the I2S bitrate and determine whether I2S\_MCK needs to be provided or not.
- Step 2: Configure the CKPL in the SPI\_I2SCTL register 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 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 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 TBE 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 which channel side the data to transfer belongs to. 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 disable 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 disable and then enable I2S before resuming the communication.

For all standards except PCM, the I2SCH flag is used to distinguish which channel side the received data belongs to. 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.

1. 16-bit data packed in 32-bit frame in the LSB justified standard (DTLEN = 00, CHLEN = 1, and I2SSTD = 10)
2. Wait for the second last RBNE.
3. Then wait 17 I2S CK clock (clock on I2S\_CK pin) cycles.
4. 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 0b10)
  1. Wait for the last RBNE.
  2. Then wait one I2S clock cycle.
  3. Clear the I2SEN bit.
  7. For all other cases
  2. Wait for the second last RBNE.
  3. Then wait one I2S clock cycle.
  4. Clear the I2SEN bit.

### I2S slave transmission sequence

The transmission sequence in slave mode is similar to that in master mode. The differences between them are 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 disable and enable I2S to resume the communication. In slave mode, I2SCH is sensitive to the I2S\_WS signal coming from the external master.

In order to disable 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 differences between them are 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 disable I2S, it is mandatory to clear the I2SEN bit immediately after receiving the last RBNE.

#### 18.9.4. DMA function

DMA function is the same as SPI mode. The only difference is that the CRC function is not available in I2S mode.

## 18.10. I2S interrupts

### 18.10.1. 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 ongoing flag (TRANS)

TRANS is a status flag to indicate whether the transfer is ongoing or not. It is set and cleared by hardware and not controlled by software. This flag will not 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 will not generate any interrupt.

### 18.10.2. Error flags

There are three error flags:

- Transmission underrun error flag (TXURERR)

This situation occurs when the transmit buffer is empty if the valid SCK signal starts in slave transmission mode.

- Reception overrun error flag (RXORERR)

This situation 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.

- Format error (FERR)

In slave I2S mode, the I2S monitors the I2S\_WS signal and an error flag will be set if I2S\_WS toggles at an unexpected position.

I2S interrupt events and corresponding enable bits are summed up in the [Table 18-7. I2S interrupt.](#)

**Table 18-7. I2S interrupt**

Interrupt 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
TXURERR	Transmission underrun error	Read SPI_STAT register	ERRIE
RXORERR	Reception overrun error	Read SPI_DATA register and then read SPI_STAT register.	
FERR	I2S format error	Read SPI_STAT register	

## 18.11. Register definition

SPI0 base address: 0x4001 3000

SPI1/I2S1 base address: 0x4000 3800

SPI2/I2S2 base address: 0x4000 3C00

### 18.11.1. 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		

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 (TCRC) 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	Receive only mode When BDEN is cleared, this bit determines the direction of transfer. 0: Full-duplex mode 1: Receive-only mode
9	SWNSSEN	NSS software mode enable 0: NSS hardware mode. The NSS level depends on NSS pin. 1: NSS software mode. The NSS level depends on SWNSS bit. This bit has no meaning in SPI TI mode.
8	SWNSS	NSS pin selection in NSS software mode 0: NSS pin is pulled low 1: NSS pin is pulled high This bit effects only when the SWNSSEN bit is set. This bit has no meaning in SPI TI mode.
7	LF	LSB first mode 0: Transmit MSB first 1: Transmit LSB first This bit has no meaning in SPI TI mode.
6	SPIEN	SPI enable 0: SPI peripheral is disabled 1: SPI peripheral is enabled
5:3	PSC[2:0]	Master clock prescaler selection 000: PCLK/2 001: PCLK/4 010: PCLK/8 011: PCLK/16 100: PCLK/32 101: PCLK/64 110: PCLK/128 111: PCLK/256 PCLK means PCLK2 when using SPI0 or PCLK1 when using SPI1 and SPI2.
2	MSTMOD	Master mode enable 0: Slave mode 1: Master mode

---

1	CKPL	Clock polarity selection 0: CLK pin is pulled low when SPI is idle 1: CLK pin is pulled high when SPI is idle
0	CKPH	Clock phase selection 0: Capture the first data at the first clock transition 1: Capture the first data at the second clock transition

### 18.11.2. 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).

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								TBEIE	RBNEIE	ERRIE	TMOD	NSSP	NSSDRV	DMATEN	DMAREN

Bits	Fields	Descriptions
31:8	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, the CONFERR bit, the RXORERR bit or the TXURERR bit is set.
4	TMOD	SPI TI mode enable 0: SPI TI mode disabled. 1: SPI TI mode enabled.
3	NSSP	SPI NSS pulse mode enable 0: SPI NSS pulse mode disabled. 1: SPI NSS pulse mode enabled.
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, there will be a DMA request on 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, there will be a DMA request on corresponding DMA channel.

### 18.11.3. 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							FERR	TRANS	RXORER	CONFER	CRCERR	TXURER	I2SCH	TBE	RBNE
rc_w0							r	r	R	R	rc_w0	r	r	r	r

Bits	Fields	Descriptions
31:9	Reserved	Must be kept at reset value.
8	FERR	Format error SPI TI Mode: 0: No TI mode format error 1: TI mode format error occurs I2S Mode: 0: No I2S format error 1: I2S format error occurs This bit is set by hardware and cleared by writing 0.
7	TRANS	Transmitting ongoing 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 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

#### 18.11.4. Data register (SPI\_DATA)

Address offset: 0x0C

Reset value: 0x0000

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

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
SPI_DATA[15:0]															

rw

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	SPI_DATA[15:0]	<p>Data transfer register</p> <p>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.</p> <p>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-bit. 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.</p>

### 18.11.5. 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).

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
CRCPOLY[15:0]															

rw

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	CRCPOLY[15:0]	<p>CRC polynomial value</p> <p>These bits contain the CRC polynomial and they are used for CRC calculation. The default value is 0007h.</p>

### 18.11.6. 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).

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
RCRC[15:0]															

r

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	RCRC[15:0]	<p>RX CRC value</p> <p>When the CRCEN bit of SPI_CTL0 is set, the hardware computes the CRC value of the received bytes and saves them in RCRC register. If the data frame format is set to 8-bit data, CRC calculation is based on CRC8 standard, and saves the value in RCRC[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 RCRC[15:0].</p> <p>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.</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>

### 18.11.7. 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).

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
TCRC[15:0]															

r

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	TCRC[15:0]	<p>TX CRC value</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 TCRC register. If the data frame format is set to 8-bit data, CRC calculation is based on CRC8 standard, and saves the value in TCRC[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 TCRC[15:0].</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 formats (LF bit of the SPI\_CTL0) will get different CRC values.

This register is reset when the CRCEN bit in SPI\_CTL0 register or the SPIxRST bit in RCU reset register is set.

### 18.11.8. 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

Bits	Fields	Descriptions
31:12	Reserved	Must be kept at reset value.
11	I2SSEL	I2S mode selection 0: SPI mode 1: I2S mode This bit should be configured when SPI/I2S is disabled.
10	I2SEN	I2S enable 0: I2S is disabled 1: I2S is enabled This bit is not used in SPI mode.
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 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 is disabled.
		This bit is not used in SPI mode.
6	Reserved	Must be kept at reset value.
5:4	I2SSSTD[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 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 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.

### 18.11.9. 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).

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								MCKOEN	OF	DIV[7:0]						
rw								rw								

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:10	Reserved	Must be kept at reset value.
9	MCKOEN	<p>I2S_MCK output enable</p> <p>0: I2S_MCK output is disabled</p> <p>1: I2S_MCK output is enabled</p> <p>This bit should be configured when I2S is disabled.</p> <p>This bit is not used in SPI mode.</p>
8	OF	<p>Odd factor for the prescaler</p> <p>0: Real divider value is DIV * 2</p> <p>1: Real divider value is DIV * 2 + 1</p> <p>This bit should be configured when I2S is disabled.</p> <p>This bit is not used in SPI mode.</p>
7:0	DIV[7:0]	<p>Dividing factor for the prescaler</p> <p>Real divider value is DIV * 2 + OF.</p> <p>DIV must not be 0.</p> <p>These bits should be configured when I2S is disabled.</p> <p>These bits are not used in SPI mode.</p>

## 19. External memory controller (EXMC)

### 19.1. Overview

The external memory controller EXMC, is used as a translator for MCU to access a variety of external memory, it automatically converts AMBA memory access protocol into a specific memory access protocol defined in the configuration register, such as SRAM, ROM and NOR Flash. Users could also tweak with the timing parameters in the configuration registers to boost up memory access efficiency. EXMC access space is divided into multiple banks; each bank is assigned to access a specific memory type with flexible parameter configuration as defined in the control registers.

### 19.2. Characteristics

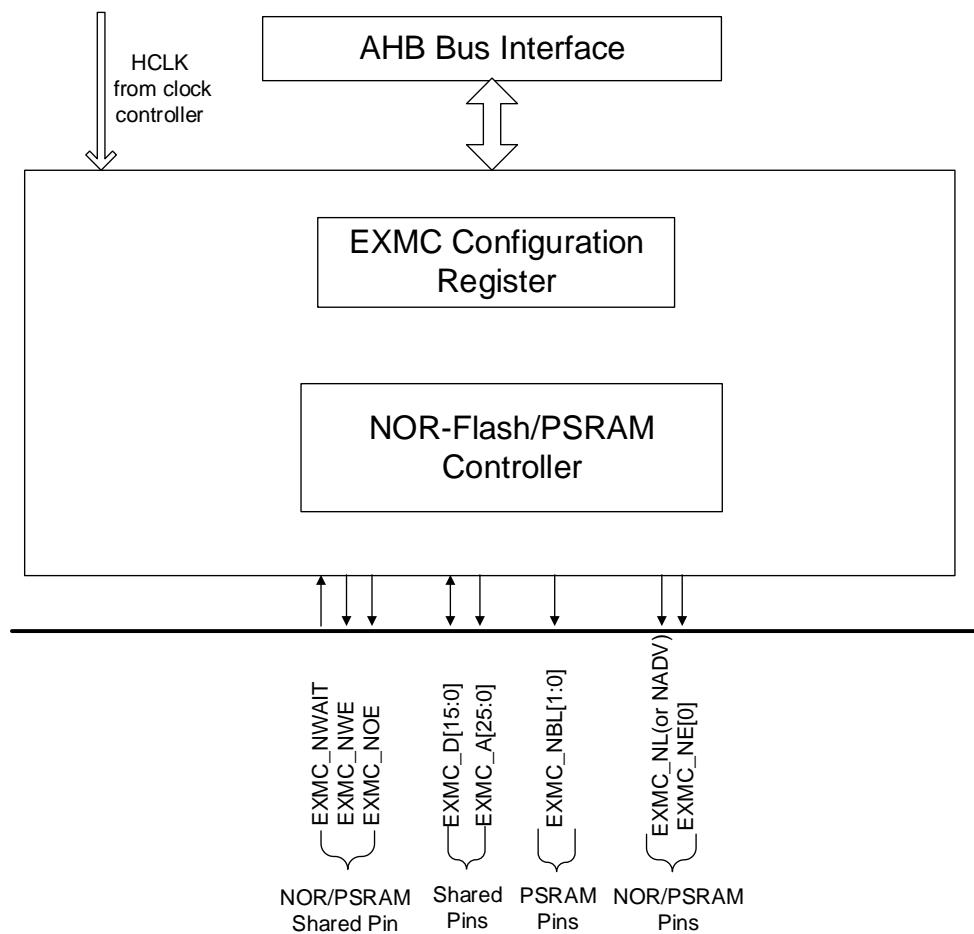
- Supported external memory:
  - SRAM
  - PSRAM
  - ROM
  - NOR Flash
- Protocol translation between the AMBA and the multitude of external memory protocol.
- Offering a variety of programmable timing parameters to meet user's specific needs.
- Independent read/write timing configuration to a sub-set memory type.
- 8 or 16 bits bus width.
- Address and data bus multiplexing mechanism for NOR Flash and PSRAM.
- Write enable and byte select are provided as needed.
- Automatic AMBA transaction split when internal and external bus width is not compatible.

### 19.3. Function overview

#### 19.3.1. Block diagram

EXMC is the combination of four modules: The AHB bus interface, EXMC configuration registers, NOR/PSRAM controller, AHB clock (HCLK) is the reference clock.

Figure 19-1. The EXMC block diagram



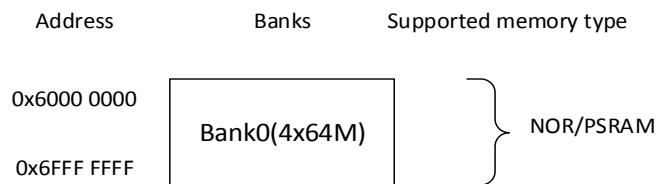
### 19.3.2. Basic regulation of EXMC access

EXMC is the conversion interface between AHB bus and external device protocol. 32-bit of AHB read/write accesses can be split into several consecutive 8-bit or 16-bit read/write operations respectively. In the process of data transfer, AHB access data width and memory data width may not be the same. In order to ensure consistency of data transmission, EXMC's read/write accesses follow the following basic regulation.

- When the width of AHB bus equals to the memory bus width. No conversion is applied.
- When the width of AHB bus is greater than memory bus width. The AHB accesses are automatically split into several continuous memory accesses.
- When the width of AHB bus is smaller than memory bus width. If the external memory devices have the byte selection function, such as SRAM, ROM, PSRAM, the application can access the corresponding byte through their byte lane EXMC\_NBL[1:0]. Otherwise, write operation is prohibited, but read operation is allowed unconditionally.

### 19.3.3. External device address mapping

**Figure 19-2. EXMC memory banks**



EXMC access space is divided into one bank. Bank0 is 4\*64 Mbytes. The first bank (Bank0) is further divided into 4 Regions, which is 64 Mbytes (only Bank0 Region0 is used).

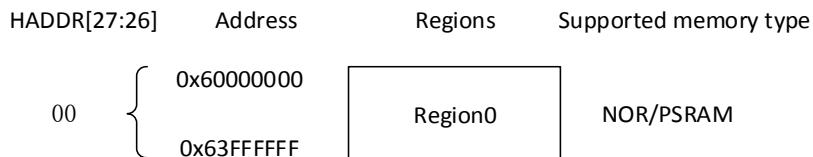
Each bank or region has a separate chip-select control signal, which can be configured independently.

Bank0 is used for NOR and PSRAM device access.

#### NOR/PSRAM address mapping

**Figure 19-3. Region of bank0 address mapping** reflects the address mapping of the region of bank0. Internal AHB address lines HADDR[27:26] bits are used to select the region, but only region0 is supported.

**Figure 19-3. Region of bank0 address mapping**



HADDR[25:0] is the byte address whereas the external memory may not be byte accessed, this will lead to address inconsistency. EXMC can adjust HADDR to accommodate the data width of the external memory according to the following rules.

- When data bus width of the external memory is 8-bits. In this case the memory address is byte aligned. HADDR[25:0] is connected to EXMC\_A[25:0] and then the EXMC\_A[25:0] is connected to the external memory address lines.
- When data bus width of the external memory is 16-bits. In this case the memory address is half-word aligned. HADDR byte address must be converted into half-word aligned by connecting HADDR[25:1] with EXMC\_A[24:0]. The EXMC\_A[24:0] is connected to the external memory address lines.

### 19.3.4. NOR/PSRAM controller

NOR/PSRAM memory controller controls bank0, which is designed to support NOR Flash, PSRAM, SRAM, ROM and honeycomb RAM external memory. EXMC has 1 independent chip-select signals for 1 sub-bank within bank0, named NE[0]. Other signals for NOR/PSRAM access are shared. Each sub-bank has its own set of configuration register.

**Note:**

In asynchronous mode, all output signals of controller will change on the rise edge of internal AHB bus clock (HCLK).

#### NOR/PSRAM memory device interface description

**Table 19-1. NOR Flash interface signals description**

EXMC Pin	Direction	Mode	Functional description
Muxed EXMC_A[25:16]	Output	Async	Address bus signal
EXMC_D[15:0]	Input/output	Async	Address/Data bus
EXMC_NE[x]	Output	Async	Chip selection, x=0
EXMC_NOE	Output	Async	Read enable
EXMC_NWE	Output	Async	Write enable
EXMC_NWAIT	Input	Async	Wait input signal
EXMC_NL(NADV)	Output	Async	Address valid

**Table 19-2. PSRAM muxed signal description**

EXMC Pin	Direction	Mode	Functional description
EXMC_A[25:16]	Output	Async	Address Bus
EXMC_D[15:0]	Input/output	Async	Data Bus
EXMC_NE[x]	Output	Async	Chip selection, x=0
EXMC_NOE	Output	Async	Read enable
EXMC_NWE	Output	Async	Write enable
EXMC_NWAIT	Input	Async	Wait input signal
EXMC_NL(NADV)	Output	Async	Latch enable (address valid enable, NADV)
EXMC_NBL[1]	Output	Async	Upper byte enable
EXMC_NBL[0]	Output	Async	Lower byte enable

#### Supported memory access mode

Table below shows an example of the supported devices type, access modes and transactions when the memory data bus is 16-bit for NOR, PSRAM and SRAM.

Table 19-3. EXMC bank 0 supports all transactions

Memory	Access Mode	R/W	AHB Transaction Size	Memory Transaction Size	Comments
NOR Flash	Async	R	8	16	
	Async	R	16	16	
	Async	W	16	16	
	Async	R	32	16	Split into 2 EXMC accesses
	Async	W	32	16	Split into 2 EXMC accesses
PSRAM	Async	R	8	16	
	Async	W	8	16	Use of byte lanes EXMC_NBL[1:0]
	Async	R	16	16	
	Async	W	16	16	
	Async	R	32	16	Split into 2 EXMC accesses
	Async	W	32	16	Split into 2 EXMC accesses
SRAM and ROM	Async	R	8	8	
	Async	R	8	16	
	Async	R	16	8	Split into 2 EXMC accesses
	Async	R	16	16	
	Async	R	32	8	Split into 4 EXMC accesses
	Async	R	32	16	Split into 2 EXMC accesses
	Async	W	8	8	
	Async	W	8	16	Use of byte lanes EXMC_NBL[1:0]
	Async	W	16	8	
	Async	W	16	16	
	Async	W	32	8	
	Async	W	32	16	

### NOR Flash/PSRAM controller timing

EXMC provides various programmable timing parameters and timing models for SRAM, ROM, PSRAM, NOR Flash and other external static memory.

**Table 19-4. NOR / PSRAM controller timing parameters**

Parameter	Function	Access mode	Unit	Min	Max
BUSLAT	Bus latency	Async read	HCLK	1	16
DSET	Data setup time	Async	HCLK	2	256
AHLD	Address hold time	Async(muxed)	HCLK	2	16
ASET	Address setup time	Async	HCLK	1	16

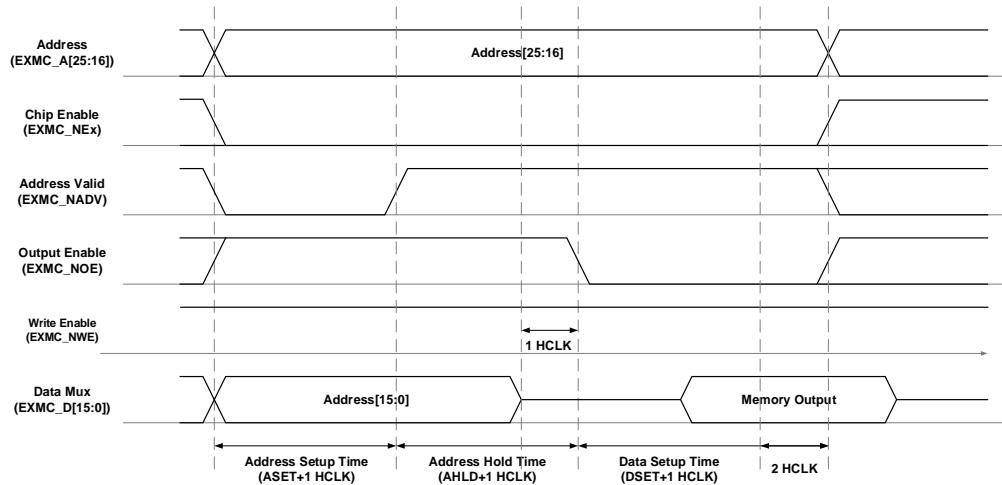
**Table 19-5. EXMC\_timing models**

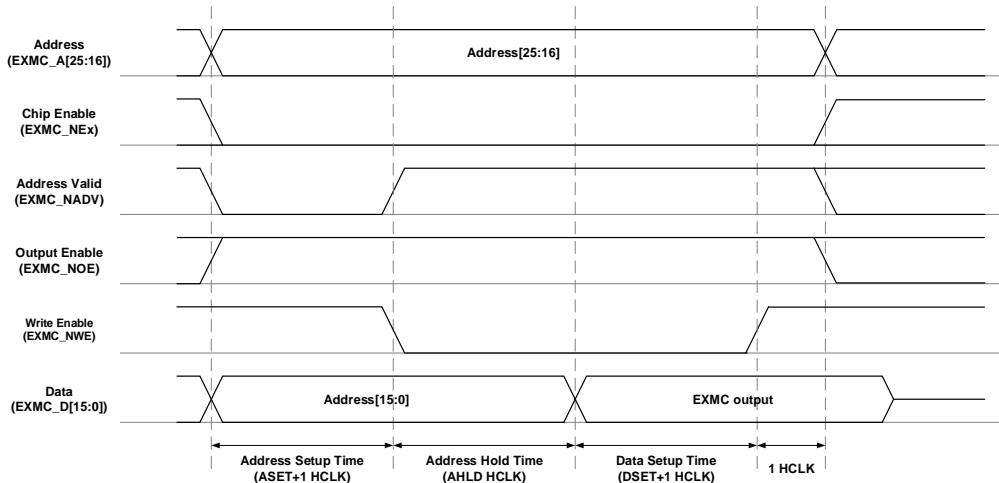
Timing model		Extend mode	Mode description	Write timing parameter	Read timing parameter
Async	Mode AM	0	NOR Flash address/data mux	DSET AHLD ASET BUSLAT	DSET AHLD ASET BUSLAT

As shown in [Table 19-5. EXMC\\_timing models](#), EXMC NOR Flash / PSRAM controller provides a timing model, users can modify those parameters listed in [Table 19-4. NOR / PSRAM controller timing parameters](#) to satisfy different external memory type and user's requirements. Different timing patterns for read and write access could be generated independently according to EXMC\_SNTCFGx register's configuration.

### Asynchronous access timing diagram

Mode AM - NOR Flash address / data bus multiplexing

**Figure 19-4. Multiplex mode read access**


**Figure 19-5. Multiplex mode write access**

**Table 19-6. Multiplex mode related registers configuration**

EXMC_SNCTLx		
Bit Position	Bit Name	Reference Setting Value
31-16	Reserved	0x0000
15	ASYNCWAIT	Depends on memory
14	Reserved	0x0
13	NRWTEN	0x0
12	WREN	Depends on memory
11:10	Reserved	0x0
9	NRWTPOL	Meaningful only when the bit 15 is set to 1
8-7	Reserved	0x1
6	NREN	0x1
5-4	NRW	Depends on memory
3-2	NRTP	0x2:NOR Flash
1	NRMUX	0x1
0	NRBKEN	0x1
EXMC_SNTCFGx		
Bit Position	Bit Name	Reference Setting Value
31-20	Reserved	0x0
19-16	BUSLAT	Minimum time between EXMC_NE[0] rising edge to EXMC_NE[0] falling edge
15-8	DSET	Depends on memory and user
7-4	AHLD	Depends on memory and user
3-0	ASET	Depends on memory and user

Wait timing of asynchronous communication

Wait feature is controlled by the bit ASYNCWAIT in register EXMC\_SNCTLx. During extern memory access, data setup phase will be automatically extended by the active EXMC\_NWAIT signal if ASYNCWAIT bit is set. The extend time is calculated as follows:

If memory wait signal is aligned to EXMC\_NOE/ EXMC\_NWE:

$$T_{DATA\_SETUP} \geq maxT_{WAIT\_ASSERTION} + 4HCLK$$

If memory wait signal is aligned to EXMC\_NE:

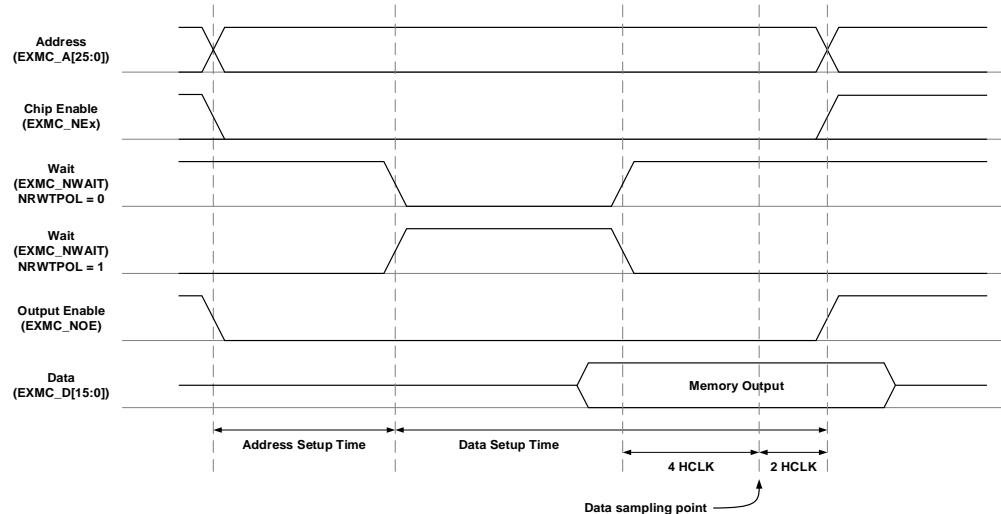
$$\text{If } maxT_{WAIT\_ASSERTION} \geq T_{ADDRES\_PHASE} + T_{HOLD\_PHASE}$$

$$T_{DATA\_SETUP} \geq (maxT_{WAIT\_ASSERTION} - T_{ADDRES\_PHASE} - T_{HOLD\_PHASE}) + 4HCLK$$

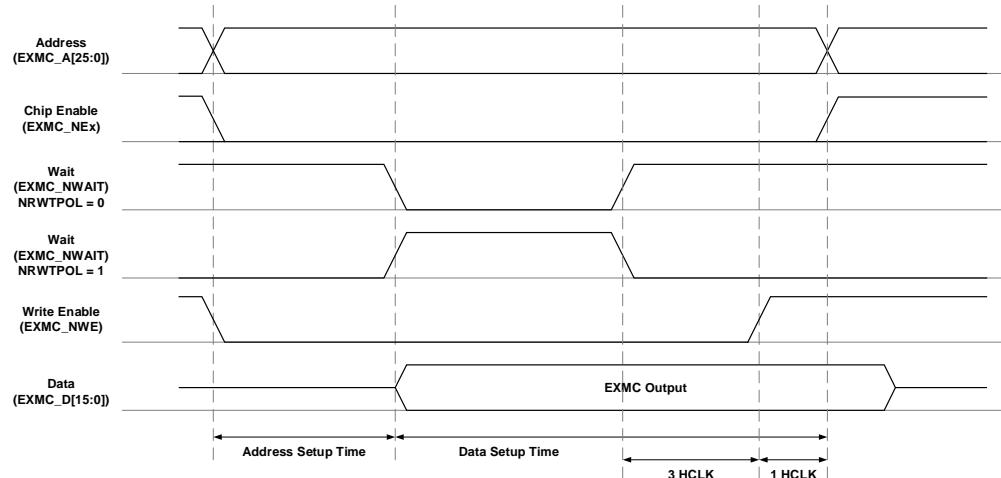
Otherwise

$$T_{DATA\_SETUP} \geq 4HCLK$$

**Figure 19-6. Read access timing diagram under async-wait signal assertion**



**Figure 19-7. Write access timing diagram under async-wait signal assertion**



## 19.4. Register definition

EXMC base address: 0xA000 0000

### 19.4.1. NOR/PSRAM controller registers

#### SRAM/NOR Flash control registers (EXMC\_SNCTLx) (x=0)

Address offset: 0x00 + 8 \* x, (x = 0)

Reset value: 0x0000 30DA for region0.

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
ASYNC WAIT	Reserved	NRWTEN	WREN	Reserved	NRWT POL	Reserved	NRREN	NRW[1:0]	NRTP[1:0]	NRMUX	NRBKEN				

rw                    rw

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15	ASYNCWAIT	Asynchronous wait 0: Disable the asynchronous wait feature 1: Enable the asynchronous wait feature
14	Reserved	Must be kept at reset value.
13	NRWTEN	NWAIT signal enable For Flash memory access in burst mode, this bit enables/disables wait-state insertion via the NWAIT signal: 0: Disable NWAIT signal 1: Enable NWAIT signal
12	WREN	Write enable 0: Disabled write in the bank by the EXMC, otherwise an AHB error is reported 1: Enabled write in the bank by the EXMC (default after reset)
11:10	Reserved	Must be kept at reset value.
9	NRWPOL	NWAIT signal polarity 0: Low level is active of NWAIT 1: High level is active of NWAIT

8:7	Reserved	Must be kept at reset value.
6	NREN	NOR Flash access enable 0: Disable NOR Flash access 1: Enable NOR Flash access
5:4	NRW[1:0]	NOR region memory data bus width 00: 8 bits 01: 16 bits(default after reset) 10/11: Reserved
3:2	NRTP[1:0]	NOR region memory type 00: SRAM 01: PSRAM (CRAM) 10: NOR Flash(default after reset for region0) 11: Reserved
1	NRMUX	NOR region memory address/data multiplexing 0: Disable address/data multiplexing function 1: Enable address/data multiplexing function
0	NRBKEN	NOR region enable 0: Disable the corresponding memory bank 1: Enable the corresponding memory bank

### SRAM/NOR Flash timing configuration registers (EXMC\_SNTCFGx) (x=0)

Address offset: 0x04 + 8 \* x, (x = 0)

Reset value: 0xFFFF FFFF

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												BUSLAT[3:0]			
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DSET[7:0]								AHLD[3:0]				ASET[3:0]			
rw								rw				rw			

Bits	Fields	Descriptions
31:24	Reserved	Must be kept at reset value.
19:16	BUSLAT[3:0]	Bus latency The bits are defined in multiplexed read mode in order to avoid bus contention, and represent the data bus to return to a high impedance state's minimum. 0x0: Bus latency = 1 * HCLK period 0x1: Bus latency = 2 * HCLK period

		.....
		0xF: Bus latency = 16 * HCLK period
15:8	DSET[7:0]	Data setup time This field is meaningful only in asynchronous access. 0x00: Reserved 0x01: Data setup time = 2 * HCLK period .....
		0xFF: Data setup time = 256 * HCLK period
7:4	AHLD[3:0]	Address hold time This field is used to set the time of address hold phase. 0x0: Reserved 0x1: Address hold time = 2 * HCLK .....
		0xF: Address hold time = 16 * HCLK
3:0	ASET[3:0]	Address setup time This field is used to set the time of address setup phase. <b>Note:</b> meaningful only in asynchronous access of SRAM,ROM,NOR Flash 0x0: Address setup time = 1 * HCLK .....
		0xF: Address setup time = 16 * HCLK

## 20. Controller area network (CAN)

### 20.1. 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, 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 used 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.

### 20.2. Characteristics

- Supports CAN protocols version 2.0A, B
- Baud rates up to 1 Mbit/s
- Supports the time-triggered communication
- Interrupt enable and clear

#### Transmission

- Supports 3 transmit mailboxes
- Prioritization of messages
- Supports Time Stamp at SOF transmission

#### Reception

- Supports 2 receive FIFOs and each has 3 messages deep
- 28 scalable/configurable identifier filter banks in GD32VF103
- FIFO lock

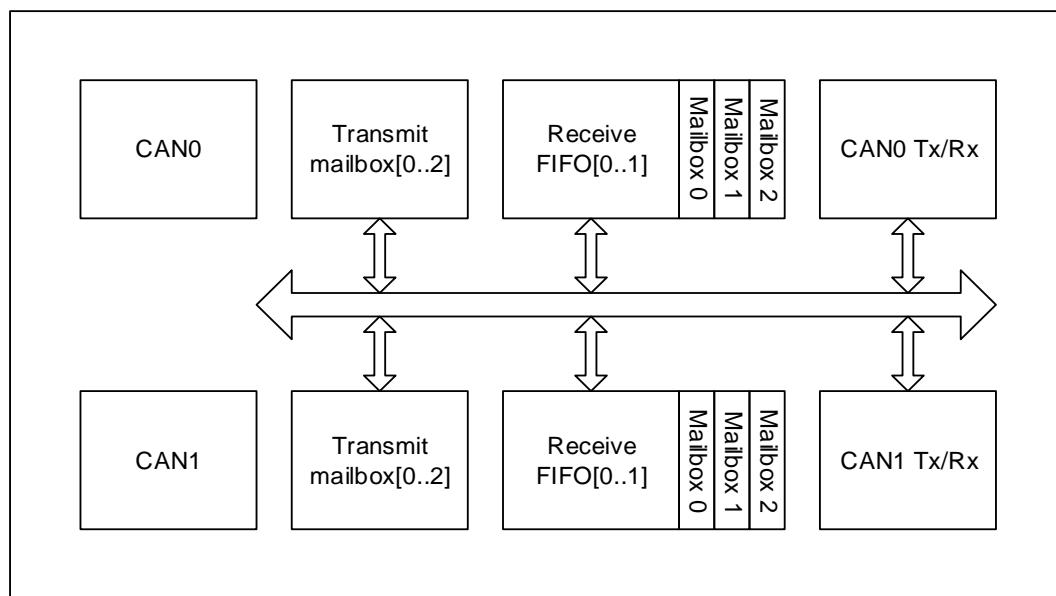
#### Time-triggered communication

- Disable retransmission automatically
- 16-bit free timer
- Time Stamp on SOF reception
- Time Stamp sent in last two data bytes

## 20.3. Function overview

[Figure 20-1. CAN module block diagram](#) shows the CAN block diagram.

Figure 20-1. CAN module block diagram



### 20.3.1. 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.

## 20.3.2. 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, the RX pin is disconnected from the CAN network and the TX pin can send messages to the CAN network.

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 transferred 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

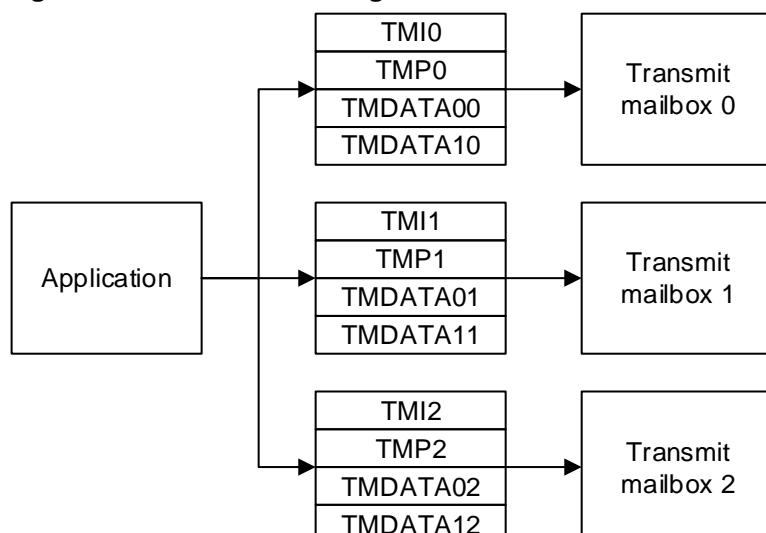
Normal communication mode is the default communication mode unless the LCMOD or SCMOD bit in CAN\_BT register is set.

## 20.3.3. Data transmission

### Transmission register

Three transmit mailboxes are transparent to the application. You can use transmit mailboxes through four transmission registers: CAN\_TMIx, CAN\_TMPx, CAN\_TMDATA0x and CAN\_TMDATA1x. As shown in [Figure 20-2. Transmission register](#).

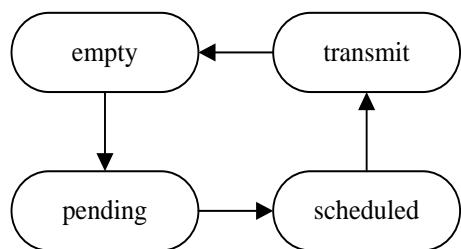
**Figure 20-2. Transmission register**



### 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 20-3. State of transmission mailbox](#).

**Figure 20-3. 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 without any error.
- 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 transmission registers with the application's acquirement.

Step 3: Set TE 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 does not take effect immediately until the transmission is finished. 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.

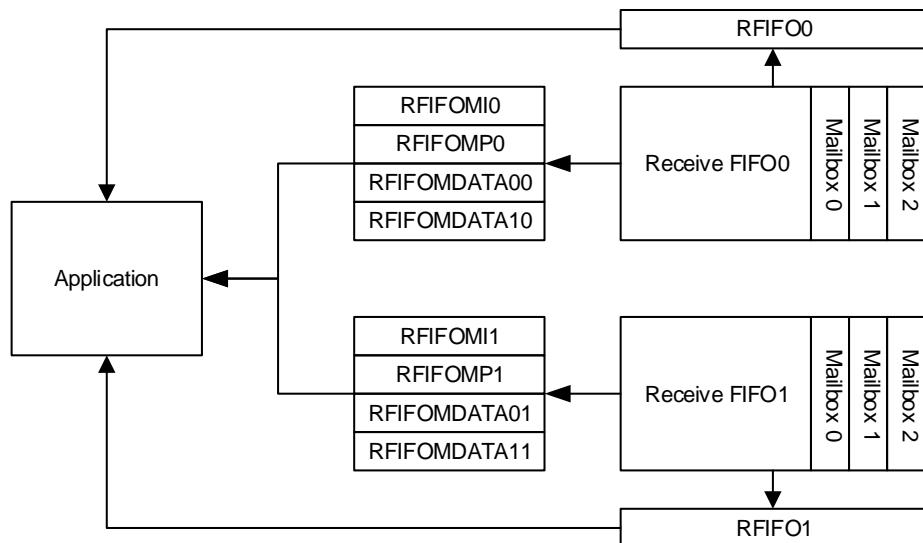
## 20.3.4. Data reception

### 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 20-4. Reception register](#).

**Figure 20-4. 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.

### 20.3.5. Filtering function

The CAN would receive frames from the CAN bus. If the frame is passed through the filter, it is stored into the receive FIFOs. Otherwise, the frame will be discarded without intervention by the software.

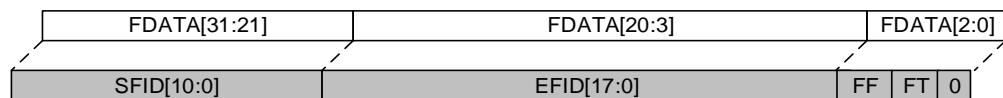
The identifier of frame from the CAN bus takes part in the matching of the filter.

#### Scale

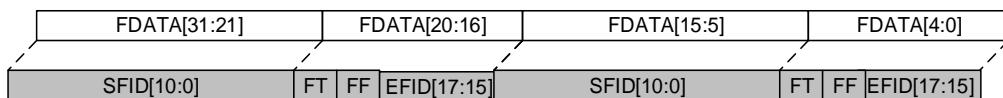
The filter consists of 28 banks: bank0 to bank27. Each bank has two 32-bit registers: CAN\_FxDATA0 and CAN\_FxDATA1.

Each filter bank can be configured to 32-bit or 16-bit.

32-bit: SFID[10:0], EFID[17:0], FF and FT bits. As shown in [Figure 20-5. 32-bit filter](#).

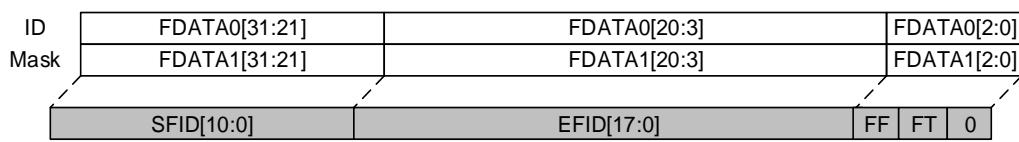
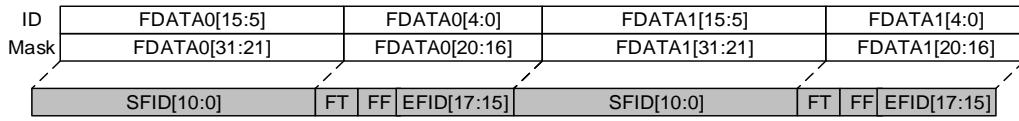
**Figure 20-5. 32-bit filter**


16-bit: SFID [10:0], FT, FF and EFID[17:15] bits. As shown in [Figure 20-6. 16-bit filter](#).

**Figure 20-6. 16-bit filter**


### Mask mode

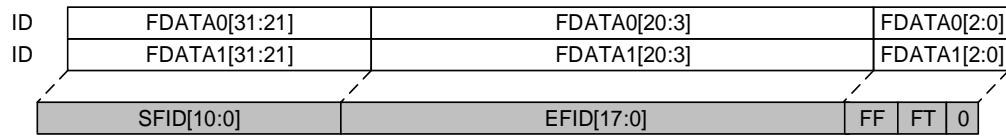
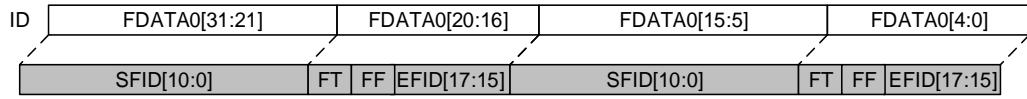
In mask mode the identifier registers are associated with mask registers specifying which bits of the identifier are handled as “must match” (when the bit in mask register is ‘1’) or as “don’t care” (when the bit in mask register is ‘0’). 32-bit mask mode example is shown in [Figure 20-7. 32-bit mask mode filter](#).

**Figure 20-7. 32-bit mask mode filter**

**Figure 20-8. 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 [Figure 20-9. 32-bit list mode filter](#).

**Figure 20-9. 32-bit list mode filter**

**Figure 20-10. 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 20-1. 32-bit filter number](#).

**Table 20-1. 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 FIFO0, 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 bits in CAN\_RFIFOMPx when the frame is read by the application.

Each FIFO numbers the filters within the banks associated with the FIFO itself whether the bank is active or not.

The example about filtering index is shown in [Table 20-2. Filtering index](#).

**Table 20-2. 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[31:16]-16bit-Mask		
1	F1DATA0-32bit-ID	Yes	1	2	F2DATA1[15:0]-16bit-ID	Yes	1
	F1DATA1-32bit-ID		2		F2DATA1[31:16]-16bit-Mask		
3	F3DATA0[15:0]-16bit-ID	No	3	4	F4DATA0-32bit-ID	No	2
	F3DATA0[31:16]-16bit-Mask				F4DATA1-32bit-Mask		

Filter Bank	FIFO0	Active	Filter Number	Filter Bank	FIFO1	Active	Filter Number	
	F3DATA1[15:0]-16bit-ID		4	5	F5DATA0-32bit-ID	No	3	
	F3DATA1[31:16]-16bit-Mask				F5DATA1-32bit-ID		4	
7	F7DATA0[15:0]-16bit-ID	No	5	6	F6DATA0[15:0]-16bit-ID	Yes	5	
	F7DATA0[31:16]-16bit-ID		6		F6DATA0[31:16]-16bit-ID		6	
	F7DATA1[15:0]-16bit-ID		7		F6DATA1[15:0]-16bit-ID		7	
	F7DATA1[31:16]-16bit-ID		8		F6DATA1[31:16]-16bit-ID		8	
8	F8DATA0[15:0]-16bit-ID	Yes	9	10	F10DATA0[15:0]-16bit-ID	No	9	
	F8DATA0[31:16]-16bit-ID		10		F10DATA0[31:16]-16bit-Mask		10	
	F8DATA1[15:0]-16bit-ID		11		F10DATA1[15:0]-16bit-ID			
	F8DATA1[31:16]-16bit-ID		12		F10DATA1[31:16]-16bit-Mask			
9	F9DATA0[15:0]-16bit-ID	Yes	13	11	F11DATA0[15:0]-16bit-ID	No	11	
	F9DATA0[31:16]-16bit-Mask				F11DATA0[31:16]-16bit-ID		12	
	F9DATA1[15:0]-16bit-ID		14		F11DATA1[15:0]-16bit-ID		13	
	F9DATA1[31:16]-16bit-Mask				F11DATA1[31:16]-16bit-ID		14	
12	F12DATA0-32bit-ID	Yes	15	13	F13DATA0-32bit-ID	Yes	15	
	F12DATA1-32bit-Mask				F13DATA1-32bit-ID		16	

## Priority

The filters have the priority:

1. 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.

### 20.3.6. Time-triggered communication

The time-triggered CAN protocol is a higher layer protocol on top of the CAN 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.

### 20.3.7. 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 ( $1 \times t_{CAN}$ ).

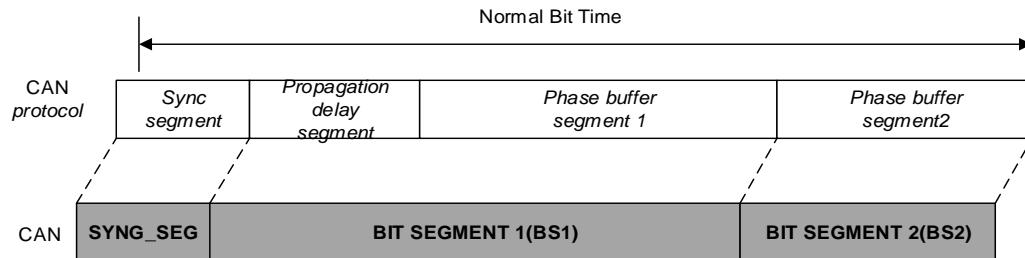
**Bit segment 1 (BS1):** defines the location of the sample point. It includes the *Propagation delay segment* and *Phase buffer segment 1* of the CAN standard. Its duration is programmable between 1 and 16 time quanta but may be automatically lengthened to compensate for positive phase drifts due to 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 20-11. The bit time](#).

**Figure 20-11. 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:

$$\text{BaudRate} = \frac{1}{\text{Normal Bit Time}} \quad (21-1)$$

$$\text{Normal Bit Time} = t_{SYNC\_SEG} + t_{BS1} + t_{BS2} \quad (21-2)$$

with:

$$t_{SYNC\_SEG} = 1 \times t_{CAN} \quad (21-3)$$

$$t_{BS1} = (1 + BT.BS1) \times t_{CAN} \quad (21-4)$$

$$t_{BS2} = (1 + BT.BS2) \times t_{CAN} \quad (21-5)$$

$$t_{CAN} = (1 + BT.BRP) \times t_{PCLK1} \quad (21-6)$$

### 20.3.8. 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.

### 20.3.9. 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.

## 20.4. Register definition

CAN0 base address: 0x4000 6400

CAN1 base address: 0x4000 6800

### 20.4.1. Control register (CAN\_CTL)

Address offset: 0x00

Reset value: 0x0001 0002

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															DFZ
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWRST	Reserved							TTC	ABOR	AWU	ARD	RFOD	TFO	SLPWMD	IWMOD
rs								rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:17	Reserved	Must be kept at reset value
16	DFZ	<p>Debug freeze</p> <p>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.</p> <p>0: CAN reception and transmission working normal even during debug</p> <p>1: CAN reception and transmission stop working during debug</p>
15	SWRST	<p>Software reset</p> <p>0: No effect</p> <p>1: Reset CAN with working mode of sleep. This bit is automatically reset to 0</p>
14:8	Reserved	Must be kept at reset value
7	TTC	<p>Time-triggered communication</p> <p>0: Disable time-triggered communication</p> <p>1: Enable time-triggered communication</p>
6	ABOR	<p>Automatic bus-off recovery</p> <p>0: The bus-off state is left manually by software</p> <p>1: The bus-off state is left automatically by hardware</p>
5	AWU	<p>Automatic wakeup</p> <p>If this bit is set, the sleep mode left when CAN bus activity detected, and</p>

		SLPWMOD bit in CAN_CTL register will be cleared automatically.
0		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 be 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

#### 20.4.2. Status register (CAN\_STAT)

Address offset: 0x04

Reset value: 0x0000 0C02

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				RXL	LASTRX	RS	TS	Reserved			SLPIF	WUIF	ERRIF	SLPWS	IWS
				r	r	r	r				rc_w1	rc_w1	rc_w1	r	r

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	<p>Receiving state</p> <p>0: CAN is not working in the receiving state</p> <p>1: CAN is working in the receiving state</p>
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>

1: CAN is the state of sleep working mode

0	IWS	Initial working state
		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 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

#### 20.4.3. 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	MTFNER R2	MTF2			
r	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	MTFNER R1	MTF1	MST0	Reserved	MTE0	MAL0	MTFNER R0	MTF0				
rs		rc_w1	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	<p>Transmit mailbox 1 empty</p> <p>0: Transmit mailbox 1 not empty</p> <p>1: Transmit mailbox 1 empty</p>
26	TME0	<p>Transmit mailbox 0 empty</p> <p>0: Transmit mailbox 0 not empty</p> <p>1: Transmit mailbox 0 empty</p>
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</p> <p>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</p> <p>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	Mailbox 1 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 MTF1 bit in CAN\_TSTAT register. This bit reset by hardware when next transmit start.

10	MAL1	Mailbox 1 arbitration lost  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 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

#### **20.4.4. Receive message FIFO0 register (CAN\_RFIFO0)**

Address offset: 0x0C

Reset value: 0x0000 0000

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

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									RFD0	RFO0	RFF0	Reserved	RFL0[1:0]		

rs      rc\_w1      rc\_w1      r

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.

#### **20.4.5. Receive message FIFO1 register (CAN\_RFIFO1)**

Address offset: 0x10

Reset value: 0x0000 0000

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

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								RFD1	RFO1	RFF1	Reserved	RFL1[1:0]			

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.

#### 20.4.6. Interrupt enable register (CAN\_INTEN)

Address offset: 0x14

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														SLPWIE	WIE
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERRIE	Reserved		ERRNIE	BOIE	PERRIE	WERRIE	Reserved	RFOIE1	RFFIE1	RFNEIE1	RFOIE0	RFFIE0	RFNEIE0	TMEIE	

rw rw

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
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

#### 20.4.7. Error register (CAN\_ERR)

Address offset: 0x18

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
RECNT[7:0]								TECNT[7:0]							
r								r							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								ERRN[2:0]		Reserved	BOERR	PERR	WERR	rw	
										r	r	r			

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.

#### 20.4.8. Bit timing register (CAN\_BT)

Address offset: 0x1C

Reset value: 0x0123 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
SCMOD	LCMOD	Reserved				SJW[1:0]	Reserved	BS2[2:0]				BS1[3:0]			
rw	rw					rw						rw			rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						BAUDPSC[9:0]									
															rw

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

#### 20.4.9. 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)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SFID[10:0]/EFID[28:18]												EFID[17:13]			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EFID[12:0]												FF	FT	TEN	
												rw	rw	rw	

Bits	Fields	Descriptions
31:21	SFID[10:0]/EFID[28:1]	The frame identifier
8]		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

### **20.4.10. 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)

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

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	DLEN[3:0]	Data length code DLEN[3:0] is the number of bytes in a frame.

### **20.4.11. Transmit mailbox data0 register (CAN\_TMDATA0x) (x=0..2)**

Address offset: 0x188, 0x198, 0x1A8

Reset value: 0xFFFF 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
DB3[7:0]								DB2[7:0]							
rw								rw							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DB1[7:0]								DB0[7:0]							
rw								rw							

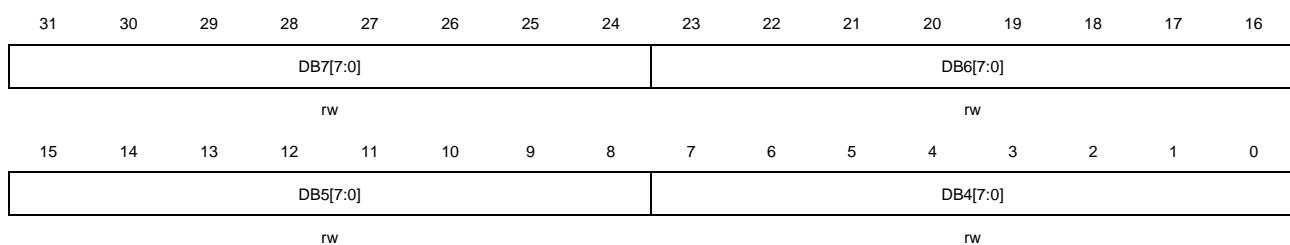
<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:24	DB3[7:0]	Data byte 3
23:16	DB2[7:0]	Data byte 2
15:8	DB1[7:0]	Data byte 1
7:0	DB0[7:0]	Data byte 0

#### 20.4.12. Transmit mailbox data1 register (CAN\_TMDATA1x) (x=0..2)

Address offset: 0x18C, 0x19C, 0x1AC

Reset value: 0xFFFF XXXX

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



<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
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

#### 20.4.13. Receive FIFO mailbox identifier register (CAN\_RFIFOIx) (x=0,1)

Address offset: 0x1B0, 0x1C0

Reset value: 0xFFFF XXXX

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



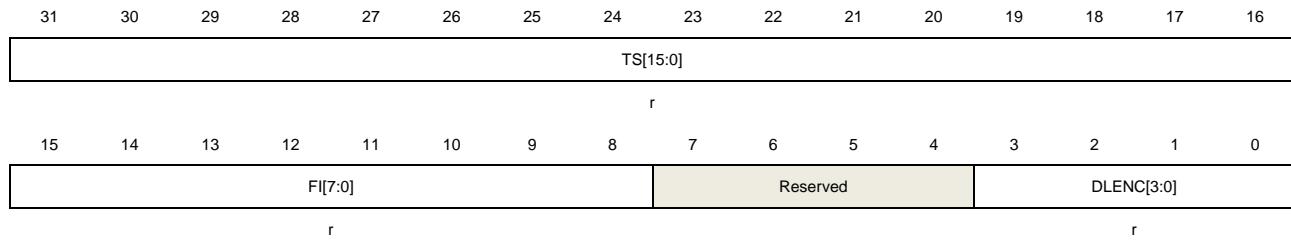
<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:21	SFID[10:0]/EFID[28:1]	The frame identifier SFID[10:0]: Standard format frame identifier EFID[28:18]: Extended format frame identifier
8]		
20:16	EFID[17:13]	The frame identifier EFID[17:13]: Extended format frame identifier
15:3	EFID[12:0]	The frame 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

#### 20.4.14. 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)



<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
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	DLEN[3:0]	Data length code DLEN[3:0] is the number of bytes in a frame.

### **20.4.15. Receive FIFO mailbox data0 register (CAN\_RFIFOMDATA0x) (x=0,1)**

Address offset: 0x1B8, 0x1C8

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
DB3[7:0]								DB2[7:0]							
r								r							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DB1[7:0]								DB0[7:0]							
r								r							

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
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

### **20.4.16. 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)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DB7[7:0]								DB6[7:0]							
r								r							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DB5[7:0]								DB4[7:0]							
r								r							

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
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

#### **20.4.17. Filter control register (CAN\_FCTL)**

Address offset: 0x200

Reset value: 0x2A1C 0E01

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	HBC1F[5:0]					Reserved					FLD				

Bits	Fields	Descriptions
31:14	Reserved	Must be kept at reset value
13:8	HBC1F[5:0]	<p>Header bank of CAN1 filter</p> <p>These bits are set and cleared by software to define the first bank for CAN1 filter.</p> <p>Bank0 ~ Bank HBC1F-1 used to CAN0. Bank HBC1F ~ Bank27 used to CAN1.</p> <p>When set 0, not bank used to CAN0. When set 28, not bank used to CAN1.</p>
7:1	Reserved	Must be kept at reset value
0	FLD	<p>Filter lock disable</p> <p>0: Filter lock enable</p> <p>1: Filter lock disable</p>

#### 20.4.18. 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.

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
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

#### 20.4.19. 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

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
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

#### 20.4.20. 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

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
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

#### 20.4.21. 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

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:28	Reserved	Must be kept at reset value
27:0	FWx	Filter working 0: Filter x working disable 1: Filter x working enable

#### 20.4.22. 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: 0xFFFF 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	<p>Filter data</p> <p>Mask mode</p> <p>0: Mask match disable</p> <p>1: Mask match enable</p> <p>List mode</p> <p>0: List identifier bit is 0</p> <p>1: List identifier bit is 1</p>

## 21. Universal serial bus full-speed interface (USBFS)

The USBFS is available on GD32VF103 series.

### 21.1. Overview

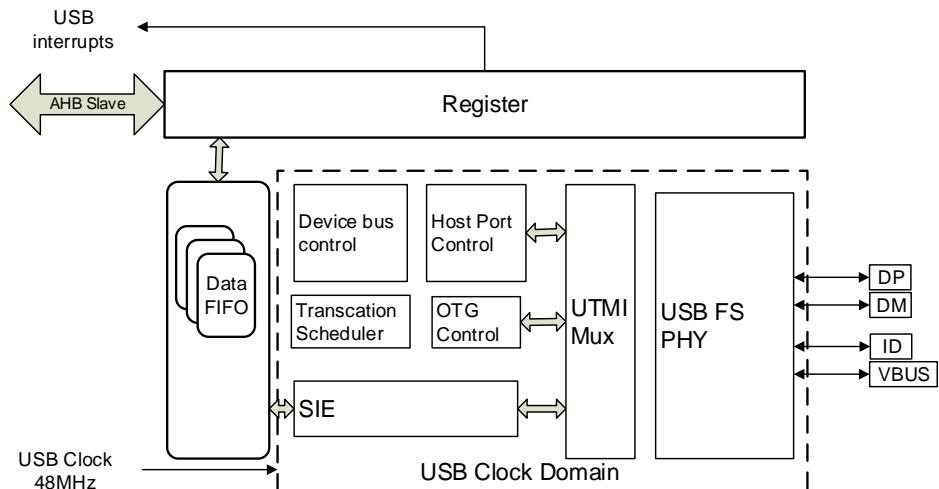
USB Full-Speed (USBFS) controller provides a USB-connection solution for portable devices. USBFS supports host and device modes, as well as OTG mode with HNP (Host Negotiation Protocol) and SRP (Session Request Protocol). USBFS contains a full-speed internal USB PHY and external PHY chip is not contained. USBFS supports all the four types of transfer (control, bulk, Interrupt and isochronous) which defined in USB 2.0 protocol.

### 21.2. Characteristics

- Supports USB 2.0 host mode at Full-Speed(12Mb/s) or Low-Speed(1.5Mb/s)
- Supports USB 2.0 device mode at Full-Speed(12Mb/s)
- Supports OTG protocol with HNP (Host Negotiation Protocol) and SRP (Session Request Protocol)
- Supports all the 4 types of transfer: control, bulk, interrupt and isochronous
- Includes a USB transaction scheduler in host mode to handle USB transaction request efficiently.
- Includes a 1.25KB FIFO RAM.
- Supports 8 channels in host mode.
- Includes 2 transmit FIFOs (periodic and non-periodic) and a receive FIFO (shared by all channels) in host mode.
- Includes 4 transmit FIFOs (one for each IN endpoint) and a receive FIFO (shared by all OUT endpoints) in device mode.
- Supports 4 OUT and 4 IN endpoints in device mode.
- Supports remote wakeup in device mode.
- Includes a Full-Speed USB PHY with OTG protocol supported.
- Time intervals of SOFs is dynamic adjustable in host mode.
- SOF pulse supports output to pad.
- Supports detecting ID pin level and VBUS voltage.
- Needs external component to supply power for connected USB device in host mode or OTG A-device mode.

## 21.3. Block diagram

Figure 21-1. USBFS block diagram



## 21.4. Signal description

Table 21-1. USBFS signal description

I/O port	Type	Description
VBUS	Input	Bus power port
DM	Input/Output	Differential D-
DP	Input/Output	Differential D+
ID	Input	USB identification: Mini connector identification port

## 21.5. Function overview

### 21.5.1. USBFS clocks and working modes

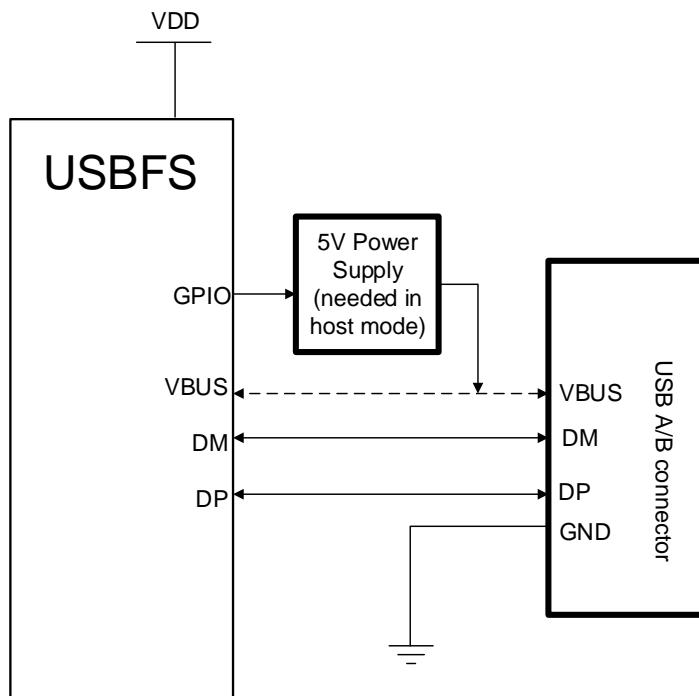
USBFS can operate as a host, a device or a DRD (Dual-role-Device), it contains an internal full-speed PHY. The maximum speed supported by USBFS is full-speed.

The internal PHY supports Full-Speed and Low-Speed in host mode, supports Full-speed in device mode, and supports OTG mode with HNP and SRP. The USB clock used by the USBFS should be 48MHz. The 48MHz USB clock is generated from internal clocks in system, and its source and divider factors are configurable in RCU.

The pull-up and pull-down resistors have already been integrated into the internal PHY and they could be controlled by USBFS automatically according to the current mode (host, device or OTG mode) and connection status. A typical connection is shown in [Figure 21-2](#).

[Connection with host or device mode](#)

**Figure 21-2. Connection with host or device mode**

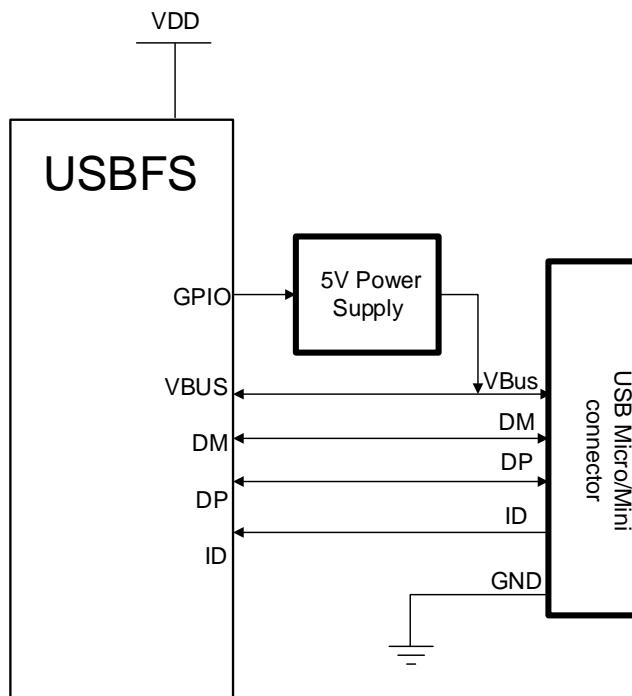


When USBFS works in host mode (FHM bit is set and FDM bit is cleared), the VBUS is 5V power detecting pin used for voltage detection defined in USB protocol. The internal PHY cannot supply 5V VBUS power and only has some voltage comparers, charge and dis-charge circuits on VBUS line. So if application needs VBUS power, an external power supply IC is needed. The VBUS connection between USBFS and the USB connector can be omitted in host mode, so USBFS doesn't detect the voltage level on VBUS pin and always assumes that the 5V power is present.

When USBFS works in device mode (FHM bit is cleared and FDM bit is set), the VBUS detection circuit is connected to a GPIO pin. USBFS continuously monitor the VBUS voltage by the GPIO pin and will immediately switch on the pull-up resistor on DP line once that the VBUS voltage rise above the needed valid value. This will cause a connection. If the VBUS voltage falls below the needed valid value, the pull-up resistor on DP line will be switched off and a disconnection will happen.

The OTG mode connection is described in the [Figure 21-3. Connection with OTG mode](#). When USBFS works in OTG mode, the FHM, FDM bits in USBFS\_GUSBCS and VBUSIG bit in USBFS\_GCCFG should be cleared. In this mode, the USBFS needs all the four pins: DM, DP, VBUS and ID, and needs to use several voltage comparers to monitor the voltage on these pins. USBFS also contains VBUS charge and discharge circuits to perform SRP request described in OTG protocol. The OTG A-device or B-device is decided by the level of ID pins. USBFS controls the pull-up or pull-down resistor during performing the HNP protocol.

Figure 21-3. Connection with OTG mode

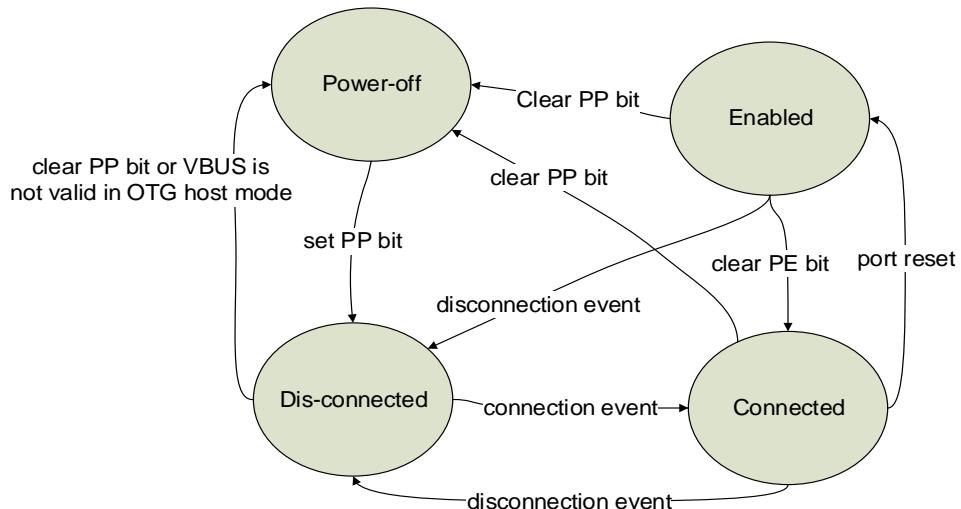


### 21.5.2. USB host function

#### USB Host Port State

Host application may control state of the USB port via USBFS\_HPCS register. After system initialization, the USB port stays at power-off state. After PP bit is set by software, the internal USB PHY is powered on, and the USB port changes into disconnected state. After a connection is detected, USB port changes into connected state. The USB port changes into enabled state after a port reset is performed on USB bus.

Figure 21-4. State transition diagram of host port



### Connection, Reset and Speed identification

As a USB host, USBFS will trigger a connection flag for application after a connection is detected and will trigger a disconnection flag after a disconnection event.

PRST bit is used for USB reset sequence. Application may set this bit to start a USB reset and clear this bit to finish the USB reset. This bit only takes effect when port is at connected or enabled state.

The USBFS performs speed identification during connection, and the speed information will be reported in PS filed in USBFS\_HPCS register. USBFS identifies the device speed by the voltage level of DM or DP. As described in USB protocol, full-speed device pulls up DP line while low-speed device pulls up DM line.

### Suspend and resume

USBFS supports suspend state and resume operation. When USBFS port is at enabled state, writing 1 to PSP bit in USBFS\_HPCS register will cause USBFS to enter suspend state. In suspend state, USBFS stops sending SOFs on USB bus and this will cause the connected USB device to enter suspend state after 3ms. Application can set the PREM bit in USBFS\_HPCS register to start a resume sequence to wake up the suspended device and clear this bit to stop the resume sequence. The WKUPIF bit in USBFS\_GINTF will be set and the USBFS wake up interrupt will be triggered if a host in suspend state detects a remote wakeup signal.

### SOF generate

USBFS sends SOF tokens on USB bus in host mode. As described in USB 2.0 protocol, SOF packets are generated (by the host controller or hub transaction translator) every 1ms in full-speed links.

Each time after USBFS enters into enabled state, it will send the SOF packet periodically which the time is defined in USB 2.0 protocol. In addition, application may adjust the length of a frame by writing FRI filed in USBFS\_HFT registers. The FRI bits define the number of USB clock cycles in a frame, so its value should be calculated based on the frequency of USB clock which is used by USBFS. The FRT filed bits show that the remaining clock cycles of the current frame and stop changing during suspend state.

USBFS is able to generate a pulse signal each SOF packet and output it to a pin. The pulse length is 16 HCLK cycle. If application desires to use this function, it needs to set SOFOEN bit in USBFS\_GCCFG register and configure the related pin registers in GPIO.

### USB Channels and Transactions

USBFS includes 8 independent channels in host mode. Each channel is able to communicate with an endpoint in USB device. The transfer type, direction, packet length and other information are all configured in channel related registers such as USBFS\_HCHxCTL and USBFS\_HCHxLEN.

USBFS supports all the four kinds of transfer types: control, bulk, interrupts and isochronous.

USB 2.0 protocol divides these transfers into 2 kinds: non-periodic transfer (control and bulk) and periodic transfer (interrupt and isochronous). Based on this, USBFS includes two request queues: periodic request queue and non-periodic request queue, to perform efficient transaction schedule. A request entry in a request queue described above may represent a USB transaction request or a channel operation request.

Application needs to write packet into data FIFO via AHB register interface if it wants to start an OUT transaction on USB bus. USBFS hardware will automatically generate a transaction request entry in request queue after the application writes a whole packet.

The request entries in request queue are processed in order by transaction control module. USBFS always tries to process periodic request queue firstly and secondly process non-periodic request queue.

After a start of frame, USBFS begins to process periodic queue until the queue is empty or bus time required by the current periodic request is not enough, and then process the non-periodic queue. This strategy ensures the bandwidth of periodic transactions in a frame. Each time the USBFS reads and pops a request entry from request queue. If this is a channel disable request, it immediately disables the channel and prepares to process the next entry.

If the current request is a transaction request and the USB bus time is enough for this transaction, USBFS will employ SIE to generate this transaction on USB bus.

When the required bus time for the current request is not enough in the current frame, and if this is a periodic request, USBFS stops processing the periodic queue and starts to process non-periodic request. If this is a non-periodic queue the USBFS will stop processing any queue and wait until the end of current frame.

### 21.5.3. USB device function

#### USB Device Connection

In device mode, USBFS stays at power-off state after initialization. After connecting to a USB host with 5V power supply through VBUS pin, USBFS enters into powered state. USBFS begins to switch on the pull-up resistor on DP line and thus, host side will detect a connection event.

**Note:** the VBUS pin must be connected to the PA9 for detecting the level.

#### Reset and Speed-Identification

The USB host always starts a USB reset when it detects a device connection, and USBFS in device mode will trigger a reset interrupt by hardware when it detects the reset event on USB bus.

After reset sequence, USBFS will trigger an ENUMF interrupt in USBFS\_GINTF register and reports current enumerated device speed in ES bits in USBFS\_DSTAT register, this bit field is always 11(full-speed).

As required by USB 2.0 protocol, USBFS doesn't support low-speed in device mode.

### Suspend and Wake-up

A USB device will enter into suspend state when the USB bus stays at IDLE state and there is no change on data lines for 3ms. When USB device is in suspend state, most of its clock are closed to save power. The USB host is able to wake up the suspended device by generating a resume signal on USB bus. When USBFS detects the resume signal, the WKUPIF flag in USBFS\_GINTF register will be set and the USBFS wake up interrupt will be triggered.

In suspend mode, USBFS is also able to remotely wake up the USB bus. Software may set RWKUP bit in USBFS\_DCTL register to send a remote wake-up signal, and if remote wake-up is supported in USB host, the host will begin to send resume signal on USB bus.

### Soft Disconnection

USBFS supports soft disconnection. After the device is powered on, USBFS will switch on the pull-up resistor on DP line so that the host can detect the connection. It is able to force a disconnection by setting the SD bit in USBFS\_DCTL register. After the SD bit is set, USBFS will directly switch off the pull-up resistor, so that USB host will detect a disconnection on USB bus.

### SOF tracking

When USBFS receives a SOF packet on USB bus, it will trigger a SOF interrupt and begin to count the bus time using local USB clock. The frame number of the current frame is reported in FNRSOF filed in USBFS\_DSTAT register. When the USB bus time reaches EOF1 or EOF2 point (End of Frame, described in USB 2.0 protocol), USBFS will trigger an EOPFIF interrupt in USBFS\_GINTF register. These flags and registers can be used to get current bus time and position information.

## 21.5.4. OTG function overview

USBFS supports OTG function described in OTG protocol 1.3, OTG function includes SRP and HNP protocols.

### A-Device and B-Device

A-Device is an OTG capable USB device with a Standard-A or Micro-A plug inserted into its receptacle. The A-Device supplies power to VBUS and it is host at the start of a session. B-Device is an OTG capable USB device with a Standard-B, Micro-B or Mini-B plug inserted into its receptacle, or a captive cable ending being a Standard-A plug. The B-Device is a peripheral at the start of a session. USBFS uses the voltage level of ID pin to identify A-Device or B-Device. The ID status is reported in IDPS bit in USBFS\_GOTGCS register. For the details of transfer states between A-Device and B-Device, please refer to OTG 1.3 protocol.

### HNP

The Host Negotiation Protocol (HNP) allows the host function to be switched between two directly connected On-The-Go devices and eliminates the necessity of switching the cable connections for the change of control of communications between the devices. HNP will be initialized typically by the user or an application on the On-The-Go B-Device. HNP may only be implemented through the Micro-AB receptacle on a device.

Since On-The-Go devices have a Micro-AB receptacle, an On-The-Go device can default to being either a host or a device, depending that which type of plug (Micro-A plug for host, Micro-B plug for device) is inserted. By utilizing the Host Negotiation Protocol (HNP), an On-The-Go B-Device, which is the default device, may make a request to be a host. The process for the exchange of the role to a host is described in this section. This protocol eliminates the necessity of switching the cable connection for the change of the roles of the connected devices.

When USBFS is in OTG A-Device host mode and it wants to give up its host role, it may firstly set PSP bit in USBFS\_HPCS register to make the USB bus enter in suspend status. Then, the B-Device will enter in suspend state 3ms later. If the B-Device wants to change to be a host, HNPREQ bit in USBFS\_GOTGCS register should be set and the USBFS will begin to perform HNP protocol on bus, and at last, the result of HNP is reported in HNPS bit in USBFS\_GOTGCS register. Besides, it is always available to get the current role (host or device) from COPM bit in USBFS\_GINTF register.

### SRP

The Session Request Protocol (SRP) allows a B-Device to request the A-Device to turn on VBUS and start a session. This protocol allows the A-Device, which may be battery powered, to conserve power by turning VBUS off when there is no bus activity while still providing a means for the B-Device to initiate bus activity. As described in OTG protocol, an OTG device must compare VBUS voltage with several threshold values and the compare result should be reported in ASV and BSV bits in USBFS\_GOTGCS register.

Set SRPREQ bit in USBFS\_GOTGCS register to start a SRP request when USBFS is in B-Device OTG mode and USBFS will generate a success flag SRPS in USBFS\_GOTGCS register if the SRP request successfully.

When USBFS is in OTG A-Device mode and it has detected a SRP request from a B-Device, it sets a SESIF flag in USBFS\_GINTF register. The 5V power supply for VBUS pin should be prepared to switch on after getting this flag.

## 21.5.5. Data FIFO

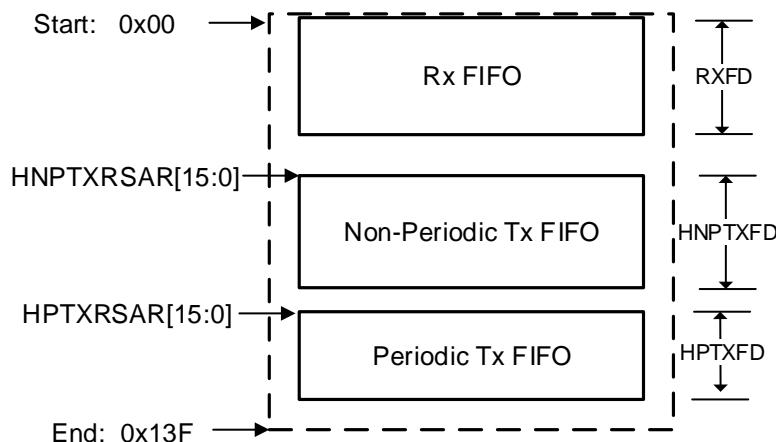
The USBFS contains a 1.25K bytes data FIFO for packet data storage. The data FIFO is implemented by using an internal SRAM in USBFS.

### Host Mode

In host mode, the data FIFO space is divided into 3 parts: Rx FIFO for received packet, Non-Periodic Tx FIFO for non-period transmission packet and Periodic Tx FIFO for periodic

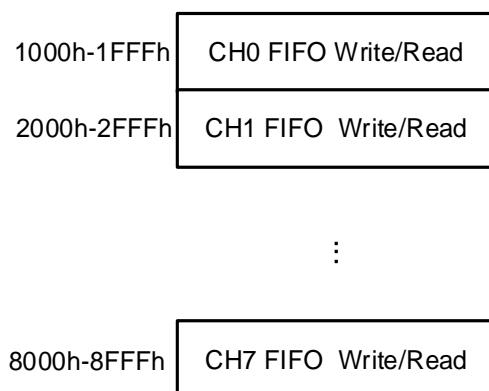
transmission packet. All IN channels shares the Rx FIFO for packets reception. All the periodic OUT channels share the periodic Tx FIFO to packets transmission. All the non-periodic OUT channels share the non-Periodic Tx FIFO for transmit packets. The size and start offset of these data FIFOs should be configured using these registers: USBFS\_GRFLEN, USBFS\_HNPTFLEN and USBFS\_HPTFLEN. [Figure 21-5. HOST mode FIFO space in SRAM](#) describes the structure of these FIFOs in SRAM. The values in the figure are in term of 32-bit words.

**Figure 21-5. HOST mode FIFO space in SRAM**



USBFS provides a special register area for the internal data FIFO reading and writing. [Figure 21-6. Host mode FIFO access register map](#) describes the register memory area that the data FIFO can write. This area can be read by any channel data FIFO. The addresses in the figure are addressed in bytes. Each channel has its own FIFO access register space, although all Non-periodic channels share the same FIFO and all the Periodic channels also share the same FIFO. It is important for USBFS to know which channel the current pushed packet belongs to. Rx FIFO is also able to be accessed using USBFS\_GRSTATR/USBFS\_GRSTATP register.

**Figure 21-6. Host mode FIFO access register map**

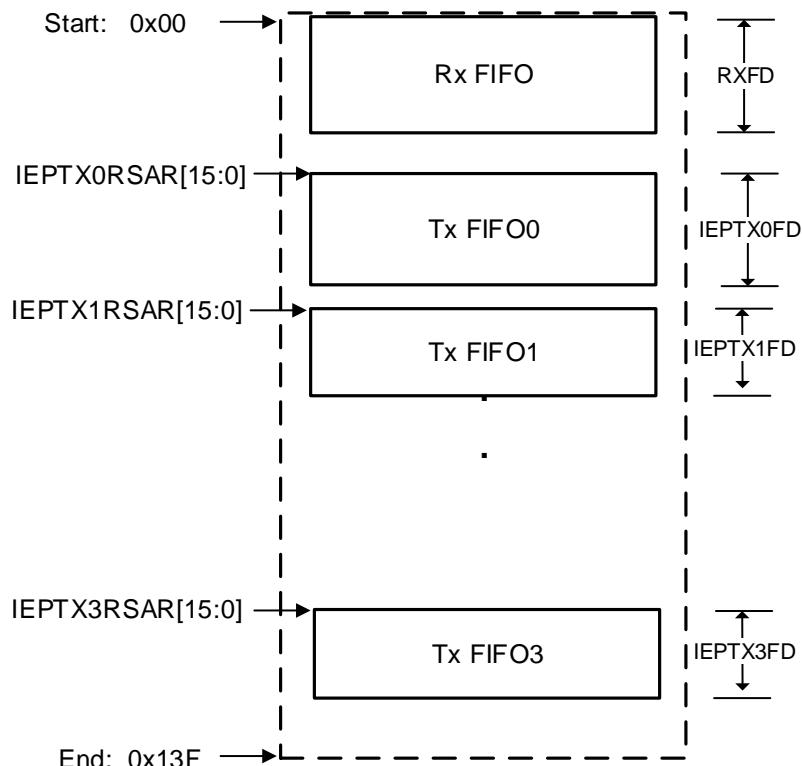


#### Device mode

In device mode, the data FIFO is divided into several parts: one Rx FIFO, and 4 Tx FIFOs

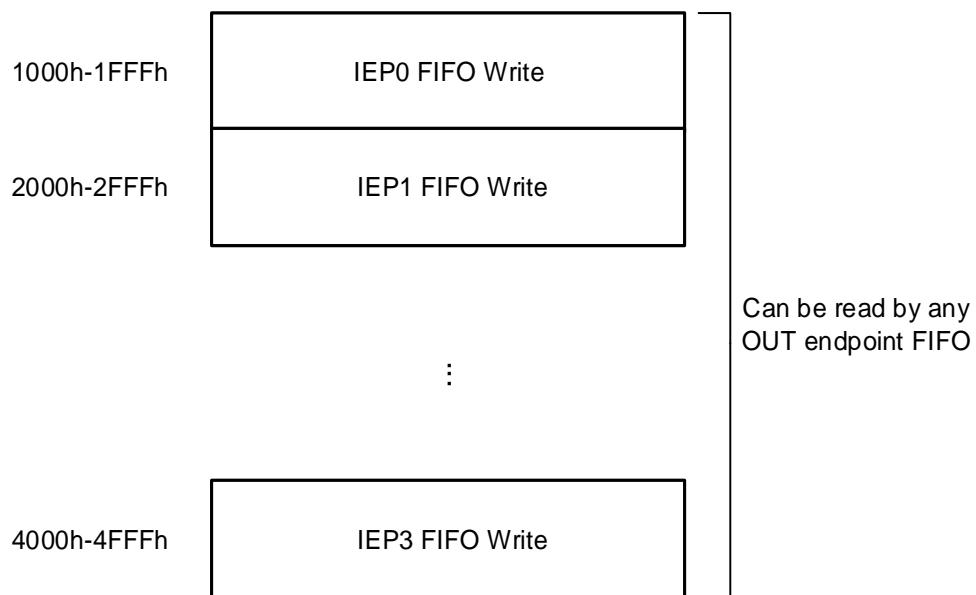
(one for each IN endpoint). All the OUT endpoints share the Rx FIFO for receiving packets. The size and start offset of these data FIFOs should be configured using USBFS\_GRFLEN and USBFS\_DIEPxTFLEN ( $x=0\dots3$ ) registers. [Figure 21-7. Device mode FIFO space in SRAM](#) describes the structure of these FIFOs in SRAM. The values in the figure are in term of 32-bit words.

**Figure 21-7. Device mode FIFO space in SRAM**



USBFS provides a special register area for the internal data FIFO reading and writing. [Figure 21-8. Device mode FIFO access register map](#) describes the register memory area where the data FIFO can write. This area can be read by any endpoint FIFO. The addresses in the figure are addressed in bytes. Each endpoint has its own FIFO access register space. Rx FIFO is also able to be accessed using USBFS\_GRSTATR/USBFS\_GRSTATP register.

Figure 21-8. Device mode FIFO access register map



### 21.5.6. Operation guide

This section describes the advised operation guide for USBFS.

#### Host mode

##### Global register initialization sequence

1. Program USBFS\_GAHBCS register according to application's demand, such as the TxFIFO's empty threshold, etc. GINTEN bit should be kept cleared at this time.
2. Program USBFS\_GUSBCS register according to application's demand, such as the operation mode (host, device or OTG) and some parameters of OTG and USB protocols.
3. Program USBFS\_GCCFG register according to application's demand.
4. Program USBFS\_GRFLEN, USBFS\_HNPTFLEN\_DIEP0TFLEN and USBFS\_HPTFLEN register to configure the data FIFOs according to application's demand.
5. Program USBFS\_GINTEN register to enable Mode Fault and Host Port interrupt and set GINTEN bit in USBFS\_GAHBCS register to enable global interrupt.
6. Program USBFS\_HPCS register and set PP bit.
7. Wait for a device's connection, and once a device is connected, the connection interrupt PCD in USBFS\_HPCS register will be triggered. Then set PRST bit to perform a port reset. Wait for at least 10ms and then clear PRST bit.
8. Wait PEDC interrupt in USBFS\_HPCS register and then read PE bit to ensure that the port is successfully enabled. Read PS [1:0] bits to get the connected device's speed and then program USBFS\_HFT register to change the SOF interval if needed.

### Channel initialization and enable sequence

1. Program USBFS\_HCHxCTL registers with desired transfer type, direction, packet size, etc. Ensure that CEN and CDIS bits keep cleared during configuration.
2. Program USBFS\_HCHxINTEN register. Set the desired interrupt enable bits.
3. Program USBFS\_HCHxLEN register. PCNT is the number of packets in a transfer and TLEN is the total bytes number of all the transmitted or received packets in a transfer.

For OUT channel: If PCNT=1, the single packet's size is equal to TLEN. If PCNT>1, the former PCNT-1 packets are considered as max-packet-length packets whose size are defined by MPL field in USBFS\_HCHxCTL register, and the last packet's size is calculated based on PCNT, TLEN and MPL. If software wants to send out a zero-length packet, it should program TLEN=0, PCNT=1.

For IN channel: Because the application doesn't know the actual received data size before the IN transaction finishes, TLEN can be set to a maximum possible value supported by Rx FIFO.

4. Set CEN bit in USBFS\_HCHxCTL register to enable the channel.

### Channel disable sequence

Software can disable the channel by setting both CEN and CDIS bits at the same time. USBFS will generate a channel disable request entry in request queue after the register setting operation. When the request entry reaches the top of request queue, it is processed by USBFS immediately:

For OUT channels, the specified channel will be disabled immediately. Then, a CH flag will be generated and the CEN and CDIS bits will be cleared by USBFS.

For IN channels, USBFS pushes a channel disable status entry into Rx FIFO. Software should then handle the Rx FIFO not empty event: read and pop this status entry, then, a CH flag will be generated and the CEN and CDIS bits will be cleared.

### IN transfers operation sequence

1. Initialize USBFS global registers.
2. Initialize the channel.
3. Enable the channel.
4. After the IN channel is enabled by software, USBFS generates an Rx request entry in the corresponding request queue.
5. When the Rx request entry reaches the top of the request queue, USBFS begins to process this request entry. If bus time for the IN transaction indicated by the request entry is enough, USBFS starts the IN transaction on USB bus.
6. If the IN transaction finishes successfully (ACK handshake received), USBFS pushes the

received data packet into the Rx FIFO and triggers ACK flag. Otherwise, the status flag (NAK) reports the transaction result.

7. If the IN transaction described in step 5 is successful and PCNT is larger than 1 in step2, return to step 3 and continues to receive the remaining packets. If the IN transaction described in step 5 is not successful, return to step 3 to re-receive the packet again.
8. After all the transactions in a transfer are successfully received on USB bus, USBFS pushes a TF status entry into the Rx FIFO on top of the last packet data. Thus after reading and popping all the received data packet, the TF status entry is need, USBFS generates TF flag to indicate that the transfer successfully finishes.
9. Disable the channel. Now the channel is in IDLE state and is ready for other transfers.

### **OUT transfers operation sequence**

1. Initialize USBFS global registers.
2. Initialize and enable the channel.
3. Write a packet into the channel's Tx FIFO (Periodic Tx FIFO or non-periodic Tx FIFO). After the whole packet data is written into the FIFO, USBFS generates a Tx request entry in the corresponding request queue and decreases the TLEN field in USBFS\_HCHxLEN register by the written packet's size.
4. When the request entry reaches the top of the request queue, USBFS begins to process this request entry. If bus time for the transaction indicated by the request entry is enough, USBFS starts the OUT transaction on USB bus.
5. When the OUT transaction indicated by the request entry finishes on USB bus, PCNT in USBFS\_HCHxLEN register is decreased by 1. If the transaction finishes successfully (ACK handshake received), the ACK flag is triggered. Otherwise, the status flag (NAK) reports the transaction result.
6. If the OUT transaction described in step 5 is successful and PCNT is larger than 1 in step2, return to step 3 and continues to send the remaining packets. If the OUT transaction described in step 5 is not successful, return to step 3 to resend the packet again.
7. After all the transactions in a transfer are successfully sent on USB bus, USBFS generates TF flag to indicate that the transfer successfully finishes.
8. Disable the channel. Now the channel is in IDLE state and is ready for other transfers.

## **Device mode**

### **Global register initialization sequence**

1. Program USBFS\_GAHBCS register according to application's demand, such as the TxFIFO's empty threshold, etc. GINTEN bit should be kept cleared at this time.

2. Program USBFS\_GUSBCS register according to application's demand, such as: the operation mode (host, device or OTG) and some parameters of OTG and USB protocols.
3. Program USBFS\_GCCFG register according to application's demand.
4. Program USBFS\_GRFLEN, USBFS\_HNPTFLEN\_DIEP0TFLEN, USBFS\_DIEPxTFLEN register to configure the data FIFOs according to application's demand.
5. Program USBFS\_GINTEN register to enable Mode Fault, Suspend, SOF, Enumeration Done and USB Reset interrupt and then, set GINTEN bit in USBFS\_GAHBCS register to enable global interrupt.
6. Program USBFS\_DCFG register according to application's demand, such as the device address, etc.
7. After the device is connected to a host, the host will perform port reset on USB bus and this will trigger the RST interrupt in USBFS\_GINTF register.
8. Wait for ENUMF interrupt in USBFS\_GINTF register.

#### Endpoint initialization and enable sequence

1. Program USBFS\_DIEPxCTL or USBFS\_DOEPxCTL register with desired transfer type, packet size, etc.
2. Program USBFS\_DIEPINTEN or USBFS\_DOEPINTEN register. Set the desired interrupt enable bits.
3. Program USBFS\_DIEPxLEN or USBFS\_DOEPxLEN register. PCNT is the number of packets in a transfer and TLEN is the total bytes number of all the transmitted or received packets in a transfer.

For IN endpoint: If PCNT=1, the single packet's size is equal to TLEN. If PCNT>1, the former PCNT-1 packets are considered as max-packet-length packets whose size are defined by MPL field in USBFS\_DIEPxCTL register, and the last packet's size is calculated based on PCNT, TLEN and MPL. If a zero-length packet is required to be sent, it should program TLEN=0, PCNT=1.

For OUT endpoint: Because the application doesn't know the actual received data size before the OUT transaction finishes, TLEN can be set to a maximum possible value supported by Rx FIFO.

4. Set EPEN bit in USBFS\_DIEPxCTL or USBFS\_DOEPxCTL register to enable the endpoint.

#### Endpoint disable sequence

The endpoint can be disabled anytime when the EPEN bit in USBFS\_DIEPxCTL or USBFS\_DOEPxCTL registers is cleared.

#### IN transfers operation sequence

1. Initialize USBFS global registers.
2. Initialize and enable the IN endpoint.
3. Write packets into the endpoint's Tx FIFO. Each time a data packet is written into the FIFO, USBFS decreases the TLEN field in USBFS\_DIEPxLEN register by the written packet's size.
4. When an IN token received, USBFS transmits the data packet, and after the transaction finishes on USB bus, PCNT in USBFS\_DIEPxLEN register is decreased by 1. If the transaction finishes successfully (ACK handshake received), the ACK flag is triggered. Otherwise, the status flags reports the transaction result.
5. After all the data packets in a transfer are successfully sent on USB bus, USBFS generates TF flag to indicate that the transfer successfully finishes and disables the IN endpoint.

#### **OUT transfers operation sequence**

1. Initialize USBFS global registers.
2. Initialize the endpoint and enable the endpoint.
3. When an OUT token received, USBFS receives the data packet or response with an NAK handshake based on the status of Rx FIFO and register configuration. If the transaction finishes successfully (USBFS receives and saves the data packet into Rx FIFO successfully and sends ACK handshake on USB bus), PCNT in USBFS\_DOEPxLEN register is decreased by 1 and the ACK flag is triggered, otherwise, the status flags report the transaction result.
4. After all the data packets in a transfer are successfully received on USB bus, USBFS pushes a TF status entry into the Rx FIFO on top of the last packet data. Thus after reading and popping all the received data packet, the TF status entry is read, USBFS generates TF flag to indicate that the transfer successfully finishes and disables the OUT endpoint.

## **21.6. Interrupts**

USBFS has two interrupts: global interrupt and wake-up interrupt.

The source flags of the global interrupt are readable in USBFS\_GINTF register and are listed in [Table 21-2. USBFS global interrupt](#).

**Table 21-2. USBFS global interrupt**

Interrupt Flag	Description	Operation Mode
SEIF	Session interrupt	Host or device mode
DISCIF	Disconnect interrupt flag	Host Mode
IDPSC	ID pin status change	Host or device mode

Interrupt Flag	Description	Operation Mode
PTXFEIF	Periodic Tx FIFO empty interrupt flag	Host Mode
HCIF	Host channels interrupt flag	Host Mode
HPIF	Host port interrupt flag	Host Mode
ISOONCIF/PXNCI F	Periodic transfer Not Complete Interrupt flag /Isochronous OUT transfer Not Complete Interrupt Flag	Host or device mode
ISOINCIF	Isochronous IN transfer Not Complete Interrupt Flag	Device mode
OEIF	OUT endpoint interrupt flag	Device mode
IEEIF	IN endpoint interrupt flag	Device mode
EOPFIF	End of periodic frame interrupt flag	Device mode
ISOOPDIF	Isochronous OUT packet dropped interrupt flag	Device mode
ENUMF	Enumeration finished	Device mode
RST	USB reset	Device mode
SP	USB suspend	Device mode
ESP	Early suspend	Device mode
GONAK	Global OUT NAK effective	Device mode
GNPINAK	Global IN Non-Periodic NAK effective	Device mode
NPTXFEIF	Non-Periodic Tx FIFO empty interrupt flag	Host Mode
RXFNEIF	Rx FIFO non-empty interrupt flag	Host or device mode
SOF	Start of frame	Host or device mode
OTGIF	OTG interrupt flag	Host or device mode
MFIF	Mode fault interrupt flag	Host or device mode

Wake-up interrupt can be triggered when USBFS is in suspend state, even when the USBFS's clocks are stopped. The source of the wake-up interrupt is WKUPIF bit in USBHS\_GINTF register.

## 21.7. Register definition

USBFS base address: 0x5000 0000

### 21.7.1. Global control and status registers

#### Global OTG control and status register (USBFS\_GOTGCS)

Address offset: 0x0000

Reset value: 0x0000 0800

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												BSV	ASV	DI	IDPS
r	r	r	r												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				DHNPEN	HNPEN	HNPREQ	HNPSS	Reserved				SRPREQ	SRPSS		
				rw	rw	rw	r					rw			r

Bits	Fields	Descriptions
31:20	Reserved	Must be kept at reset value
19	BSV	B-Session Valid (described in OTG protocol). 0: Vbus voltage level of a OTG B-Device is below VBSESSVLD 1: Vbus voltage level of a OTG B-Device is above VBSESSVLD Note: Only accessible in OTG B-Device mode.
18	ASV	A- Session valid A-host mode transceiver status. 0: Vbus voltage level of a OTG A-Device is below VASESSVLD 1: Vbus voltage level of a OTG A-Device is above VASESSVLD The A-Device is the default host at the start of a session. Note: Only accessible in OTG A-Device mode.
17	DI	Debounce interval Debounce interval of a detected connection. 0: Indicates the long debounce interval , when a plug-on and connection occurs on USB bus 1: Indicates the short debounce interval, when a soft connection is used in HNP

		protocol.
		Note: Only accessible in host mode.
16	IDPS	<p>ID pin status</p> <p>Voltage level of connector ID pin</p> <p>0: USBFS is in A-Device mode</p> <p>1: USBFS is in B-Device mode</p> <p>Note: Accessible in both device and host modes.</p>
15:12	Reserved	Must be kept at reset value
11	DHNPEN	<p>Device HNP enable</p> <p>Enable the HNP function of a B-Device. If this bit is cleared, USBFS doesn't start HNP protocol when application set HNPREQ bit in USBFS_GOTGCS register.</p> <p>0: HNP function is not enabled.</p> <p>1: HNP function is enabled</p> <p>Note: Only accessible in device mode.</p>
10	HNPEN	<p>Host HNP enable</p> <p>Enable the HNP function of an A-Device. If this bit is cleared, USBFS doesn't response to the HNP request from B-Device.</p> <p>0: HNP function is not enabled.</p> <p>1: HNP function is enabled</p> <p>Note: Only accessible in host mode.</p>
9	HNPREQ	<p>HNP request</p> <p>This bit is set by software to start a HNP on the USB. This bit can be cleared when HNPEND bit in USBFS_GOTGINTF register is set, by writing zero to it, or clearing the HNPEND bit in USBFS_GOTGINTF register.</p> <p>0: Don't send HNP request</p> <p>1: Send HNP request</p> <p>Note: Only accessible in device mode.</p>
8	HNPS	<p>HNP successes</p> <p>This bit is set by the core when HNP succeeds, and this bit is cleared when HNPREQ bit is set.</p> <p>0: HNP fails</p> <p>1: HNP succeeds</p> <p>Note: Only accessible in device mode.</p>
7:2	Reserved	Must be kept at reset value
1	SRPREQ	<p>SRP request</p> <p>This bit is set by software to start a SRP on the USB. This bit can be cleared when SRPEND bit in USBFS_GOTGINTF register is set, by writing zero to it, or clearing the SRPEND bit in USBFS_GOTGINTF register.</p> <p>0: No session request</p> <p>1: Session request</p>

Note: Only accessible in device mode.

0	SRPS	SRP success
		This bit is set by the core when SRP succeeds, and this bit is cleared when SRPREQ bit is set.
	0: SRP fails	
	1: SRP succeeds	

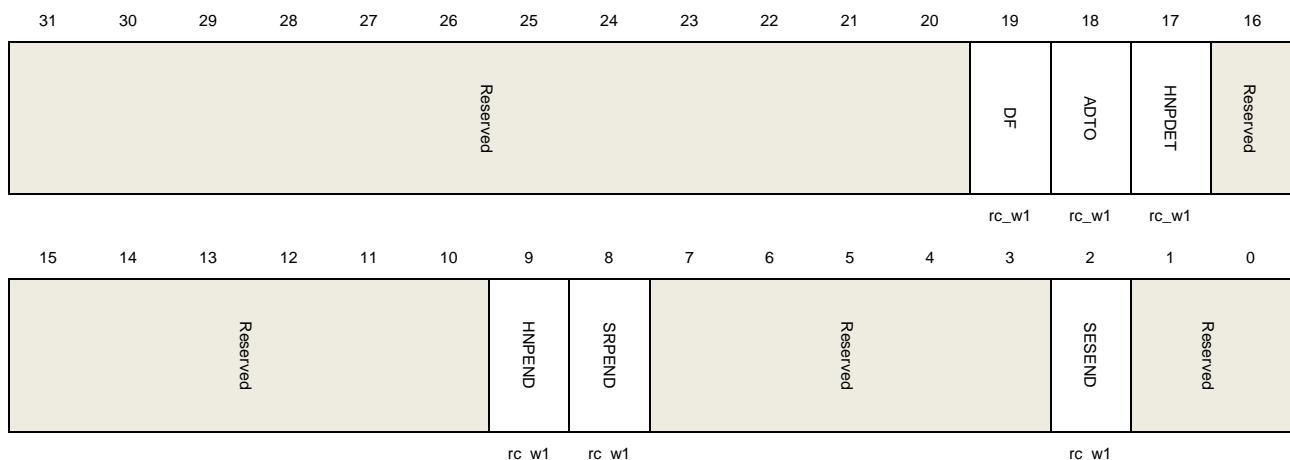
Note: Only accessible in device mode.

### Global OTG interrupt flag register (USBFS\_GOTGINTF)

Address offset: 0x0004

Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31:20	Reserved	Must be kept at reset value
19	DF	Debounce finish Set by USBFS when the debounce during device connection is done. Note: Only accessible in host mode.
18	ADTO	A-Device timeout Set by USBFS when the A-Device's waiting for a B-Device' connection has timed out. Note: Accessible in both device and host modes.
17	HNPDET	Host negotiation request detected Set by USBFS when A-Device detects a HNP request. Note: Accessible in both device and host modes.
16:10	Reserved	Must be kept at reset value
9	HNPEND	HNP end

Set by the core when a HNP ends. Read the HNPS in USBFS\_GOTGCS register to get the result of HNP.

Note: Accessible in both device and host modes.

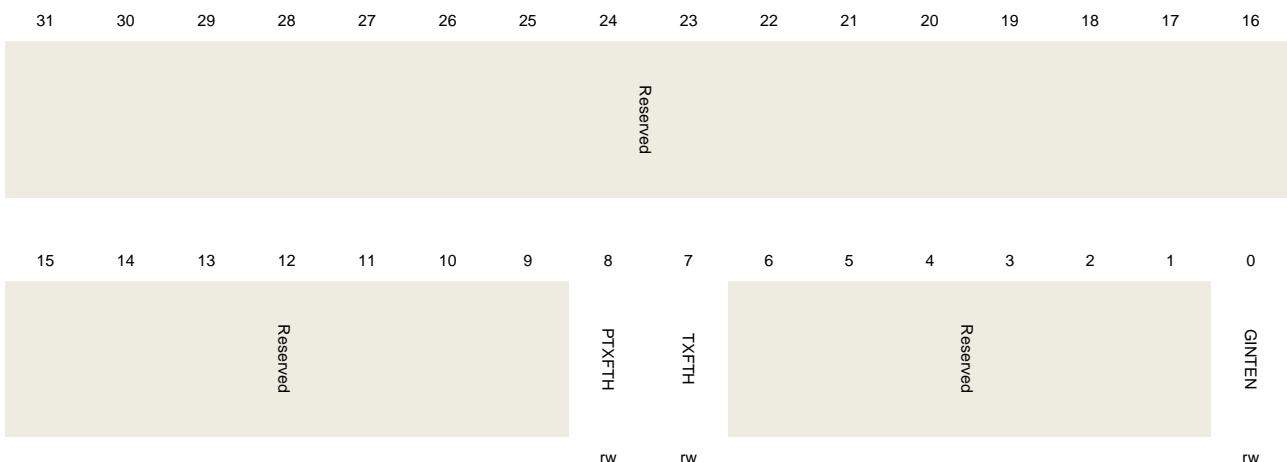
8	<b>SRPEND</b>	SRPEND
		Set by the core when a SRP ends. Read the SRPS in USBFS_GOTGCS register to get the result of SRP.
		Note: Accessible in both device and host modes.
7:3	<b>Reserved</b>	Must be kept at reset value
2	<b>SESEND</b>	Session end
		Set by the core when VBUS voltage is below Vb_ses_vld.
1:0	<b>Reserved</b>	Must be kept at reset value

### Global AHB control and status register (USBFS\_GAHBCS)

Address offset: 0x0008

Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31:9	Reserved	Must be kept at reset value
8	<b>PTXFTH</b>	Periodic Tx FIFO threshold 0: PTXFEIF will be triggered when the periodic transmit FIFO is half empty 1: PTXFEIF will be triggered when the periodic transmit FIFO is completely empty Note: Only accessible in host mode.
7	<b>TXFTH</b>	Tx FIFO threshold Device mode: 0: TXFEIF will be triggered when the IN endpoint transmit FIFO is half empty 1: TXFEIF will be triggered when the IN endpoint transmit FIFO is completely empty

**Host mode:**

0: NPTXFEIF will be triggered when the non-periodic transmit FIFO is half empty  
 1: NPTXFEIF will be triggered when the non-periodic transmit FIFO is completely empty

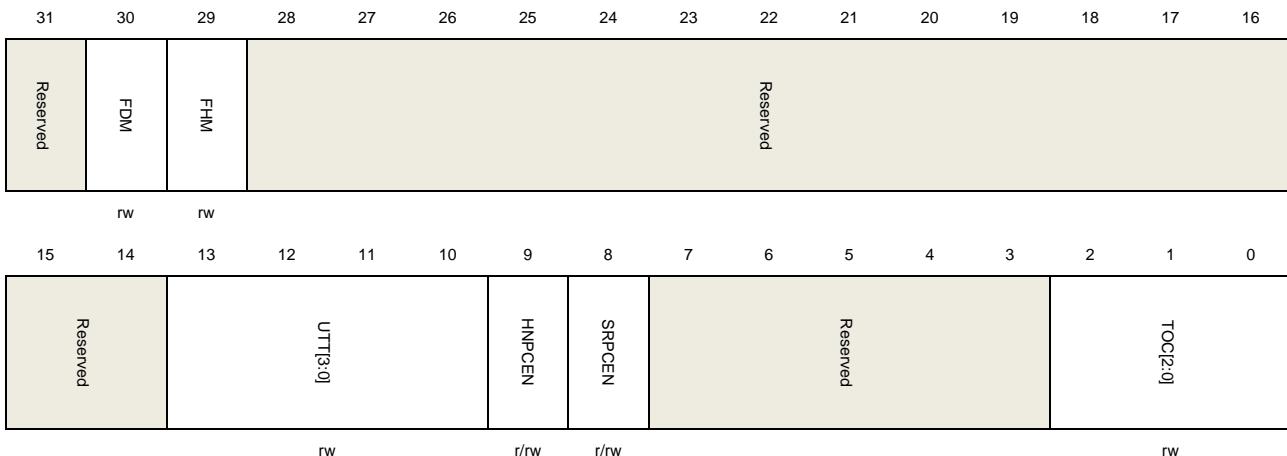
6: 1	Reserved	Must be kept at reset value
0	GINTEN	Global interrupt enable 0: Global interrupt is not enabled. 1: Global interrupt is enabled. Note: Accessible in both device and host modes.

**Global USB control and status register (USBFS\_GUSBCS)**

Address offset: 0x000C

Reset value: 0x0000 0A80

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



Bits	Fields	Descriptions
31	Reserved	Must be kept at reset value
30	FDM	Force device mode Setting this bit will force the core to device mode irrespective of the USBFS ID input pin. 0: Normal mode 1: Device mode The application must wait at least 25 ms for the change taking effect after setting the force bit. Note: Accessible in both device and host modes.
29	FHM	Force host mode Setting this bit will force the core to host mode irrespective of the USBFS ID input pin.

		0: Normal mode 1: Host mode The application must wait at least 25 ms for the change taking effect after setting the force bit. Note: Accessible in both device and host modes.
28:14	Reserved	Must be kept at reset value
13:10	UTT[3:0]	USB turnaround time Turnaround time in PHY clocks. Note: Only accessible in device mode.
9	HNP CEN	HNP capability enable Controls whether the HNP capability is enabled 0: HNP capability is disabled 1: HNP capability is enabled Note: Accessible in both device and host modes.
8	SRP CEN	SRP capability enable Controls whether the SRP capability is enabled 0: SRP capability is disabled 1: SRP capability is enabled Note: Accessible in both device and host modes.
7:3	Reserved	Must be kept at reset value
2:0	TOC[2:0]	Timeout calibration USBFS always uses time-out value required in USB 2.0 when waiting for a packet. Application may use TOC [2:0] to add the value is in terms of PHY clock. (The frequency of PHY clock is 48MHZ.).

### Global reset control register (USBFS\_GRSTCTL)

Address offset: 0x0010

Reset value: 0x8000 0000

The application uses this register to reset various hardware features inside the core.

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	TXFNUM[4:0]	TXFF	RXFF	Reserved	HFCRST	HCSRST	CSRST
	rw	rs	rs		rs	rs	rs

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:11	Reserved	Must be kept at reset value
10:6	TXFNUM[4:0]	<p><b>Tx FIFO number</b></p> <p>Indicates which Tx FIFO will be flushed when TXFF bit in the same register is set.</p> <p><b>Host Mode:</b></p> <ul style="list-style-type: none"> <li>00000: Only non-periodic Tx FIFO is flushed</li> <li>00001: Only periodic Tx FIFO is flushed</li> <li>1XXXX: Both periodic and non-periodic Tx FIFOs are flushed</li> <li>Other: Non data FIFO is flushed</li> </ul> <p><b>Device Mode:</b></p> <ul style="list-style-type: none"> <li>00000: Only Tx FIFO0 is flushed</li> <li>00001: Only Tx FIFO1 is flushed</li> <li>...</li> <li>00011: Only Tx FIFO3 is flushed</li> <li>1XXXX: All Tx FIFOs are flushed</li> <li>Other: Non data FIFO is flushed</li> </ul>
5	TXFF	<p><b>Tx FIFO flush</b></p> <p>Application set this bit to flush data Tx FIFOs and TXFNUM[4:0] bits decide the FIFO number to be flushed. Hardware automatically clears this bit after the flush process completes. After setting this bit, application should wait until this bit is cleared before any other operation on USBFS.</p> <p>Note: Accessible in both device and host modes.</p>
4	RXFF	<p><b>Rx FIFO flush</b></p> <p>Application set this bit to flush data Rx FIFO. Hardware automatically clears this bit after the flush process completes. After setting this bit, application should wait until this bit is cleared before any other operation on USBFS.</p> <p>Note: Accessible in both device and host modes.</p>
3	Reserved	Must be kept at reset value
2	HFCRST	<p><b>Host frame counter reset</b></p> <p>Set by the application to reset the frame number counter in USBFS. After this bit is set, the frame number of the following SOF returns to 0. Hardware automatically clears this bit after the reset process completes. After setting this bit, application should wait until this bit is cleared before any other operation on USBFS.</p> <p>Note: Only accessible in host mode.</p>
1	HCSRST	HCLK soft reset

Set by the application to reset AHB clock domain circuit.

Hardware automatically clears this bit after the reset process completes. After setting this bit, application should wait until this bit is cleared before any other operation on USBFS.

Note: Accessible in both device and host modes.

0	CSRST	Core soft reset
		Resets the AHB and USB clock domains circuits, as well as most of the registers.

### Global interrupt flag register (USBFS\_GINTF)

Address offset: 0x0014

Reset value: 0x0400 0021

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
WKUPIF	SESIF	DISCIF	IDPSC	Reserved.	PTXFEIF	HClF	HPIF	Reserved	Reserved	PXNCIF/ ISOONCIF	ISOINCIF	OEIF	IEPIF		Reserved
rc_w1	rc_w1	rc_w1	rc_w1		r	r	r			rc_w1	rc_w1	r	r		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EOPEIF	ISOOPDIF	ENUMF	RST	SP	ESP	Reserved	GONAK	GNPINAK	NPTXFEIF	RXFNEIF	SOF	OTGIF	MIFIF	COPM	
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1			r	r	r	r	rc_w1	rc_w1	rc_w1	r

Bits	Fields	Descriptions
31	WKUPIF	Wakeup interrupt flag  This interrupt is triggered when a resume signal (in device mode) or a remote wakeup signal (in host mode) is detected on the USB.  Note: Accessible in both device and host modes.
30	SESIF	Session interrupt flag  This interrupt is triggered when a SRP is detected (in A-Device mode) or V <sub>BUS</sub> becomes valid for a B- Device (in B-Device mode).  Note: Accessible in both device and host modes.
29	DISCIF	Disconnect interrupt flag  This interrupt is triggered after a device disconnection.  Note: Only accessible in host mode.
28	IDPSC	ID pin status change  Set by the core when ID status changes.

		Note: Accessible in both device and host modes.
27	Reserved	Must be kept at reset value
26	PTXFEIF	<p>Periodic Tx FIFO empty interrupt flag</p> <p>This interrupt is triggered when the periodic transmit FIFO is either half or completely empty. The threshold is determined by the periodic Tx FIFO empty level bit (PTXFTH) in the USBFS_GAHBCS register.</p> <p>Note: Only accessible in host mode.</p>
25	HCIF	<p>Host channels interrupt flag</p> <p>Set by USBFS when one of the channels in host mode has raised an interrupt. First read USBFS_HACHINT register to get the channel number, and then read the corresponding USBFS_HCHxINTF register to get the flags of the channel that cause the interrupt. This bit will be automatically cleared after the respective channel's flags which cause channel interrupt are cleared.</p> <p>Note: Only accessible in host mode.</p>
24	HPIF	<p>Host port interrupt flag</p> <p>Set by the core when USBFS detects that port status changes in host mode. Software should read USBFS_HPCS register to get the source of this interrupt. This bit will be automatically cleared after the flags that causing a port interrupt are cleared.</p> <p>Note: Only accessible in host mode.</p>
23:22	Reserved	Must be kept at reset value
21	PXNCIF	<p>Periodic transfer Not Complete Interrupt flag</p> <p>USBFS sets this bit when there are periodic transactions for current frame not completed at the end of frame. (Host mode)</p>
	ISOONCIF	<p>Isochronous OUT transfer Not Complete Interrupt Flag</p> <p>At the end of a periodic frame (defined by EOPFT bit in USBFS_DCFG), USBFS will set this bit if there are still isochronous OUT endpoints for that not completed transactions. (Device Mode)</p>
20	ISOINCIF	<p>Isochronous IN transfer Not Complete Interrupt Flag</p> <p>At the end of a periodic frame (defined by EOPFT [1:0] bits in USBFS_DCFG), USBFS will set this bit if there are still isochronous IN endpoints for that not completed transactions. (Device Mode)</p> <p>Note: Only accessible in device mode.</p>
19	OEPIF	<p>OUT endpoint interrupt flag</p> <p>Set by USBFS when one of the OUT endpoints in device mode has raised an interrupt. Software should first read USBFS_DAEPINT register to get the device number, and then read the corresponding USBFS_DOEPxINTF register to get the flags of the endpoint that cause the interrupt. This bit will be automatically cleared after the respective endpoint's flags which cause this interrupt are cleared.</p>

---

		Note: Only accessible in device mode.
18	IEPIF	<p>IN endpoint interrupt flag</p> <p>Set by USBFS when one of the IN endpoints in device mode has raised an interrupt. Software should first read USBFS_DAEPINT register to get the device number, and then read the corresponding USBFS_DIEPxINTF register to get the flags of the endpoint that cause the interrupt. This bit will be automatically cleared after the respective endpoint's flags which cause this interrupt are cleared.</p> <p>Note: Only accessible in device mode.</p>
17:16	Reserved	Must be kept at reset value
15	EOPFIG	<p>End of periodic frame interrupt flag</p> <p>When USB bus time in a frame reaches the value defined by EOPFT [1:0] bits in USBFS_DCFG register, USBFS sets this flag.</p> <p>Note: Only accessible in device mode.</p>
14	ISOOPDIF	<p>Isochronous OUT packet dropped interrupt flag</p> <p>USBFS set this bit if it receives an isochronous OUT packet but cannot save it into Rx FIFO because the FIFO doesn't have enough space.</p> <p>Note: Only accessible in device mode.</p>
13	ENUMF	<p>Enumeration finished</p> <p>USBFS sets this bit after the speed enumeration finishes. Read USBFS_DSTAT register to get the current device speed.</p> <p>Note: Only accessible in device mode.</p>
12	RST	<p>USB reset</p> <p>USBFS sets this bit when it detects a USB reset signal on bus.</p> <p>Note: Only accessible in device mode.</p>
11	SP	<p>USB suspend</p> <p>USBFS sets this bit when it detects that the USB bus is idle for 3 ms and enters suspend state.</p> <p>Note: Only accessible in device mode.</p>
10	ESP	<p>Early suspend</p> <p>USBFS sets this bit when it detects that the USB bus is idle for 3 ms.</p> <p>Note: Only accessible in device mode.</p>
9:8	Reserved	Must be kept at reset value
7	GONAK	<p>Global OUT NAK effective</p> <p>Write 1 to SGONAK bit in the USBFS_DCTL register and USBFS will set GONAK flag after the writing to SGONAK takes effect.</p> <p>Note: Only accessible in device mode.</p>
6	GNPINAK	<p>Global Non-Periodic IN NAK effective</p> <p>Write 1 to SGINAK bit in the USBFS_DCTL register and USBFS will set GNPINAK</p>

		flag after the writing to SGINAK takes effect. Note: Only accessible in device mode.
5	NPTXFEIF	<p>Non-Periodic Tx FIFO empty interrupt flag</p> <p>This interrupt is triggered when the non-periodic transmit FIFO is either half or completely empty. The threshold is determined by the non-periodic Tx FIFO empty level bit (TXFTH) in the USBFS_GAHBCS register.</p> <p>Note: Only accessible in host mode.</p>
4	RXFNEIF	<p>Rx FIFO non-empty interrupt flag</p> <p>USBFS sets this bit when there is at least one packet or status entry in the Rx FIFO.</p> <p>Note: Accessible in both host and device modes.</p>
3	SOF	<p>Start of frame</p> <p>Host Mode: USBFS sets this bit when it prepares to transmit a SOF or Keep-Alive on USB bus. Software can clear this bit by writing 1.</p> <p>Device Mode: USBFS sets this bit after it receives a SOF token. The application can read the Device Status register to get the current frame number. Software can clear this bit by writing 1.</p> <p>Note: Accessible in both host and device modes.</p>
2	OTGIF	<p>OTG interrupt flag</p> <p>USBFS sets this bit when the flags in USBFS_GOTGINTF register generate an interrupt. Software should read USBFS_GOTGINTF register to get the source of this interrupt. This bit is cleared after the flags in USBFS_GOTGINTF causing this interrupt are cleared.</p> <p>Note: Accessible in both host and device modes.</p>
1	MFIF	<p>Mode fault interrupt flag</p> <p>USBFS sets this bit when software operates host-only register in device mode, or operates device-mode in host mode. These fault operations won't take effect.</p> <p>Note: Accessible in both host and device modes.</p>
0	COPM	<p>Current operation mode</p> <p>0: Device mode 1: Host mode</p> <p>Note: Accessible in both host and device modes.</p>

### Global interrupt enable register (USBFS\_GINTEN)

Address offset: 0x0018

Reset value: 0x0000 0000

This register works with the global interrupt flag register (USBFS\_GINTF) to interrupt the application. When an interrupt enable bit is disabled, the interrupt associated with that bit is not generated. However, the global Interrupt flag register bit corresponding to that interrupt is still set.

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
WKUPIE	SEIE	DISCIE	IDPSCIE	Reserved.	PTXFEIE	HCIE	HPIE	Reserved	ISOONCIE	PXNCIE/	ISOINCIE	OEPIE	IEPIE	Reserved	
rw	rw	rw	rw		rw	rw	r		rw	rw	rw	rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EOPFIE	ISOOPDIE	ENUMFIE	RSTIE	SPIE	ESPIE	Reserved	GONAKIE	GNPINAKIE	NPTXFEIE	RXFNEIE	SOFIE	OTGIE	MFIE	Reserved	
rw	rw	rw	rw	rw	rw		rw		rw	rw	rw	rw	rw	rw	

Bits	Fields	Descriptions
31	WKUPIE	Wakeup interrupt enable 0: Disable wakeup interrupt 1: Enable wakeup interrupt  Note: Accessible in both host and device modes.
30	SEIE	Session interrupt enable 0: Disable session interrupt 1: Enable session interrupt  Note: Accessible in both host and device modes.
29	DISCIE	Disconnect interrupt enable 0: Disable disconnect interrupt 1: Enable disconnect interrupt  Note: Only accessible in device mode.
28	IDPSCIE	ID pin status change interrupt enable 0: Disable connector ID pin status interrupt 1: Enable connector ID pin status interrupt  Note: Accessible in both host and device modes.
27	Reserved	Must be kept at reset value
26	PTXFEIE	Periodic Tx FIFO empty interrupt enable 0: Disable periodic Tx FIFO empty interrupt 1: Enable periodic Tx FIFO empty interrupt  Note: Only accessible in host mode.
25	HCIE	Host channels interrupt enable 0: Disable host channels interrupt 1: Enable host channels interrupt  Note: Only accessible in host mode.

24	HPIE	Host port interrupt enable 0: Disable host port interrupt 1: Enable host port interrupt Note: Only accessible in host mode.
23:22	Reserved	Must be kept at reset value
21	PXNCIE	Periodic transfer not complete Interrupt enable 0: Disable periodic transfer not complete interrupt 1: Enable periodic transfer not complete interrupt Note: Only accessible in host mode.
	ISOONCIE	Isochronous OUT transfer not complete interrupt enable 0: Disable isochronous OUT transfer not complete interrupt 1: Enable isochronous OUT transfer not complete interrupt Note: Only accessible in device mode.
20	ISOINCIE	Isochronous IN transfer not complete interrupt enable 0: Disable isochronous IN transfer not complete interrupt 1: Enable isochronous IN transfer not complete interrupt Note: Only accessible in device mode.
19	OEPIE	OUT endpoints interrupt enable 0: Disable OUT endpoints interrupt 1: Enable OUT endpoints interrupt Note: Only accessible in device mode.
18	IEPIE	IN endpoints interrupt enable 0: Disable IN endpoints interrupt 1: Enable IN endpoints interrupt Note: Only accessible in device mode.
17:16	Reserved	Must be kept at reset value
15	EOPFIE	End of periodic frame interrupt enable 0: Disable end of periodic frame interrupt 1: Enable end of periodic frame interrupt Note: Only accessible in device mode.
14	ISOOPDIE	Isochronous OUT packet dropped interrupt enable 0: Disable isochronous OUT packet dropped interrupt 1: Enable isochronous OUT packet dropped interrupt Note: Only accessible in device mode.
13	ENUMFIE	Enumeration finish enable 0: Disable enumeration finish interrupt 1: Enable enumeration finish interrupt Note: Only accessible in device mode.
12	RSTIE	USB reset interrupt enable

---

		0: Disable USB reset interrupt 1: Enable USB reset interrupt Note: Only accessible in device mode.
11	SPIE	USB suspend interrupt enable 0: Disable USB suspend interrupt 1: Enable USB suspend interrupt Note: Only accessible in device mode.
10	ESPIE	Early suspend interrupt enable 0: Disable early suspend interrupt 1: Enable early suspend interrupt Note: Only accessible in device mode.
9:8	Reserved	Must be kept at reset value
7	GONAKIE	Global OUT NAK effective interrupt enable 0: Disable global OUT NAK interrupt 1: Enable global OUT NAK interrupt Note: Only accessible in device mode.
6	GNPINAKIE	Global non-periodic IN NAK effective interrupt enable 0: Disable global non-periodic IN NAK effective interrupt 1: Enable global non-periodic IN NAK effective interrupt Note: Only accessible in device mode.
5	NPTXFEIE	Non-periodic Tx FIFO empty interrupt enable 0: Disable non-periodic Tx FIFO empty interrupt 1: Enable non-periodic Tx FIFO empty interrupt Note: Only accessible in Host mode.
4	RXFNEIE	Receive FIFO non-empty interrupt enable 0: Disable receive FIFO non-empty interrupt 1: Enable receive FIFO non-empty interrupt Note: Accessible in both device and host modes.
3	SOFIE	Start of frame interrupt enable 0: Disable start of frame interrupt 1: Enable start of frame interrupt Note: Accessible in both device and host modes.
2	OTGIE	OTG interrupt enable 0: Disable OTG interrupt 1: Enable OTG interrupt Note: Accessible in both device and host modes.
1	MFIE	Mode fault interrupt enable 0: Disable mode fault interrupt 1: Enable mode fault interrupt

Note: Accessible in both device and host modes.

0 Reserved Must be kept at reset value

**Global receive status read/receive status read and pop registers  
(USBFS\_GRSTATR/USBFS\_GRSTATP)**

Address offset for Read: 0x001C

Address offset for Pop: 0x0020

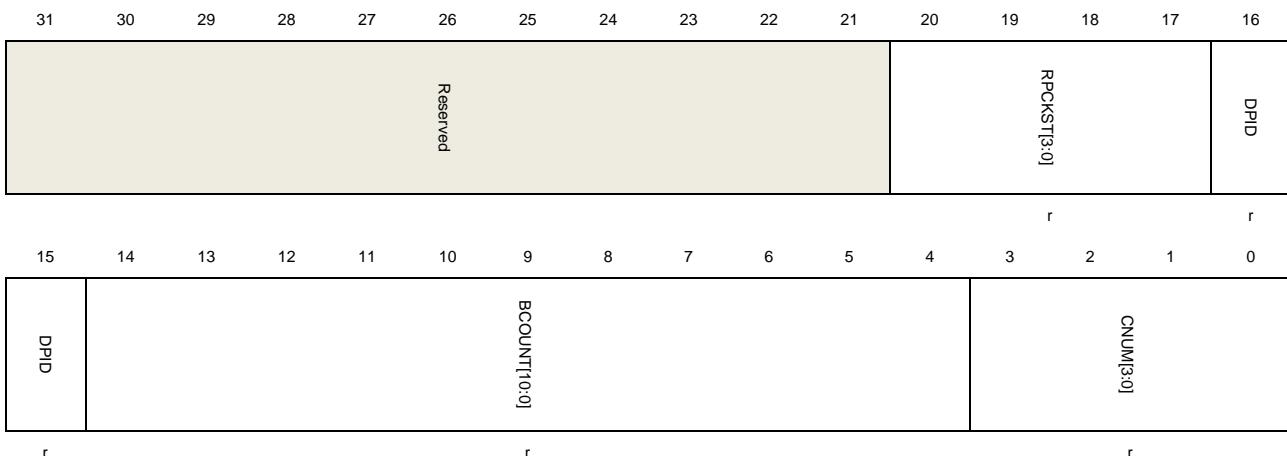
Reset value: 0x0000 0000

A read to the receive status read register returns the entry of the top of the Rx FIFO. A read to the Receive status read and pop register additionally pops the top entry out of the Rx FIFO.

The entries in RxFIFO have different meanings in host and device modes. Software should only read this register after when Receive FIFO non-empty interrupt flag bit of the global interrupt flag register (RXFNEIF bit in USBFS\_GINTF) is triggered.

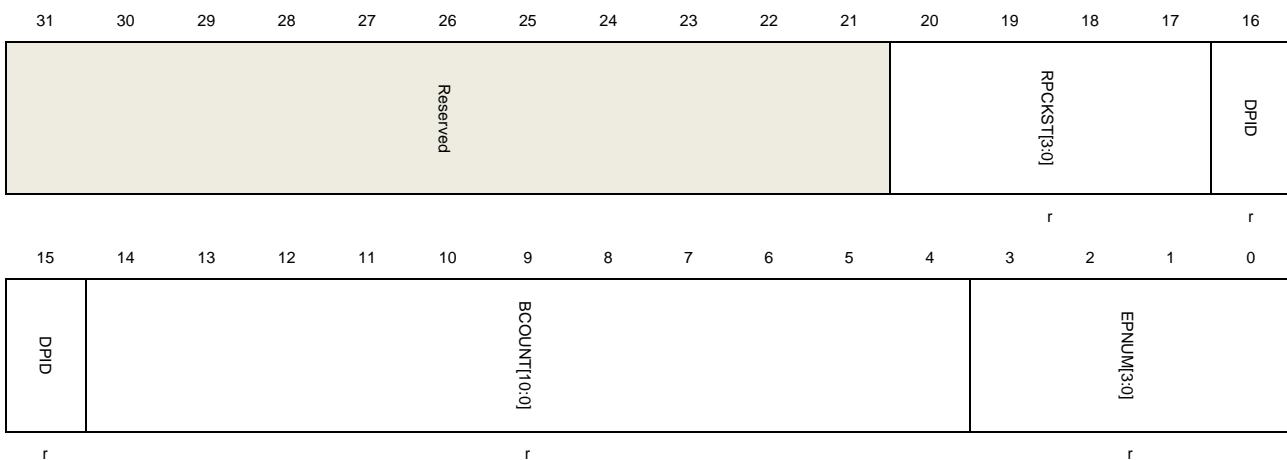
This register has to be accessed by word (32-bit)

## Host mode:



Bits	Fields	Descriptions
31:21	Reserved	Must be kept at reset value
20:17	RPCKST[3:0]	<p>Received packet status</p> <p>0010: IN data packet received</p> <p>0011: IN transfer completed (generates an interrupt if poped)</p> <p>0101: Data toggle error (generates an interrupt if poped)</p> <p>0111: Channel halted (generates an interrupt if poped)</p> <p>Others: Reserved</p>
16:15	DPID[1:0]	<p>Data PID</p> <p>The Data PID of the received packet</p> <p>00: DATA0</p> <p>10: DATA1</p>

		01: DATA2
		11: MDATA
14:4	BCOUNT[10:0]	Byte count The byte count of the received IN data packet.
3:0	CNUM[3:0]	Channel number The channel number to which the current received packet belongs.

**Device mode:**


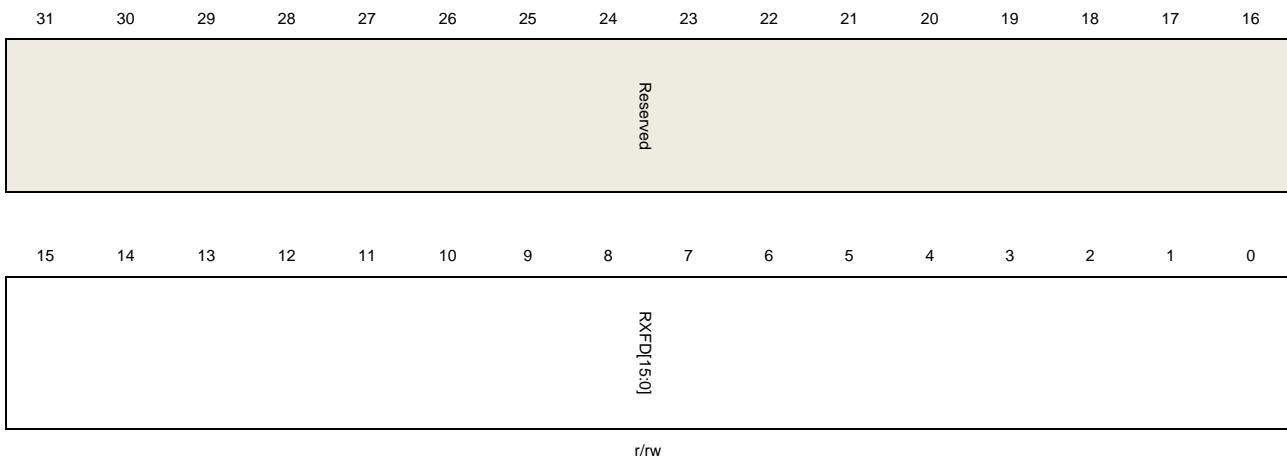
Bits	Fields	Descriptions
31:21	Reserved	Must be kept at reset value
20:17	RPCKST[3:0]	Received packet status 0001: Global OUT NAK (generates an interrupt) 0010: OUT data packet received 0011: OUT transfer completed (generates an interrupt) 0100: SETUP transaction completed (generates an interrupt) 0110: SETUP data packet received Others: Reserved
16:15	DPID[1:0]	Data PID The Data PID of the received OUT data packet 00: DATA0 10: DATA1 01: DATA2 11: MDATA
14:4	BCOUNT[10:0]	Byte count The byte count of the received data packet.
3:0	EPNUM[3:0]	Endpoint number The endpoint number to which the current received packet belongs.

### Global receive FIFO length register (USBFS\_GRFLEN)

Address offset: 0x024

Reset value: 0x0000 0200

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



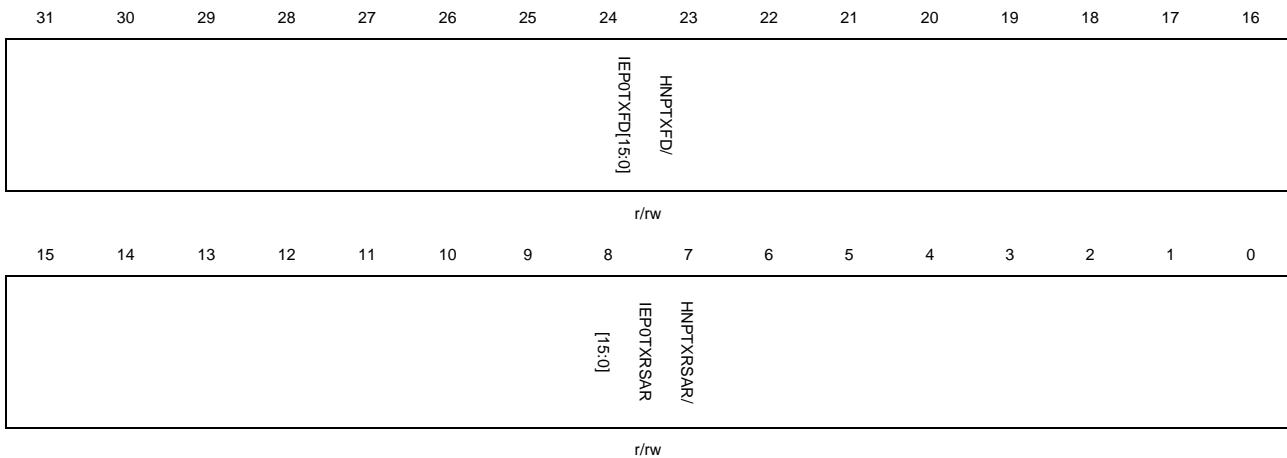
Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	RXFD[15:0]	Rx FIFO depth In terms of 32-bit words. $1 \leqslant \text{RXFD} \leqslant 1024$

### Host non-periodic transmit FIFO length register /Device IN endpoint 0 transmit FIFO length (USBFS\_HNPTFLEN \_DIEP0TFLEN)

Address offset: 0x028

Reset value: 0x0200 0200

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



**Host Mode:**

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:16	HNPTXFD[15:0]	Host Non-periodic Tx FIFO depth In terms of 32-bit words. $1 \leq HNPTXFD \leq 1024$
15:0	HNPTXRSAR[15:0]	Host Non-periodic Tx RAM start address The start address for non-periodic transmit FIFO RAM is in term of 32-bit words.

**Device Mode:**

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:16	IEP0TXFD[15:0]	IN Endpoint 0 Tx FIFO depth In terms of 32-bit words. $16 \leq IEP0TXFD \leq 140$
15:0	IEP0TXRSAR[15:0]	IN Endpoint 0 TX RAM start address The start address for endpoint0 transmit FIFO RAM is in term of 32-bit words.

**Host non-periodic transmit FIFO/queue status register (USBFS\_HNPTFQSTAT)**

Address offset: 0x002C

Reset value: 0x0008 0200

This register reports the current status of the non-periodic Tx FIFO and request queue. The request queue holds IN, OUT or other request entries in host mode.

**Note:** In Device mode, this register is not valid.

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



<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31	Reserved	Must be kept at reset value

30:24	NPTXRQTOP[6:0]	Top entry of the non-periodic Tx request queue Entry in the non-periodic transmit request queue. Bits 30:27: Channel number Bits 26:25: <ul style="list-style-type: none"><li>- 00: IN/OUT token</li><li>- 01: Zero-length OUT packet</li><li>- 11: Channel halt request</li></ul> Bit 24: Terminate Flag, indicating last entry for selected channel.
23:16	NPTXRQS[7:0]	Non-periodic Tx request queue space The remaining space of the non-periodic transmit request queue. 0: Request queue is Full 1: 1 entry 2: 2 entries ... n: n entries ( $0 \leq n \leq 8$ ) Others: Reserved
15:0	NPTXFS[15:0]	Non-periodic Tx FIFO space The remaining space of the non-periodic transmit FIFO. In terms of 32-bit words. 0: Non-periodic Tx FIFO is full 1: 1 word 2: 2 words n: n words ( $0 \leq n \leq \text{NPTXFD}$ ) Others: Reserved

### Global core configuration register (USBFS\_GCCFG)

Address offset: 0x0038

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
										VBUSIG	SOFOEN	VBUSBCEN		Reserved	PWRON
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Reserved

Reserved

Bits	Fields	Descriptions
31:22	Reserved	Must be kept at reset value
21	VBUSIG	<p>VBUS ignored</p> <p>When this bit is set, USBFS doesn't monitor the voltage on VBUS pin and always consider VBUS voltage as valid both in host mode and in device mode, then free the VBUS pin for other usage.</p> <p>0: VBUS is not ignored.</p> <p>1: VBUS is ignored and always consider VBUS voltage as valid.</p>
20	SOFOEN	<p>SOF output enable</p> <p>0: SOF pulse output disabled.</p> <p>1: SOF pulse output enabled.</p>
19	VBUSBCEN	<p>The V<sub>BUS</sub> B-device Comparer enable</p> <p>0: V<sub>BUS</sub> B-device comparer disabled</p> <p>1: V<sub>BUS</sub> B-device comparer enabled</p>
18	VBUSACEN	<p>The V<sub>BUS</sub> A-device Comparer enable</p> <p>0: V<sub>BUS</sub> A-device comparer disabled</p> <p>1: V<sub>BUS</sub> A-device comparer enabled</p>
17	Reserved	Must be kept at reset value
16	PWRON	<p>Power on</p> <p>This bit is the power switch for the internal embedded Full-Speed PHY.</p> <p>0: Embedded Full-Speed PHY power off.</p> <p>1: Embedded Full-Speed PHY power on.</p>
15:0	Reserved	Must be kept at reset value.

### Core ID register (USBFS\_CID)

Address offset: 0x003C

Reset value: 0x0000 1000

This register contains the Product ID.

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
CID[31:16]															
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

CID[15:0]
rw

Bits	Fields	Descriptions
31:0	CID[31:0]	Core ID  Software can write or read this field and uses this field as a unique ID for its application

### Host periodic transmit FIFO length register (USBFS\_HPTFLEN)

Address offset: 0x0100

Reset value: 0x0200 0600

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
HPTXFD															
r/rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HPTXFSAR															
r/rw															

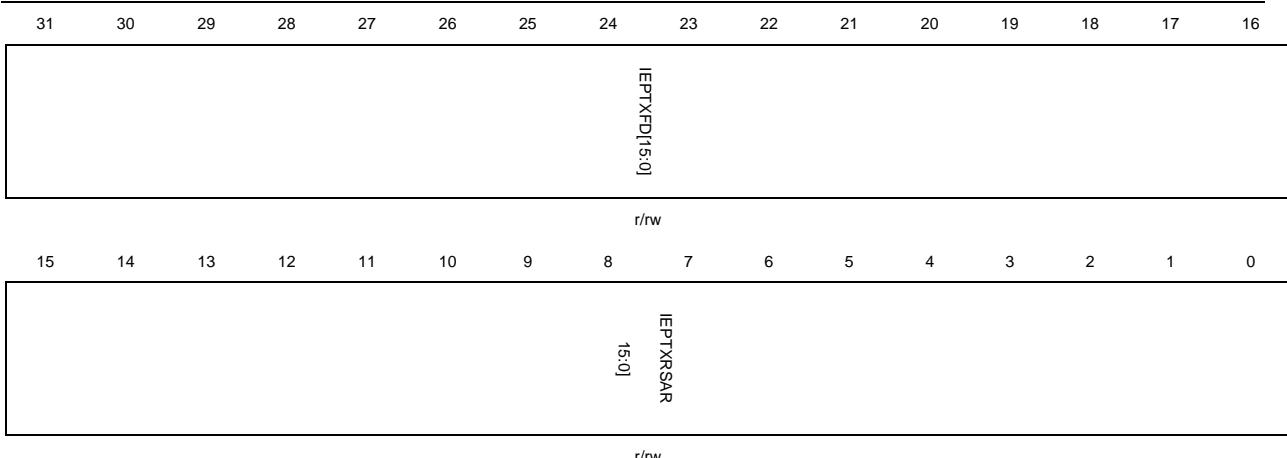
Bits	Fields	Descriptions
31:16	HPTXFD[15:0]	Host Periodic Tx FIFO depth  In terms of 32-bit words.  $1 \leq HPTXFD \leq 1024$
15:0	HPTXFSAR[15:0]	Host periodic Tx FIFO RAM start address  The start address for host periodic transmit FIFO RAM is in term of 32-bit words.

### Device IN endpoint transmit FIFO length register (USBFS\_DIEPxTFLEN) (x = 1..3, where x is the FIFO\_number)

Address offset: 0x0104 + (FIFO\_number – 1) × 0x04

Reset value: 0x0200 0400

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



Bits	Fields	Descriptions
31:16	IEPTXFD[15:0]	IN endpoint Tx FIFO depth In terms of 32-bit words. $1 \leq HPTXFD \leq 1024$
15:0	IEPTXRSAR[15:0]	IN endpoint FIFO Tx RAM start address The start address for IN endpoint transmit FIFOx is in term of 32-bit words.

## 21.7.2. Host control and status registers

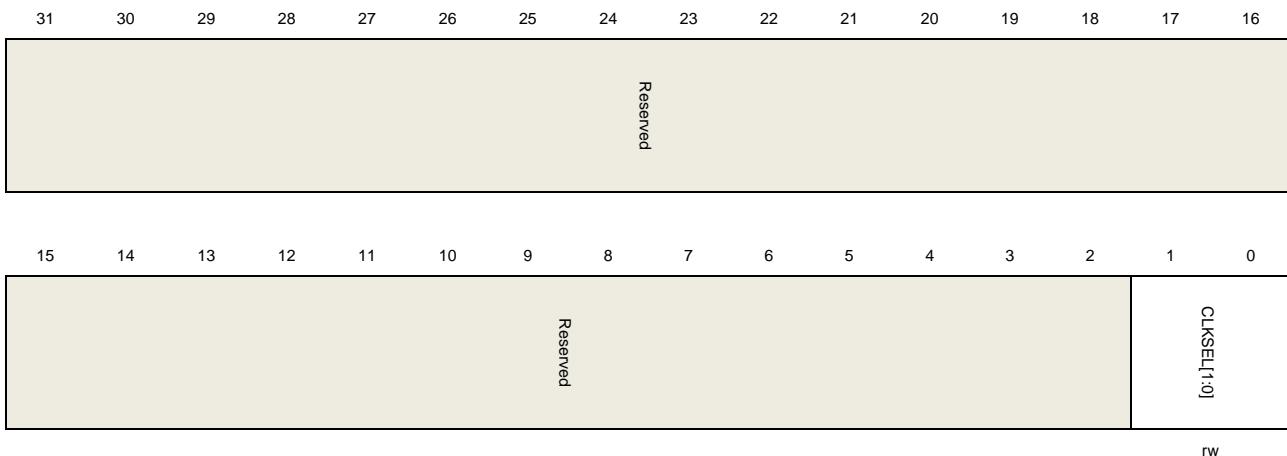
### Host control register (USBFS\_HCTL)

Address offset: 0x0400

Reset value: 0x0000 0000

This register configures the core after power on in host mode. Do not modify it after host initialization.

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



<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:2	Reserved	Must be kept at reset value
1:0	CLKSEL[1:0]	Clock select for usbclock. 01: 48MHz clock others: reserved

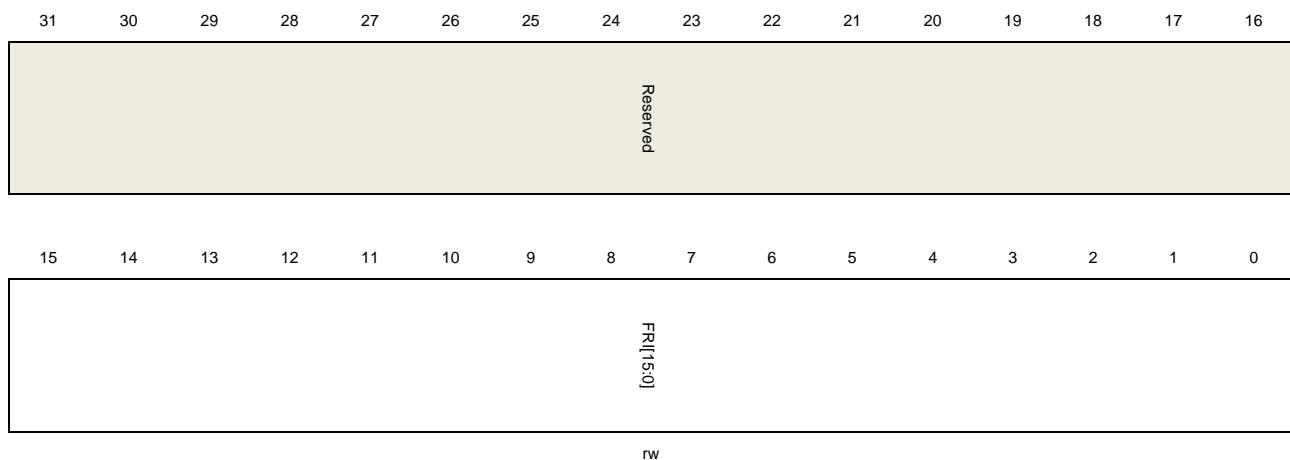
### **Host frame interval register (USBFS\_HFT)**

Address offset: 0x0404

Reset value: 0x0000 BB80

This register sets the frame interval for the current enumerating speed when USBFS controller is enumerating.

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



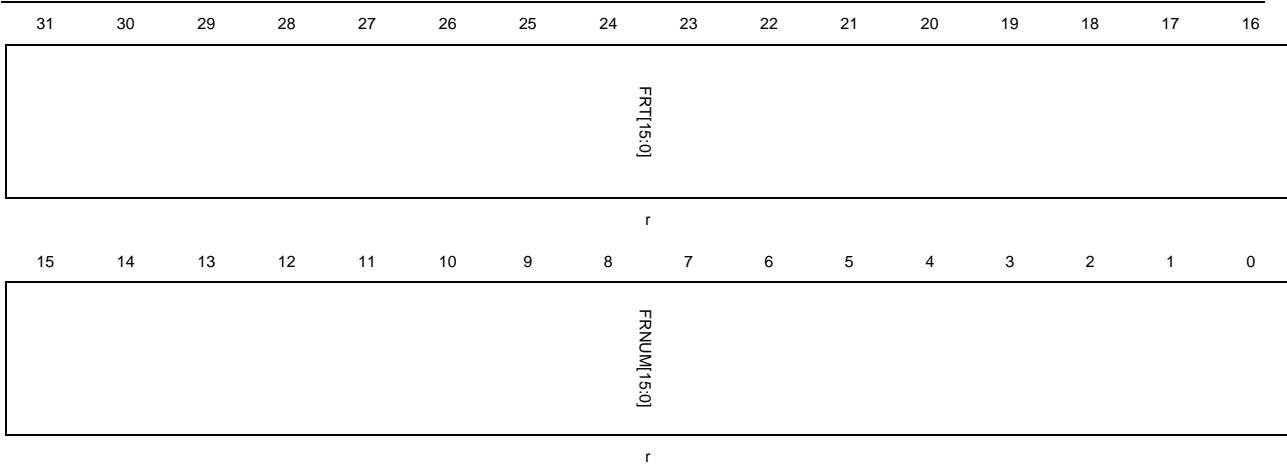
<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:16	Reserved	Must be kept at reset value
15:0	FRI[15:0]	Frame interval  This value describes the frame time in terms of PHY clocks. Each time when port is enabled after a port reset operation, USBFS use a proper value according to the current speed, and software can write to this field to change the value. This value should be calculated using the frequency described below: Full-Speed: 48MHz Low-Speed: 6MHz

### **Host frame information remaining register (USBFS\_HFINFR)**

Address offset: 0x408

Reset value: 0xBB80 0000

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



<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:16	FRT[15:0]	Frame remaining time This field reports the remaining time of current frame in terms of PHY clocks.
15:0	FRNUM[15:0]	Frame number This field reports the frame number of current frame and returns to 0 after it reaches 0x3FFF.

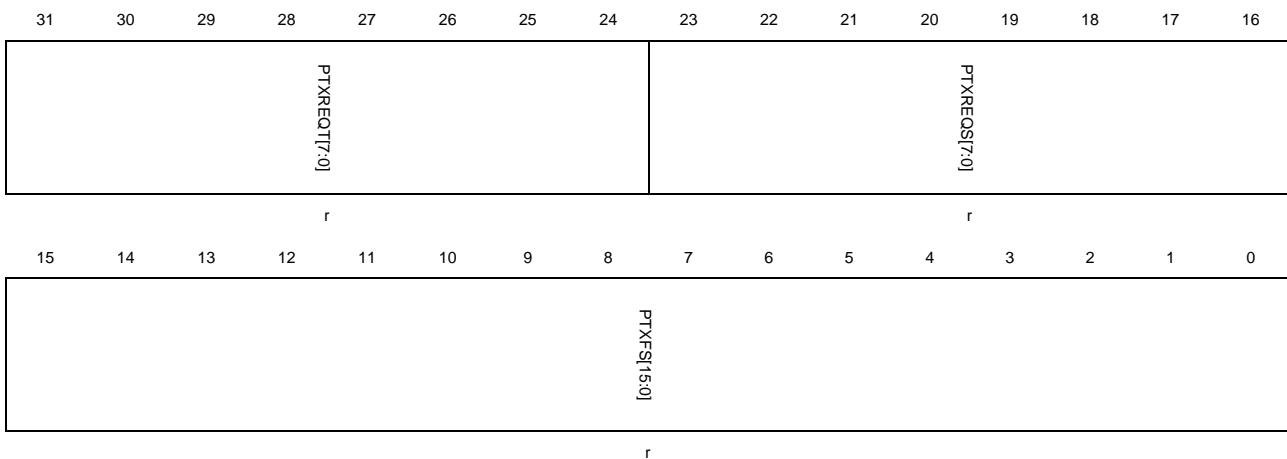
### Host periodic transmit FIFO/queue status register (**USBFS\_HPTFQSTAT**)

Address offset: 0x0410

Reset value: 0x0008 0200

This register reports the current status of the host periodic Tx FIFO and request queue. The request queue holds IN, OUT or other request entries in host mode.

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



<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:24	PTXREQT[7:0]	Top entry of the periodic Tx request queue

		Entry in the periodic transmit request queue.
		Bits 30:27: Channel Number
		Bits 26:25:
		00: IN/OUT token
		01: Zero-length OUT packet
		11: Channel halt request
		Bit 24: Terminate Flag, indicating last entry for selected channel.
23:16	PTXREQS[7:0]	<p>Periodic Tx request queue space</p> <p>The remaining space of the periodic transmit request queue.</p> <p>0: Request queue is Full</p> <p>1: 1 entry</p> <p>2: 2 entries</p> <p>...</p> <p>n: n entries (<math>0 \leq n \leq 8</math>)</p> <p>Others: Reserved</p>
15:0	PTXFS[15:0]	<p>Periodic Tx FIFO space</p> <p>The remaining space of the periodic transmit FIFO.</p> <p>In terms of 32-bit words.</p> <p>0: periodic Tx FIFO is full</p> <p>1: 1 word</p> <p>2: 2 words</p> <p>n: n words (<math>0 \leq n \leq \text{PTXFD}</math>)</p> <p>Others: Reserved</p>

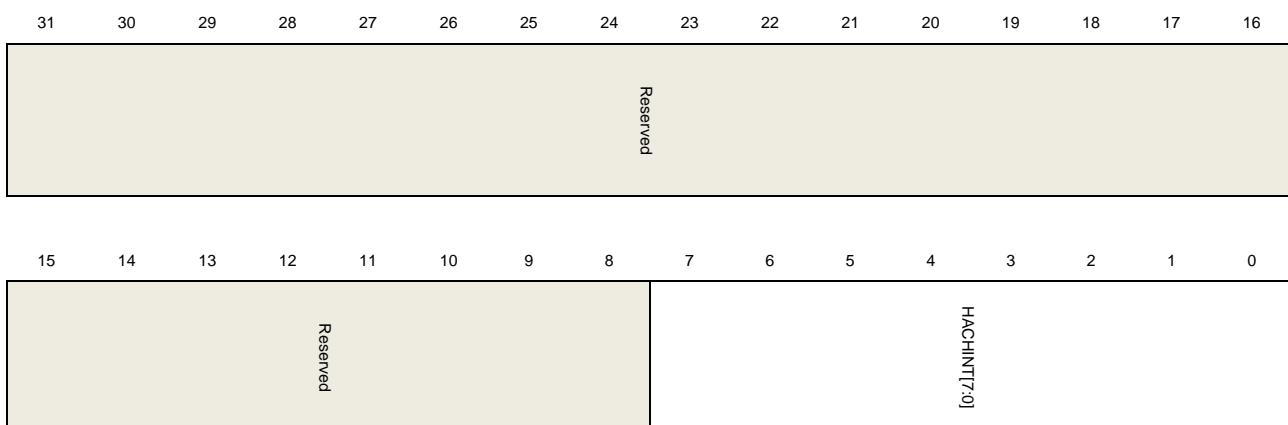
### Host all channels interrupt register (USBFS\_HACHINT)

Address offset: 0x0414

Reset value: 0x0000 0000

When a channel interrupt is triggered, USBFS set corresponding bit in this register and software should read this register to know which channel is asserting interrupts.

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



<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:8	Reserved	Must be kept at reset value
7:0	HACHINT[7:0]	Host all channel interrupts Each bit represents a channel: Bit 0 for channel 0, bit 7 for channel 7.

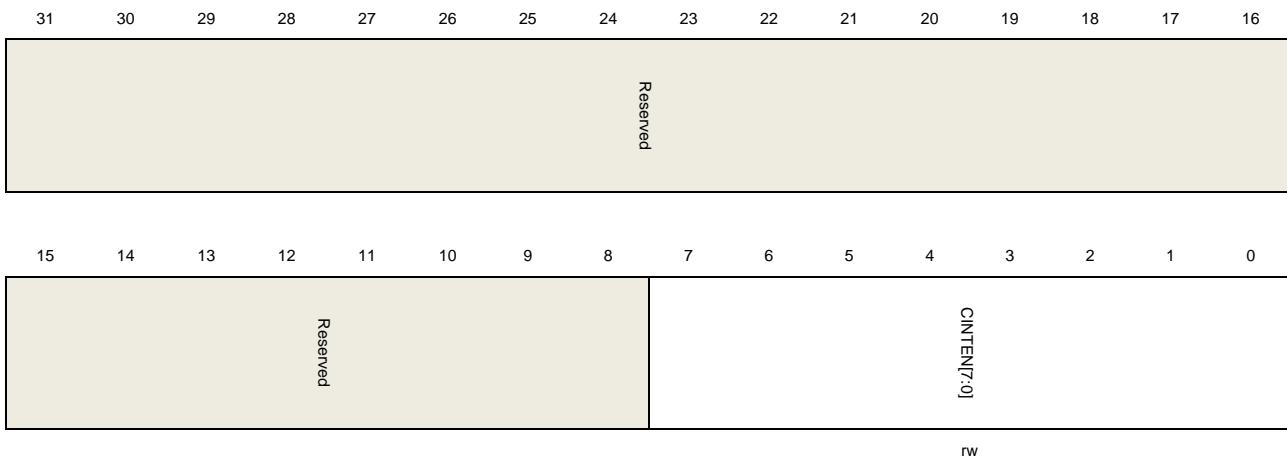
### **Host all channels interrupt enable register (USBFS\_HACHINTEN)**

Address offset: 0x0418

Reset value: 0x0000 0000

This register can be used by software to enable or disable a channel's interrupt. Only the channel whose corresponding bit in this register is set is able to cause the channel interrupt flag HCIF in USBFS\_GINTF register.

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



<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:8	Reserved	Must be kept at reset value
7:0	CINTEN[7:0]	Channel interrupt enable 0: Disable channel-n interrupt 1: Enable channel-n interrupt Each bit represents a channel: Bit 0 for channel 0, bit 7 for channel 7.

### **Host port control and status register (USBFS\_HPCS)**

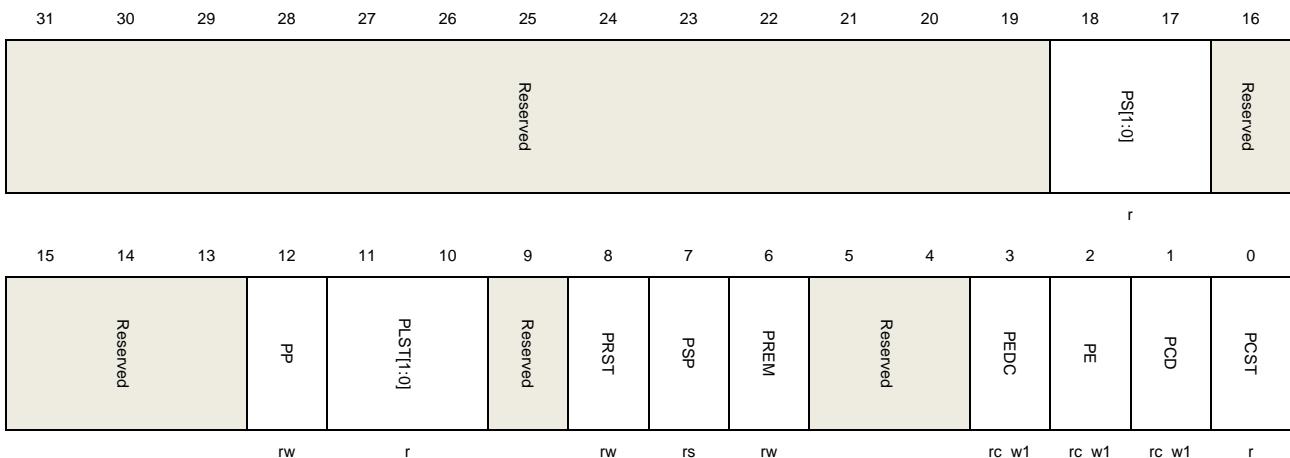
Address offset: 0x0440

Reset value: 0x0000 0000

This register controls the port's behavior and also has some flags which report the status of the port. The HPIF flag in USBFS\_GINTF register will be triggered if one of these flags in this

register is set by USBFS: PRST, PEDC and PCD.

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



<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:19	Reserved	Must be kept at reset value
18:17	PS[1:0]	Port speed Report the enumerated speed of the device attached to this port. 01: Full speed 10: Low speed Others: Reserved
16:13	Reserved	Must be kept at reset value
12	PP	Port power This bit should be set before a port is used. Because USBFS doesn't have power supply ability, it only uses this bit to know whether the port is in powered state. Software should ensure the true power supply on VBUS before setting this bit. 0: Port is powered off 1: Port is powered on
11:10	PLST[1:0]	Port line status Report the current state of USB data lines Bit 10: State of DP line Bit 11: State of DM line
9	Reserved	Must be kept at reset value
8	PRST	Port reset Application sets this bit to start a reset signal on USB port. Application should clear this bit when it wants to stop the reset signal. 0: Port is not in reset state 1: Port is in reset state

7	PSP	Port suspend Application sets this bit to put port into suspend state. When this bit is set the port stops sending SOF tokens. This bit can only be cleared by the following operations: PRST bit in this register is set by application PREM bit in this register is set A remote wakeup signal is detected A device disconnect is detected 0: Port is not in suspend state 1: Port is in suspend state
6	PREM	Port resume Application sets this bit to start a resume signal on USB port. Application should clear this bit when it wants to stop the resume signal. 0: No resume driven 1: Resume driven
5:4	Reserved	Must be kept at reset value
3	PEDC	Port enable/disable change Set by the core when the status of the Port enable bit 2 in this register changes.
2	PE	Port Enable This bit is automatically set by USBFS after a USB reset signal finishes and cannot be set by software. This bit is cleared by the following events: A disconnect condition Software clearing this bit 0: Port disabled 1: Port enabled
1	PCD	Port connect detected Set by USBFS when a device connection is detected. This bit can be cleared by writing 1 to this bit.
0	PCST	Port connect status 0: Device is not connected to the port 1: Device is connected to the port

**Host channel-x control register (USBFS\_HCHxCTL) (x = 0..7 where x = channel\_number)**

Address offset: 0x0500 + (channel\_number × 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
CEN	CDIS	ODDFRM			DAR[6:0]			Reserved		EPTYPE[1:0]		LSD		Reserved	
rs	rs	rw			rw					rw		rw		rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EPDIR		EPNUM[3:0]						MPL[10:0]							
rw		rw						rw							

Bits	Fields	Descriptions
31	CEN	<p>Channel enable</p> <p>Set by the application and cleared by USBFS.</p> <p>0: Channel disabled</p> <p>1: Channel enabled</p> <p>Software should follow the operation guide to disable or enable a channel.</p>
30	CDIS	<p>Channel disable</p> <p>Software can set this bit to disable the channel from processing transactions.</p> <p>Software should follow the operation guide to disable or enable a channel.</p>
29	ODDFRM	<p>Odd frame</p> <p>For periodic transfers (interrupt or isochronous transfer), this bit controls that whether in an odd frame or even frame this channel's transaction is desired to be processed.</p> <p>0: Even frame</p> <p>1: Odd frame</p>
28:22	DAR[6:0]	<p>Device address</p> <p>The address of the USB device that this channel wants to communicate with.</p>
21:20	Reserved	Must be kept at reset value
19:18	EPTYPE[1:0]	<p>Endpoint type</p> <p>The transfer type of the endpoint that this channel wants to communicate with.</p> <p>00: Control</p> <p>01: Isochronous</p> <p>10: Bulk</p> <p>11: Interrupt</p>
17	LSD	<p>Low-Speed device</p> <p>The device that this channel wants to communicate with is a Low-Speed Device.</p>
16	Reserved	Must be kept at reset value

---

15	EPDIR	Endpoint direction The transfer direction of the endpoint that this channel wants to communicate with. 0: OUT 1: IN
14:11	EPNUM[3:0]	Endpoint number The number of the endpoint that this channel wants to communicate with.
10:0	MPL[10:0]	Maximum packet length The target endpoint's maximum packet length.

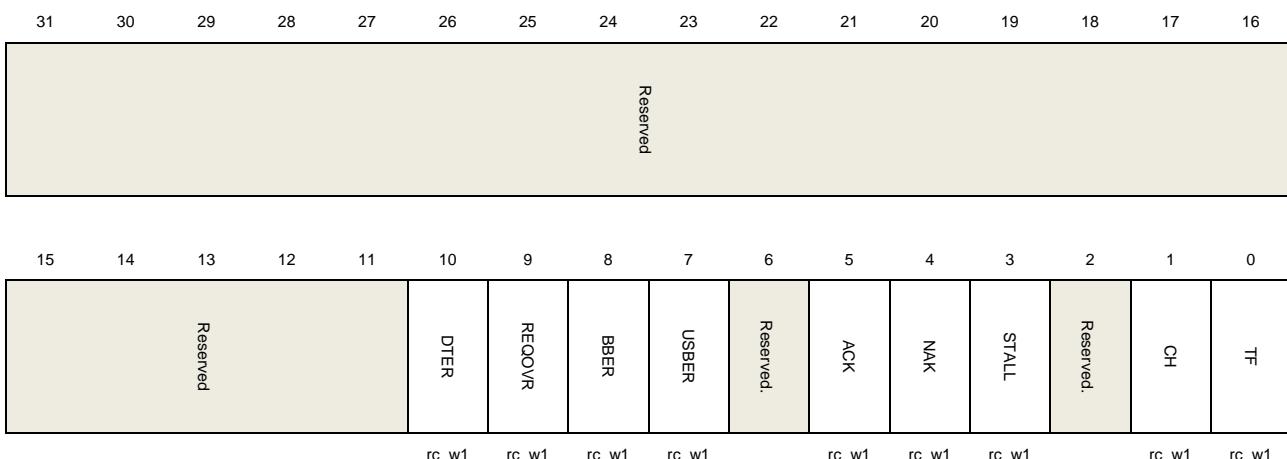
### **Host channel-x interrupt flag register (USBFS\_HCHxINTF) (x = 0..7 where x = channel number)**

Address offset: 0x0508 + (channel\_number × 0x20)

Reset value: 0x0000 0000

This register contains the status and events of a channel, when software get a channel interrupt, it should read this register for the respective channel to know the source of the interrupt. The flag bits in this register are all set by hardware and cleared by writing 1.

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




---

Bits	Fields	Descriptions
31:11	Reserved	Must be kept at reset value
10	DTER	Data toggle error The IN transaction gets a data packet but the PID of this packet doesn't match DPID [1:0] bits in USBFS_HCHxLEN register.
9	REQOVR	Request queue overrun The periodic request queue is full when software starts new transfers.
8	BBER	Babble error A babble condition occurs on USB bus. A typical reason for babble condition is that

a device sends a data packet and the packet length exceeds the endpoint's maximum packet length.

7	USBER	USB Bus Error
		The USB error flag is set when the following conditions occurs during receiving a packet:
		A received packet has a wrong CRC field
		A stuff error detected on USB bus
		Timeout when waiting for a response packet
6	Reserved	Must be kept at reset value
5	ACK	ACK
		An ACK response is received or transmitted
4	NAK	NAK
		A NAK response is received.
3	STALL	STALL
		A STALL response is received.
2	Reserved	Must be kept at reset value
1	CH	Channel halted
		This channel is disabled by a request, and it will not response to other requests during the request processing.
0	TF	Transfer finished
		All the transactions of this channel finish successfully, and no error occurs. For IN channel, this flag will be triggered after PCNT bits in USBFS_HCHxLEN register reach zero. For OUT channel, this flag will be triggered when software reads and pops a TF status entry from the RxFIFO.

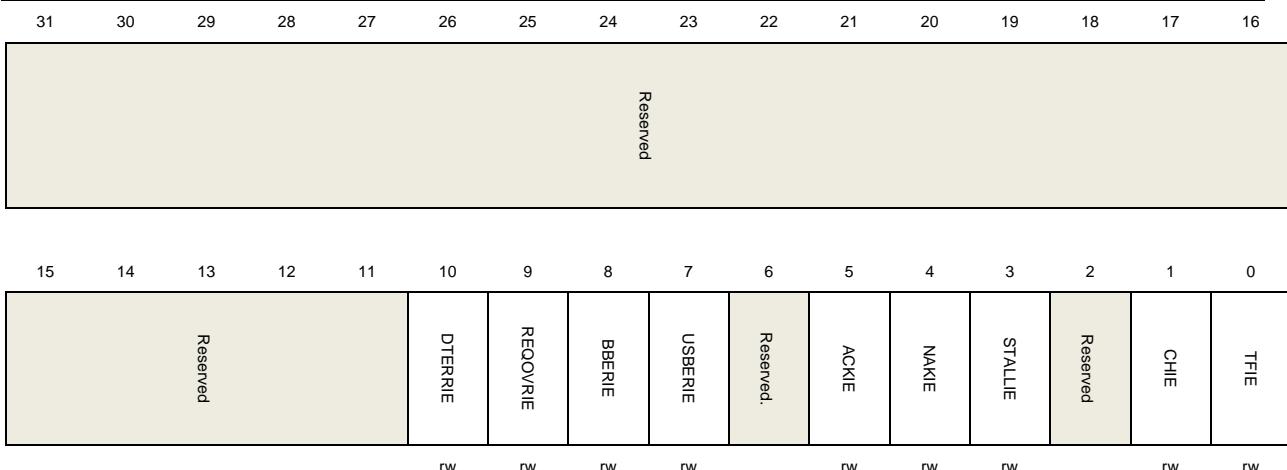
### **Host channel-x interrupt enable register (USBFS\_HCHxINTEN) (x = 0..7, where x = channel number)**

Address offset: 0x050C + (channel\_number × 0x20)

Reset value: 0x0000 0000

This register contains the interrupt enable bits for the flags in USBFS\_HCHxINTF register. If a bit in this register is set by software, the corresponding bit in USBFS\_HCHxINTF register is able to trigger a channel interrupt. The bits in this register are set and cleared by software.

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



Bits	Fields	Descriptions
31:11	Reserved	Must be kept at reset value
10	DTERIE	Data toggle error interrupt enable 0: Disable data toggle error interrupt 1: Enable data toggle error interrupt
9	REQOVRIE	Request queue overrun interrupt enable 0: Disable request queue overrun interrupt 1: Enable request queue overrun interrupt
8	BBERIE	Babble error interrupt enable 0: Disable babble error interrupt 1: Enable babble error interrupt
7	USBERIE	USB bus error interrupt enable 0: Disable USB bus error interrupt 1: Enable USB bus error interrupt
6	Reserved	Must be kept at reset value
5	ACKIE	ACK interrupt enable 0: Disable ACK interrupt 1: Enable ACK interrupt
4	NAKIE	NAK interrupt enable 0: Disable NAK interrupt 1: Enable NAK interrupt
3	STALLIE	STALL interrupt enable 0: Disable STALL interrupt 1: Enable STALL interrupt
2	Reserved	Must be kept at reset value

---

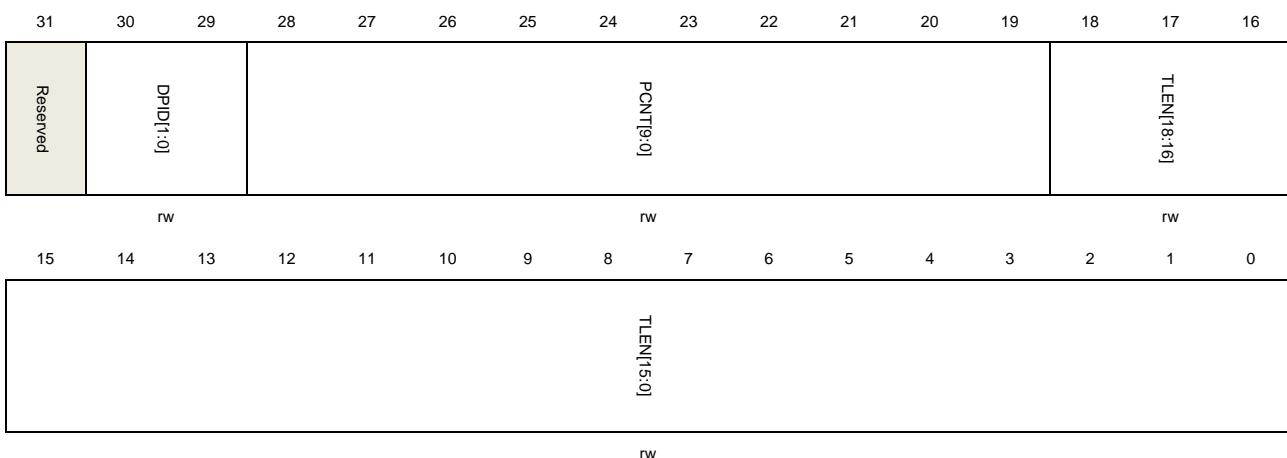
1	CHIE	Channel halted interrupt enable 0: Disable channel halted interrupt 1: Enable channel halted interrupt
0	TFIE	Transfer finished interrupt enable 0: Disable transfer finished interrupt 1: Enable transfer finished interrupt

### **Host channel-x transfer length register (USBFS\_HCHxLEN) (x = 0..7, where x = channel number)**

Address offset: 0x0510 + (channel\_number × 0x20)

Reset value: 0x0000 0000

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




---

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31	Reserved	Must be kept at reset value
30:29	DPID[1:0]	Data PID Software should write this field before the transfer starts. For OUT transfers, this field controls the Data PID of the first transmitted packet. For IN transfers, this field controls the expected Data PID of the first received packet, and DTERR will be triggered if the Data PID doesn't match. After the transfer starts, USBFS changes and toggles this field automatically following the USB protocol. 00: DATA0 10: DATA1 11: SETUP (For control transfer only) 01: Reserved
28:19	PCNT[9:0]	Packet count The number of data packets desired to be transmitted (OUT) or received (IN) in a transfer.

Software should program this field before the channel is enabled. After the transfer starts, this field is decreased automatically by USBFS after each successful data packet transmission.

18:0	TLEN[18:0]	<p>Transfer length</p> <p>The total data bytes number of a transfer.</p> <p>For OUT transfers, this field is the total data bytes of all the data packets desired to be transmitted in an OUT transfer. Software should program this field before the channel is enabled. When software successfully writes a packet into the channel's data TxFIFO, this field is decreased by the byte size of the packet.</p> <p>For IN transfer each time software or DMA reads out a packet from the RxFIFO, this field is decreased by the byte size of the packet.</p>
------	------------	---

### 21.7.3. Device control and status registers

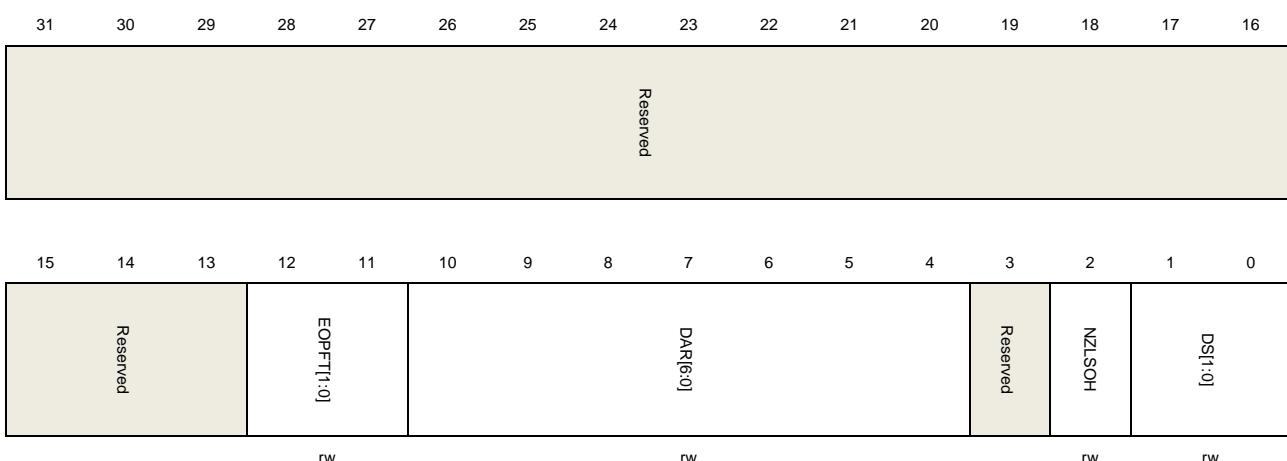
#### Device configuration register (USBFS\_DCFG)

Address offset: 0x0800

Reset value: 0x0000 0000

This register configures the core in device mode after power on or after certain control commands or enumeration. Do not change this register after device initialization.

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



Bits	Fields	Descriptions
31:13	Reserved	Must be kept at reset value
12:11	EOPFT[1:0]	<p>End of periodic frame time</p> <p>This field defines the percentage time point in a frame that the end of periodic frame (EOPF) flag should be triggered.</p> <p>00: 80% of the frame time</p> <p>01: 85% of the frame time</p>

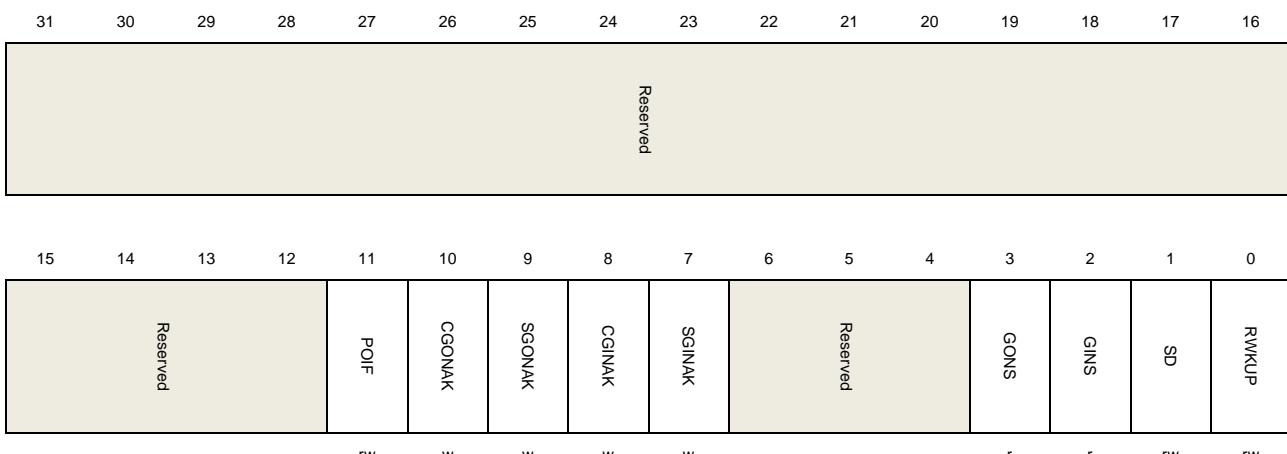
		10: 90% of the frame time 11: 95% of the frame time
10:4	DAR[6:0]	Device address  This field defines the USB device's address. USBFS uses this field to match with the incoming token's device address field. Software should program this field after receiving a Set Address command from USB host.
3	Reserved	Must be kept at reset value
2	NZLSOH	Non-zero-length status OUT handshake  When a USB device receives a non-zero-length data packet during status OUT stage, this field controls that either USBFS should receive this packet or reject this packet with a STALL handshake.  0: Treat this packet as a normal packet and response according to the status of NAKS and STALL bits in USBFS_DOEPxCTL register. 1: Send a STALL handshake and don't save the received OUT packet.
1:0	DS[1:0]	Device speed  This field controls the device speed when the device connected to a host. 11: Full speed Others: Reserved

### Device control register (USBFS\_DCTL)

Address offset: 0x0804

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	POIF	Power-on initialization finished

		Software should set this bit to notify USBFS that the registers are initialized after waking up from power down state.
10	CGONAK	<p>Clear global OUT NAK</p> <p>Software sets this bit to clear GONS bit in this register.</p>
9	SGONAK	<p>Set global OUT NAK</p> <p>Software sets this bit to set GONS bit in this register.</p> <p>When GONS bit is zero, setting this bit will also cause GONAK flag in USBFS_GINTF register triggered after a while. Software should clear the GONAK flag before writing this bit again.</p>
8	CGINAK	<p>Clear global IN NAK</p> <p>Software sets this bit to clear GINS bit in this register.</p>
7	SGINAK	<p>Set global IN NAK</p> <p>Software sets this bit to set GINS bit in this register.</p> <p>When GINS bit is zero, setting this bit will also cause GINAK flag in USBFS_GINTF register triggered after a while. Software should clear the GINAK flag before writing this bit again.</p>
6:4	Reserved	Must be kept at reset value
3	GONS	<p>Global OUT NAK status</p> <p>0: The handshake that USBFS response to OUT transaction packet and whether to save the OUT data packet are decided by Rx FIFO status, endpoint's NAK and STALL bits.</p> <p>1: USHBS always responses to OUT transaction with NAK handshake and doesn't save the incoming OUT data packet.</p>
2	GINS	<p>Global IN NAK status</p> <p>0: The response to IN transaction is decided by Tx FIFO status, endpoint's NAK and STALL bits.</p> <p>1: USBFS always responses to IN transaction with a NAK handshake.</p>
1	SD	<p>Soft disconnect</p> <p>Software can use this bit to generate a soft disconnect condition on USB bus. After this bit is set, USBFS switches off the pull up resistor on DP line. This will cause the host to detect a device disconnect.</p> <p>0: No soft disconnect generated.</p> <p>1: Generate a soft disconnection.</p>
0	RWKUP	<p>Remote wakeup</p> <p>In suspend state, software can use this bit to generate a Remote wake up signal to inform host that it should resume the USB bus.</p> <p>0: No remote wakeup signal generated.</p> <p>1: Generate remote wakeup signal.</p>

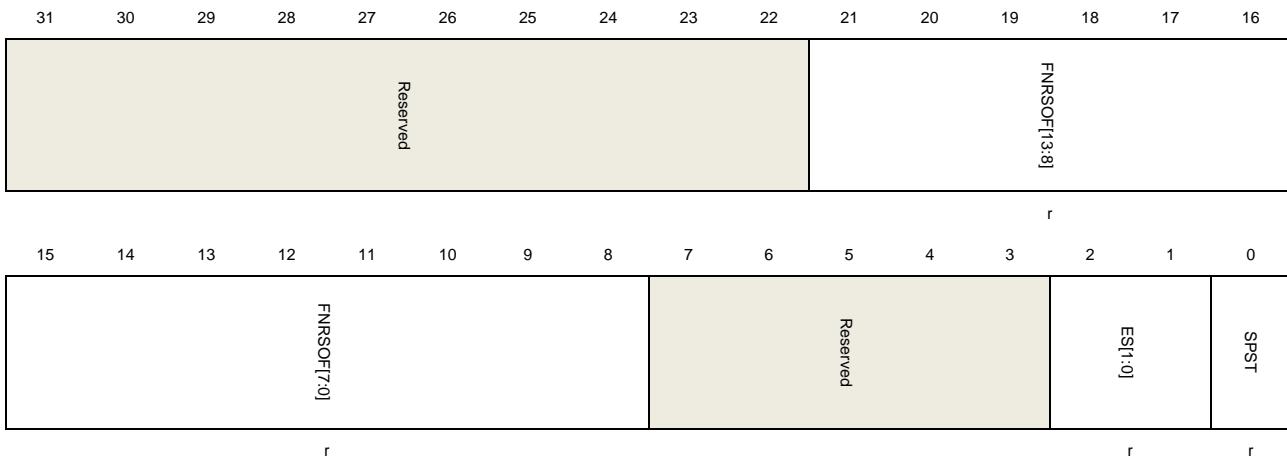
### Device status register (USBFS\_DSTAT)

Address offset: 0x0808

Reset value: 0x0000 0000

This register contains status and information of the USBFS in device mode.

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



Bits	Fields	Descriptions
31:22	Reserved	Must be kept at reset value
21:8	FNRSOF[13:0]	The frame number of the received SOF. USBFS always update this field after receiving a SOF token
7:3	Reserved	Must be kept at reset value
2:1	ES[1:0]	Enumerated speed This field reports the enumerated device speed. Read this field after the ENUMF flag in USBFS_GINTF register is triggered. 11: Full speed Others: reserved
0	SPST	Suspend status This bit reports whether device is in suspend state. 0: Device is not in suspend state. 1: Device is in suspend state.

### Device IN endpoint common interrupt enable register (USBFS\_DIEPINTEN)

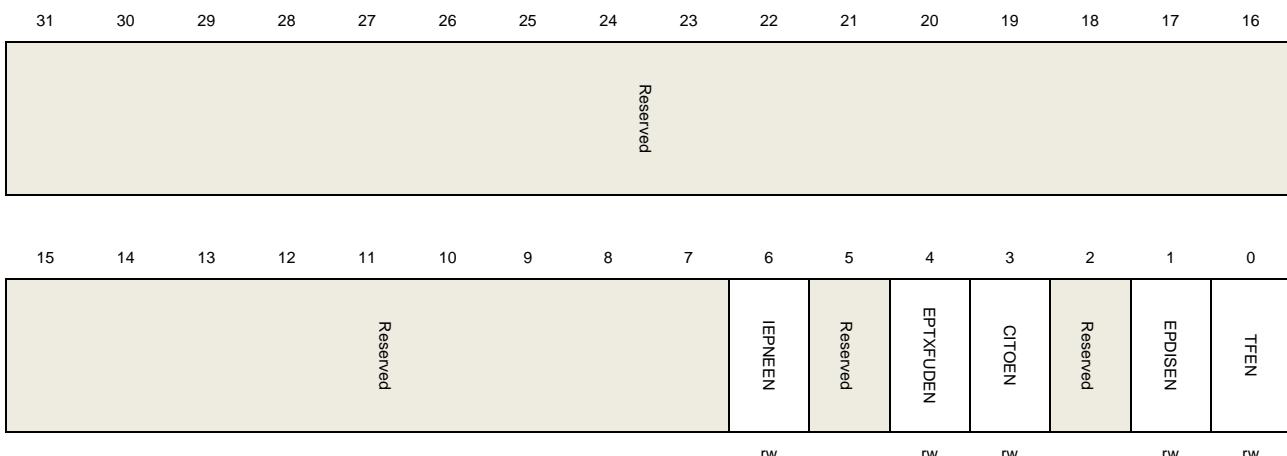
Address offset: 0x810

Reset value: 0x0000 0000

This register contains the interrupt enable bits for the flags in USBFS\_DIEPxINTF register. If a bit in this register is set by software, the corresponding bit in USBFS\_DIEPxINTF register is able to trigger an endpoint interrupt in USBFS\_DAEPINT register. The bits in this register

are set and cleared by software.

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



Bits	Fields	Descriptions
31:7	Reserved	Must be kept at reset value
6	IEPNEEN	IN endpoint NAK effective interrupt enable bit 0: Disable IN endpoint NAK effective interrupt 1: Enable IN endpoint NAK effective interrupt
5	Reserved	Must be kept at reset value
4	EPTXFUDEN	Endpoint Tx FIFO underrun interrupt enable bit 0: Disable endpoint Tx FIFO underrun interrupt 1: Enable endpoint Tx FIFO underrun interrupt
3	CITOEN	Control In timeout interrupt enable bit 0: Disable control In timeout interrupt 1: Enable control In timeout interrupt
2	Reserved	Must be kept at reset value
1	EPDISEN	Endpoint disabled interrupt enable bit 0: Disable endpoint disabled interrupt 1: Enable endpoint disabled interrupt
0	TFEN	Transfer finished interrupt enable bit 0: Disable transfer finished interrupt 1: Enable transfer finished interrupt

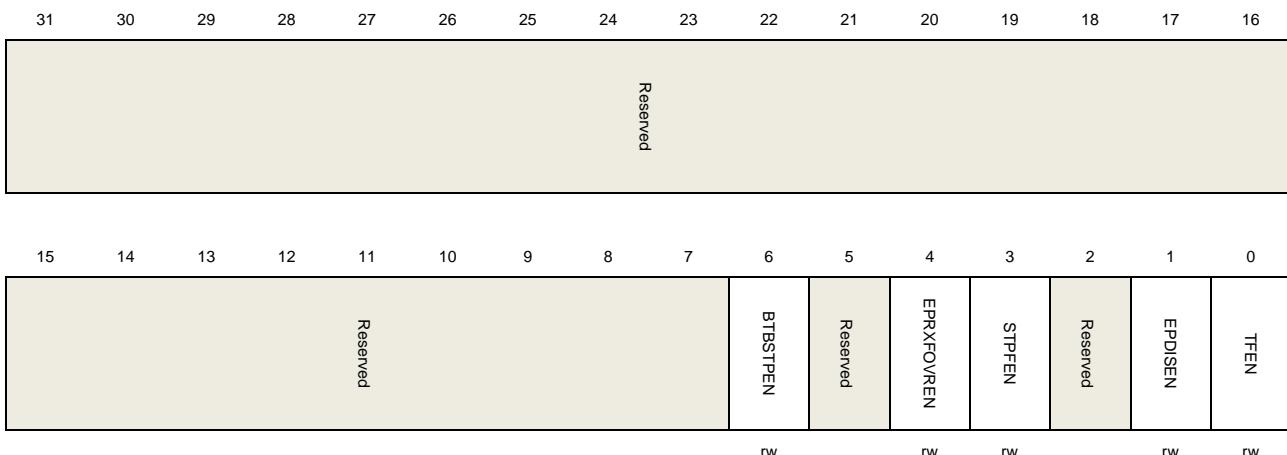
### Device OUT endpoint common interrupt enable register (USBFS\_DOEPINTEN)

Address offset: 0x0814

Reset value: 0x0000 0000

This register contains the interrupt enable bits for the flags in USBFS\_DOEPxINTF register. If a bit in this register is set by software, the corresponding bit in USBFS\_DOEPxINTF register is able to trigger an endpoint interrupt in USBFS\_DAEPINT register. The bits in this register are set and cleared by software.

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



Bits	Fields	Descriptions
31:7	Reserved	Must be kept at reset value
6	BTBSTPEN	Back-to-back SETUP packets ( Only for control OUT endpoint) interrupt enable bit 0: Disable back-to-back SETUP packets interrupt 1: Enable back-to-back SETUP packets interrupt
5	Reserved	Must be kept at reset value
4	EPRXFOVREN	Endpoint Rx FIFO overrun interrupt enable bit 0: Disable endpoint Rx FIFO overrun interrupt 1: Enable endpoint Rx FIFO overrun interrupt
3	STPFEN	SETUP phase finished (Only for control OUT endpoint) interrupt enable bit 0: Disable SETUP phase finished interrupt 1: Enable SETUP phase finished interrupt
2	Reserved	Must be kept at reset value
1	EPDISEN	Endpoint disabled interrupt enable bit 0: Disable endpoint disabled interrupt 1: Enable endpoint disabled interrupt
0	TFEN	Transfer finished interrupt enable bit 0: Disable transfer finished interrupt 1: Enable transfer finished interrupt

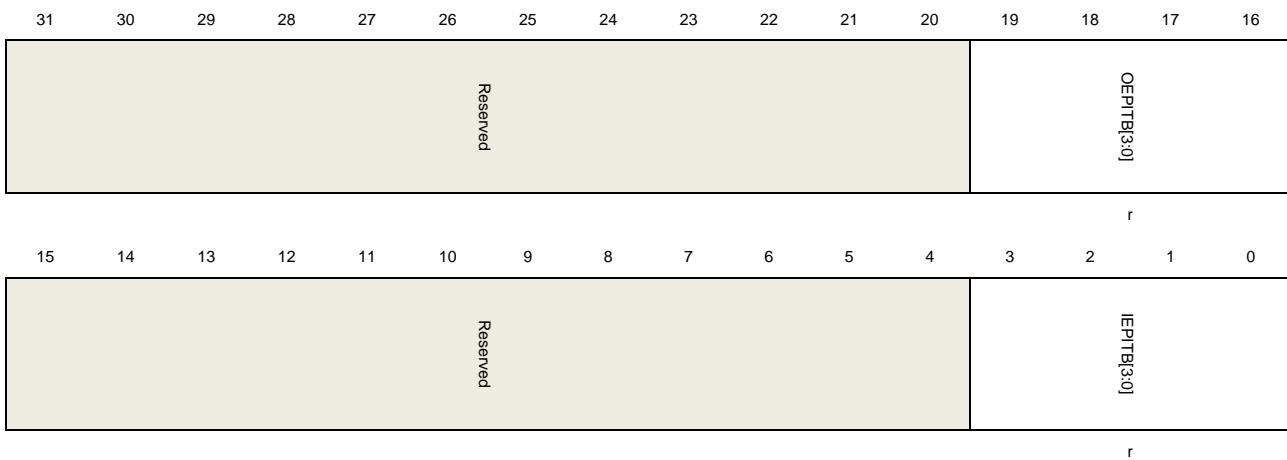
### **Device all endpoints interrupt register (USBFS\_DAEPINT)**

Address offset: 0x0818

Reset value: 0x0000 0000

When an endpoint interrupt is triggered, USBFS sets corresponding bit in this register and software should read this register to know which endpoint is asserting an interrupt.

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



<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:20	Reserved	Must be kept at reset value
19:16	OEPITB[3:0]	Device all OUT endpoint interrupt bits Each bit represents an OUT endpoint: Bit 16 for OUT endpoint 0, bit 19 for OUT endpoint 3.
15:4	Reserved	Must be kept at reset value
3:0	IEPITB[3:0]	Device all IN endpoint interrupt bits Each bit represents an IN endpoint: Bit 0 for IN endpoint 0, bit 3 for IN endpoint 3.

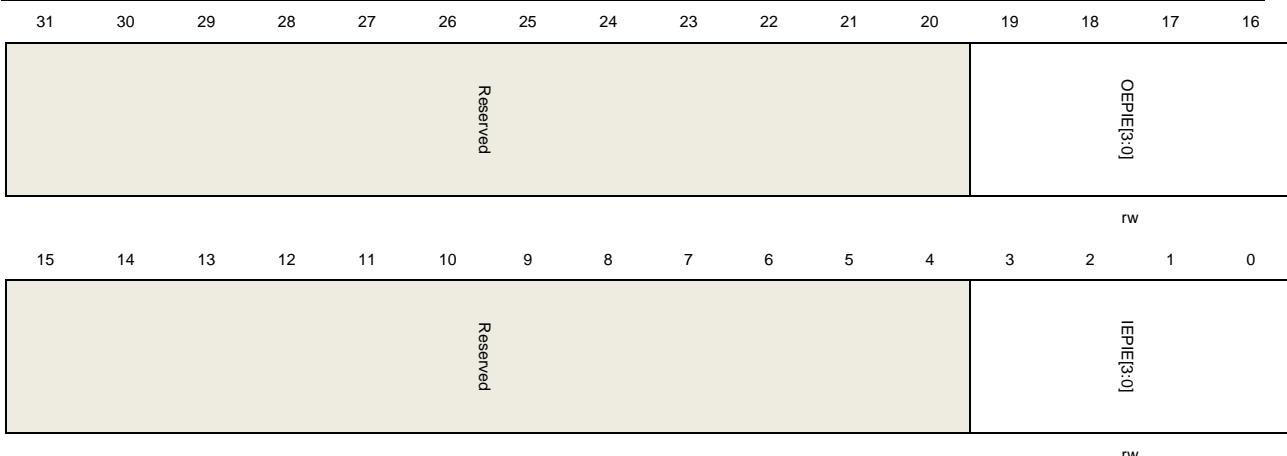
### **Device all endpoints interrupt enable register (USBFS\_DAEPINTEN)**

Address offset: 0x081C

Reset value: 0x0000 0000

This register can be used by software to enable or disable an endpoint's interrupt. Only the endpoint whose corresponding bit in this register is set is able to cause the endpoint interrupt flag OEIF or IEPIF in USBFS\_GINTF register.

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



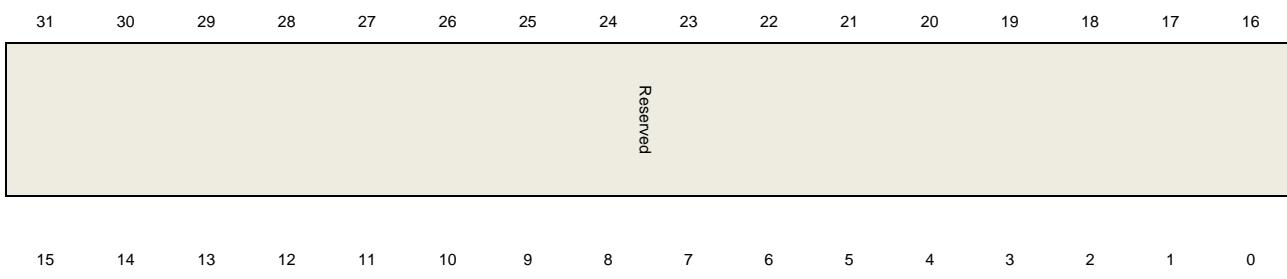
Bits	Fields	Descriptions
31:20	Reserved	Must be kept at reset value
19:16	OEPIE[3:0]	Out endpoint interrupt enable 0: Disable OUT endpoint-n interrupt 1: Enable OUT endpoint-n interrupt Each bit represents an OUT endpoint: Bit 16 for OUT endpoint 0, bit 19 for OUT endpoint 3.
15:4	Reserved	Must be kept at reset value
3:0	IEPIE[3:0]	IN endpoint interrupt enable bits 0: Disable IN endpoint-n interrupt 1: Enable IN endpoint-n interrupt Each bit represents an IN endpoint: Bit 0 for IN endpoint 0, bit 3 for IN endpoint 3.

### Device VBUS discharge time register (USBFS\_DVBUSDT)

Address offset: 0x0828

Reset value: 0x0000 17D7

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





rw

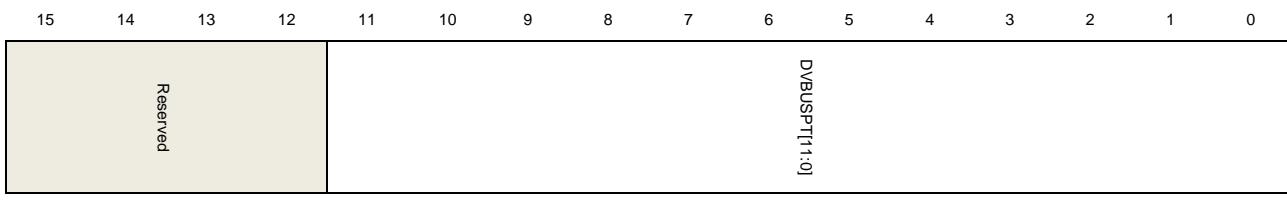
<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:16	Reserved	Must be kept at reset value
15:0	DVBUSDT[15:0]	Device VBUS discharge time There is a discharge process after VBUS pulsing in SRP protocol. This field defines the discharge time of VBUS. The true discharge time is $1024 * \text{DVBUSDT}[15:0] * \text{TUSBCLOCK}$ , where TUSBCLOCK is the period time of USB clock.

### Device VBUS pulsing time register (USBFS\_DVBUSPT)

Address offset: 0x082C

Reset value: 0x0000 05B8

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



rw

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:12	Reserved	Must be kept at reset value
11:0	DVBUSPT[11:0]	Device Vbus pulsing time This field defines the pulsing time for VBUS. The true pulsing time is $1024 * \text{DVBUSPT}[11:0] * \text{TUSBCLOCK}$ , where TUSBCLOCK is the period time of USB clock.

### Device IN endpoint FIFO empty interrupt enable register (USBFS\_DIEPFEINTEN)

Address offset: 0x0834

Reset value: 0x0000 0000

This register contains the enable bits for the Tx FIFO empty interrupts of IN endpoints.

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										IEPTXFEIE[3:0]					

rw

Bits	Fields	Descriptions
31:4	Reserved	Must be kept at reset value
3:0	IEPTXFEIE[3:0]	IN endpoint Tx FIFO empty interrupt enable bits This field controls whether the TXFE bits in USBFS_DIEPxINTF registers are able to generate an endpoint interrupt bit in USBFS_DAEPINT register. Bit 0 for IN endpoint 0, bit 3 for IN endpoint 3 0: Disable FIFO empty interrupt 1: Enable FIFO empty interrupt

### Device IN endpoint 0 control register (USBFS\_DIEP0CTL)

Address offset: 0x0900

Reset value: 0x0000 8000

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
EPEN	EPD	Reserved		SNAK	CNAK	TXFNUM[3:0]				STALL	Reserved		EPTYPE[1:0]	Reserved	
r	r			w	w			rw		r		r		r	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EPACT	Reserved										MPL[1:0]				

rw

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31	EPEN	<p>Endpoint enable</p> <p>Set by the application and cleared by USBFS.</p> <p>0: Endpoint disabled</p> <p>1: Endpoint enabled</p> <p>Software should follow the operation guide to disable or enable an endpoint.</p>
30	EPD	<p>Endpoint disable</p> <p>Software can set this bit to disable the endpoint. Software should following the operation guide to disable or enable an endpoint.</p>
29:28	Reserved	Must be kept at reset value
27	SNAK	<p>Set NAK</p> <p>Software sets this bit to set NAKS bit in this register.</p>
26	CNAK	<p>Clear NAK</p> <p>Software sets this bit to clear NAKS bit in this register.</p>
25:22	TXFNUM[3:0]	<p>Tx FIFO number</p> <p>Defines the Tx FIFO number of IN endpoint 0.</p>
21	STALL	<p>STALL handshake</p> <p>Software can set this bit to make USBFS sends STALL handshake when receiving IN token. USBFS will clear this bit after a SETUP token is received on the corresponding OUT endpoint 0. This bit has a higher priority than NAKS bit in this register and GINS bit in USBFS_DCTL register. If both STALL and NAKS bits are set, the STALL bit takes effect.</p>
20	Reserved	Must be kept at reset value
19:18	EPTYPE[1:0]	<p>Endpoint type</p> <p>This field is fixed to '00' for control endpoint.</p>
17	NAKS	<p>NAK status</p> <p>This bit controls the NAK status of USBFS when both STALL bit in this register and GINS bit in USBFS_DCTL register are cleared:</p> <p>0: USBFS sends data or handshake packets according to the status of the endpoint's Tx FIFO.</p> <p>1: USBFS always sends NAK handshake to the IN token.</p> <p>This bit is read-only and software should use CNAK and SNAK in this register to control this bit.</p>
16	Reserved	Must be kept at reset value
15	EPACT	<p>Endpoint active</p> <p>This field is fixed to '1' for endpoint 0.</p>
14:2	Reserved	Must be kept at reset value

---

1:0	MPL[1:0]	Maximum packet length This field defines the maximum packet length for a control data packet. As described in USB 2.0 protocol, there are 4 kinds of length for control transfers: 00: 64 bytes 01: 32 bytes 10: 16 bytes 11: 8 bytes
-----	----------	--

### Device IN endpoint-x control register (USBFS\_DIEPxCTL) (x = 1..3, where x = endpoint\_number)

Address offset: 0x0900 + (endpoint\_number × 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
EPEN	EPD	SODDFRM/SD1 PID	SD0PID/SEVNF RM	SNAK	CNAK	TXFNUM[3:0]			STALL	Reserved	EPTYPE[1:0]	NAKS	EOFRM/DPID		
rs	rs	w	w	w	w		rw		rw/rs		rw	r	r		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EPACT	Reserved									MPL[10:0]					
rw										rw					

---

Bits	Fields	Descriptions
31	EPEN	Endpoint enable Set by the application and cleared by USBFS. 0: Endpoint disabled 1: Endpoint enabled Software should follow the operation guide to disable or enable an endpoint.
30	EPD	Endpoint disable Software can set this bit to disable the endpoint. Software should following the operation guide to disable or enable an endpoint.
29	SODDFRM	Set odd frame (For isochronous IN endpoints) This bit has effect only if this is an isochronous IN endpoint. Software sets this bit to set EOFRM bit in this register.
	SD1PID	Set DATA1 PID (For interrupt/bulk IN endpoints) Software sets this bit to set DPID bit in this register.

28	SEVENFRM	Set even frame (For isochronous IN endpoints) Software sets this bit to clear EOFRM bit in this register.
	SD0PID	Set DATA0 PID (For interrupt/bulk IN endpoints) Software sets this bit to clear DPID bit in this register.
27	SNAK	Set NAK Software sets this bit to set NAKS bit in this register.
26	CNAK	Clear NAK Software sets this bit to clear NAKS bit in this register.
25:22	TXFNUM[3:0]	Tx FIFO number Defines the Tx FIFO number of this IN endpoint.
21	STALL	STALL handshake Software can set this bit to make USBFS sends STALL handshake when receiving IN token. This bit has a higher priority than NAKS bit in this register and GINS bit in USBFS_DCTL register. If both STALL and NAKS bits are set, the STALL bit takes effect.  For control IN endpoint: Only USBFS can clear this bit when a SETUP token is received on the corresponding OUT endpoint. Software is not able to clear it.  For interrupt or bulk IN endpoint: Only software can clear this bit
20	Reserved	Must be kept at reset value
19:18	EPTYPE[1:0]	Endpoint type This field defines the transfer type of this endpoint: 00: Control 01: Isochronous 10: Bulk 11: Interrupt
17	NAKS	NAK status This bit controls the NAK status of USBFS when both STALL bit in this register and GINS bit in USBFS_DCTL register are cleared: 0: USBFS sends data or handshake packets according to the status of the endpoint's Tx FIFO. 1: USBFS always sends NAK handshake to the IN token. This bit is read-only and software should use CNAK and SNAK in this register to control this bit.
16	EOFRM	Even/odd frame (For isochronous IN endpoints) For isochronous transfers, software can use this bit to control that USBFS only sends data packets for IN tokens in even or odd frames. If the parity of the current frame number doesn't match with this bit, USBFS only responses with a zero-length packet.

		0: Only sends data in even frames 1: Only sends data in odd frames
	DPID	Endpoint data PID (For interrupt/bulk IN endpoints)  There is a data PID toggle scheme in interrupt or bulk transfer. Set SD0PID to set this bit before a transfer starts and USBFS maintains this bit during transfers according to the data toggle scheme described in USB protocol.
		0: Data packet's PID is DATA0 1: Data packet's PID is DATA1
15	EPACT	Endpoint active  This bit controls whether this endpoint is active. If an endpoint is not active, it ignores all tokens and doesn't make any response.
14:11	Reserved	Must be kept at reset value
10:0	MPL[10:0]	This field defines the maximum packet length in bytes.

### Device OUT endpoint 0 control register (USBFS\_DOEP0CTL)

Address offset: 0x0B00

Reset value: 0x0000 8000

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
EPEN	EPD	Reserved.		SNAK	CNAK		Reserved		STALL		SNOOP	EPTYPE[1:0]		NAKS	Reserved
rs	r		w	w				rs		rw		r		r	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EPACT							Reserved							MPL[1:0]	
r														r	

Bits	Fields	Descriptions
31	EPEN	Endpoint enable  Set by the application and cleared by USBFS. 0: Endpoint disabled 1: Endpoint enabled  Software should follow the operation guide to disable or enable an endpoint.
30	EPD	Endpoint disable  This bit is fixed to 0 for OUT endpoint 0.

---

29:28	Reserved	Must be kept at reset value
27	SNAK	<p>Set NAK</p> <p>Software sets this bit to set NAKS bit in this register.</p>
26	CNAK	<p>Clear NAK</p> <p>Software sets this bit to clear NAKS bit in this register</p>
25:22	Reserved	Must be kept at reset value
21	STALL	<p>STALL handshake</p> <p>Set this bit to make USBFS send STALL handshake during an OUT transaction.</p> <p>USBFS will clear this bit after a SETUP token is received on OUT endpoint 0. This bit has a higher priority than NAKS bit in this register, i.e. if both STALL and NAKS bits are set, the STALL bit takes effect.</p>
20	SNOOP	<p>Snoop mode</p> <p>This bit controls the snoop mode of an OUT endpoint. In snoop mode, USBFS doesn't check the received data packet's CRC value.</p> <p>0:Snoop mode disabled</p> <p>1:Snoop mode enabled</p>
19:18	EPTYPE[1:0]	<p>Endpoint type</p> <p>This field is fixed to '00' for control endpoint.</p>
17	NAKS	<p>NAK status</p> <p>This bit controls the NAK status of USBFS when both STALL bit in this register and GONS bit in USBFS_DCTL register are cleared:</p> <p>0: USBFS sends data or handshake packets according to the status of the endpoint's Rx FIFO.</p> <p>1: USBFS always sends NAK handshake for the OUT token.</p> <p>This bit is read-only and software should use CNAK and SNAK in this register to control this bit.</p>
16	Reserved	Must be kept at reset value
15	EPACT	<p>Endpoint active</p> <p>This field is fixed to '1' for endpoint 0.</p>
14:2	Reserved	Must be kept at reset value
1:0	MPL[1:0]	<p>Maximum packet length</p> <p>This is a read-only field, and its value comes from the MPL field of USBFS_DIEP0CTL register:</p> <p>00: 64 bytes</p> <p>01: 32 bytes</p> <p>10: 16 bytes</p> <p>11: 8 bytes</p>

### Device OUT endpoint-x control register (USBFS\_DOEPxCTL) (x = 1..3, where x = endpoint\_number)

Address offset: 0x0B00 + (endpoint\_number × 0x20)

Reset value: 0x0000 0000

The application uses this register to control the operations of each logical OUT endpoint other than OUT endpoint 0.

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
EPEN	EPD	SODDFRM/SD1PID	SEVENFRM/SD0PID	SNAK	CNAK	Reserved				STALL	SNOOP	EPTYPE[1:0]	NAKS	EOFRM/DPID	
rs	rs	w	w	w	w					rw/rs	rw	rw	r	r	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EPACT	Reserved				MPI[10:0]										

Bits	Fields	Descriptions
31	EPEN	Endpoint enable Set by the application and cleared by USBFS. 0: Endpoint disabled 1: Endpoint enabled Software should follow the operation guide to disable or enable an endpoint.
30	EPD	Endpoint disable Software can set this bit to disable the endpoint. Software should follow the operation guide to disable or enable an endpoint.
29	SODDFRM SD1PID	Set odd frame (For isochronous OUT endpoints) This bit has effect only if this is an isochronous OUT endpoint. Software sets this bit to set EOFRM bit in this register. Set DATA1 PID (For interrupt/bulk OUT endpoints) Software sets this bit to set DPID bit in this register.
28	SEVENFRM SD0PID	Set even frame (For isochronous OUT endpoints) Software sets this bit to clear EOFRM bit in this register. Set DATA0 PID (For interrupt/bulk OUT endpoints) Software sets this bit to clear DPID bit in this register.
27	SNAK	Set NAK

		Software sets this bit to set NAKS bit in this register.
26	CNAK	<p>Clear NAK</p> <p>Software sets this bit to clear NAKS bit in this register.</p>
25:22	Reserved	Must be kept at reset value
21	STALL	<p>STALL handshake</p> <p>Software can set this bit to make USBFS sends STALL handshake during an OUT transaction. This bit has a higher priority than NAKS bit in this register and GINAK in USBFS_DCTL register. If both STALL and NAKS bits are set, the STALL bit takes effect.</p> <p>For control OUT endpoint:</p> <p>Only USBFS can clear this bit when a SETUP token is received on the corresponding OUT endpoint. Software is not able to clear it.</p> <p>For interrupt or bulk OUT endpoint:</p> <p>Only software can clear this bit.</p>
20	SNOOP	<p>Snoop mode</p> <p>This bit controls the snoop mode of an OUT endpoint. In snoop mode, USBFS doesn't check the received data packet's CRC value.</p> <p>0:Snoop mode disabled</p> <p>1:Snoop mode enabled</p>
19:18	EPTYPE[1:0]	<p>Endpoint type</p> <p>This field defines the transfer type of this endpoint:</p> <p>00: Control</p> <p>01: Isochronous</p> <p>10: Bulk</p> <p>11: Interrupt</p>
17	NAKS	<p>NAK status</p> <p>This bit controls the NAK status of USBFS when both STALL bit in this register and GONS bit in USBFS_DCTL register are cleared:</p> <p>0: USBFS sends handshake packets according to the status of the endpoint's Rx FIFO.</p> <p>1: USBFS always sends NAK handshake to the OUT token.</p> <p>This bit is read-only and software should use CNAK and SNAK in this register to control this bit.</p>
16	EOFRM	<p>Even/odd frame (For isochronous OUT endpoints)</p> <p>For isochronous transfers, software can use this bit to control that USBFS only receives data packets in even or odd frames. If the current frame number's parity doesn't match with this bit, USBFS just drops the data packet.</p> <p>0: Only sends data in even frames</p> <p>1: Only sends data in odd frames</p>
	DPID	Endpoint data PID (For interrupt/bulk OUT endpoints)

These is a data PID toggle scheme in interrupt or bulk transfer. Software should set SD0PID to set this bit before a transfer starts and USBFS maintains this bit during transfers following the data toggle scheme described in USB protocol.

0: Data packet's PID is DATA0

1: Data packet's PID is DATA1

15	EPACT	Endpoint active
14:11	Reserved	Must be kept at reset value
10:0	MPL[10:0]	This field defines the maximum packet length in bytes.

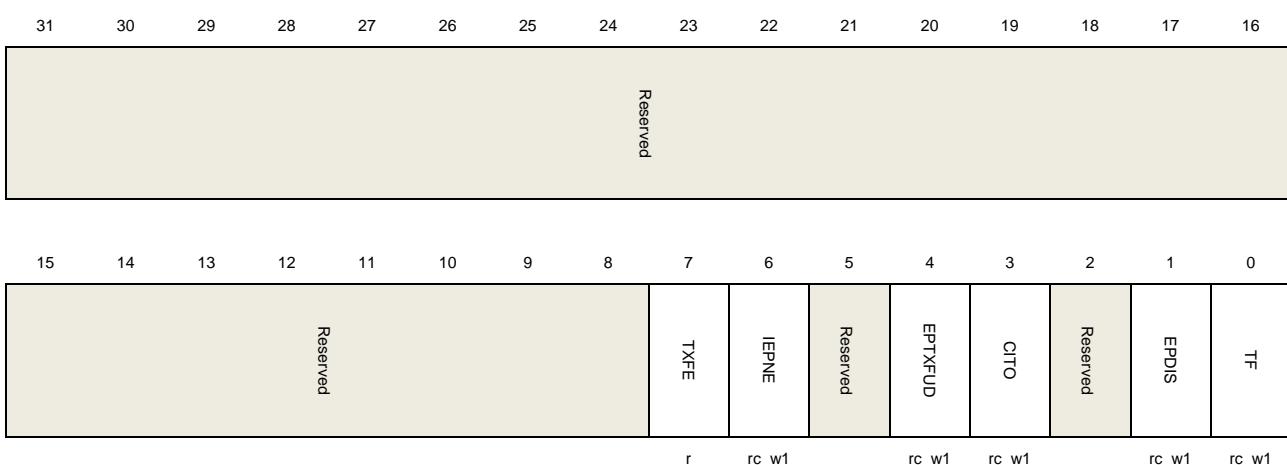
### Device IN endpoint-x interrupt flag register (USBFS\_DIEPxINTF) (x = 0..3, where x = endpoint\_number)

Address offset: 0x0908 + (endpoint\_number × 0x20)

Reset value: 0x0000 0080

This register contains the status and events of an IN endpoint, when an IN endpoint interrupt occurs, read this register for the respective endpoint to know the source of the interrupt. The flag bits in this register are all set by hardware and cleared by writing 1 except the read-only TXFE bit.

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



Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value
7	TXFE	Transmit FIFO empty The Tx FIFO of this IN endpoint has reached the empty threshold value defined by TXFTH field in USBFS_GAHBCS register.
6	IEPNE	IN endpoint NAK effective

The setting of SNAK bit in USBFS\_DIEPxCTL register takes effect. This bit can be cleared either by writing 1 to it or by setting CNAK bit in USBFS\_DIEPxCTL register.

5	Reserved	Must be kept at reset value
4	EPTXFUD	Endpoint Tx FIFO underrun This flag is triggered if the Tx FIFO has no packet data when an IN token is incoming
3	CITO	Control In Timeout interrupt This flag is triggered if the device waiting for a handshake is timeout in a control IN transaction.
2	Reserved	Must be kept at reset value
1	EPDIS	Endpoint disabled This flag is triggered when an endpoint is disabled by the software's request.
0	TF	Transfer finished This flag is triggered when all the IN transactions assigned to this endpoint have been finished.

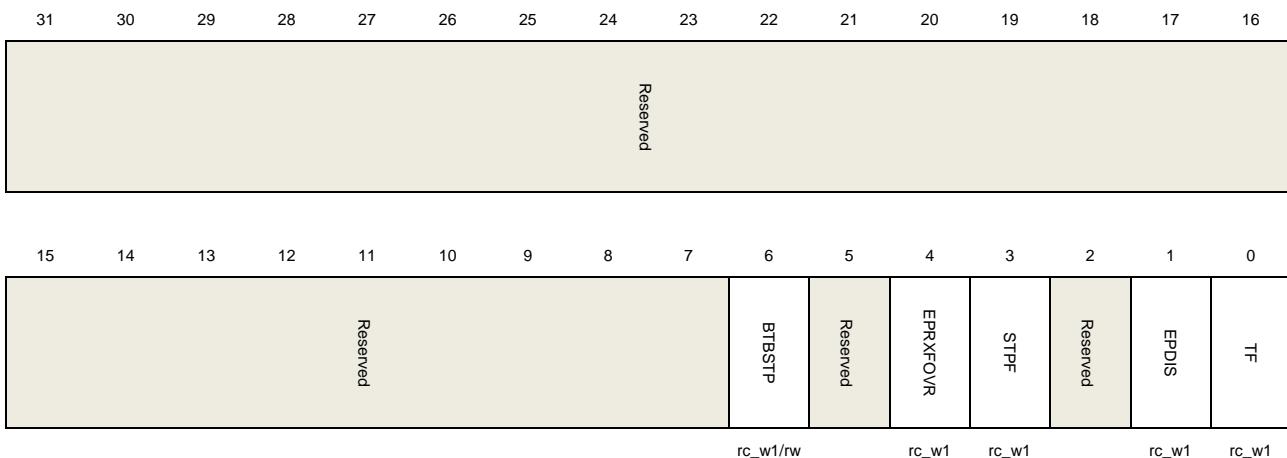
### Device OUT endpoint-x interrupt flag register (USBFS\_DOEPxINTF) (x = 0..3, where x = endpoint\_number)

Address offset: 0x0B08 + (endpoint\_number × 0x20)

Reset value: 0x0000 0000

This register contains the status and events of an OUT endpoint, when an OUT endpoint interrupt occurs, read this register for the respective endpoint to know the source of the interrupt. The flag bits in this register are all set by hardware and cleared by writing 1.

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



Bits	Fields	Descriptions
31:7	Reserved	Must be kept at reset value

---

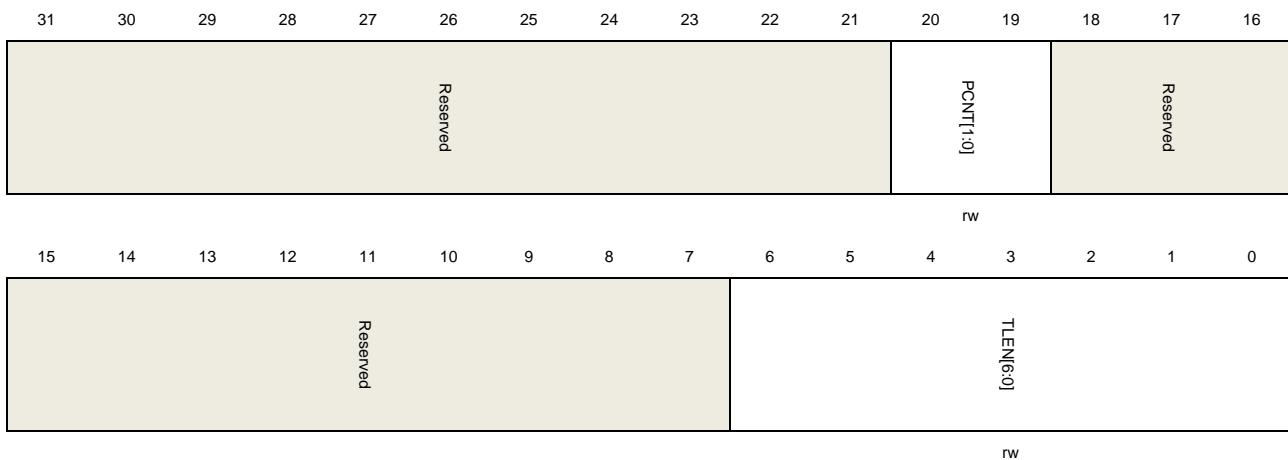
6	BTBSTP	Back-to-back SETUP packets (Only for control OUT endpoint) This flag is triggered when a control out endpoint has received more than 3 back-to-back setup packets.
5	Reserved	Must be kept at reset value
4	EPRXFOVR	Endpoint Rx FIFO overrun This flag is triggered if the OUT endpoint's Rx FIFO has no enough space for a packet data when an OUT token is incoming. USBFS will drop the incoming OUT data packet and sends a NAK handshake in this case.
3	STPF	SETUP phase finished (Only for control OUT endpoint) This flag is triggered when a setup phase finished, i.e. USBFS receives an IN or OUT token after a setup token.
2	Reserved	Must be kept at reset value
1	EPDIS	Endpoint disabled This flag is triggered when an endpoint is disabled by the software's request.
0	TF	Transfer finished This flag is triggered when all the OUT transactions assigned to this endpoint have been finished.

### Device IN endpoint 0 transfer length register (USBFS\_DIEP0LEN)

Address offset: 0x0910

Reset value: 0x0000 0000

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




---

Bits	Fields	Descriptions
31:21	Reserved	Must be kept at reset value
20:19	PCNT[1:0]	Packet count

The number of data packets desired to be transmitted in a transfer.

Program this field before the endpoint is enabled. After the transfer starts, this field is decreased automatically by USBFS after each successful data packet transmission.

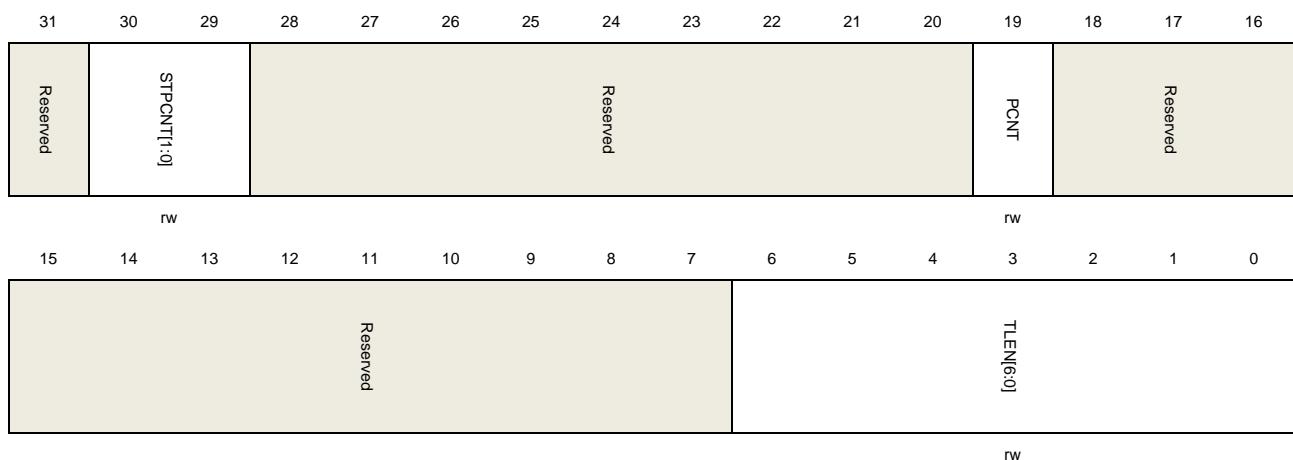
18:7	Reserved	Must be kept at reset value
6:0	TLEN[6:0]	<p>Transfer length</p> <p>The total data bytes number of a transfer.</p> <p>This field is the total data bytes of all the data packets desired to be transmitted in an IN transfer. Program this field before the endpoint is enabled. When software successfully writes a packet into the endpoint's Tx FIFO, this field is decreased by the byte size of the packet.</p>

### Device OUT endpoint 0 transfer length register (USBFS\_DOEP0LEN)

Address offset: 0x0B10

Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31	Reserved	Must be kept at reset value
30:29	STPCNT[1:0]	<p>SETUP packet count</p> <p>This field defines the maximum number of back-to-back SETUP packets this endpoint can accept.</p> <p>Program this field before setup transfers. Each time a back-to-back setup packet is received, USBFS decrease this field by one. When this field reaches zero, the BTBSTP flag in USBFS_DOEP0INTF register will be triggered.</p> <p>00: 0 packet 01: 1 packet 10: 2 packets</p>

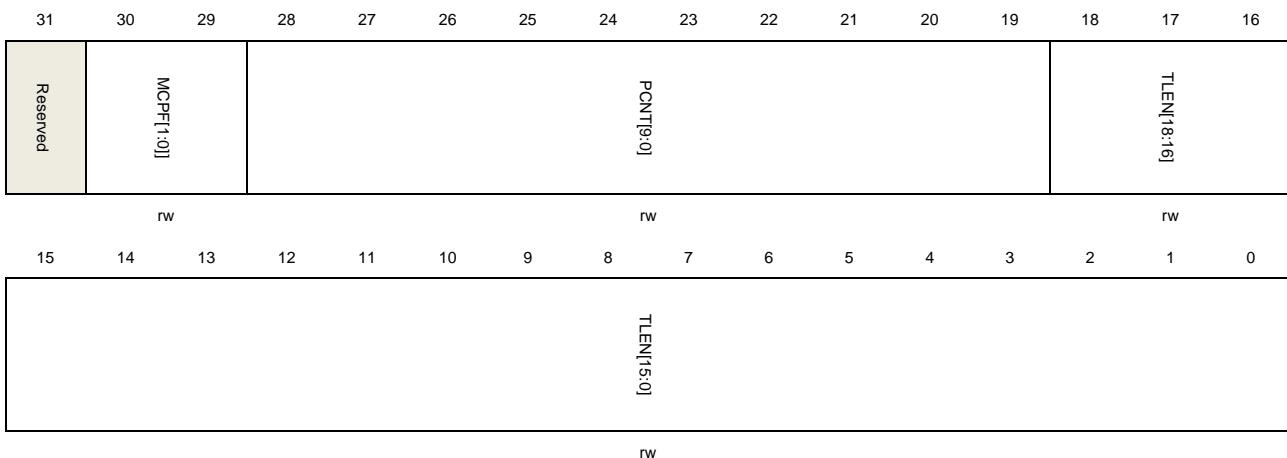
		11: 3 packets
28:20	Reserved	Must be kept at reset value
19	PCNT	<p>Packet count</p> <p>The number of data packets desired to receive in a transfer.</p> <p>Program this field before the endpoint is enabled. After the transfer starts, this field is decreased automatically by USBFS after each successful data packet reception on bus.</p>
18:7	Reserved	Must be kept at reset value
6:0	TLEN[6:0]	<p>Transfer length</p> <p>The total data bytes number of a transfer.</p> <p>This field is the total data bytes of all the data packets desired to receive in an OUT transfer. Program this field before the endpoint is enabled. Each time software reads out a packet from the Rx FIFO, this field is decreased by the byte size of the packet.</p>

**Device IN endpoint-x transfer length register (USBFS\_DIEPxLEN) (x = 1..3,  
where x = endpoint\_number)**

Address offset: 0x910 + (endpoint\_number × 0x20)

Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31	Reserved	Must be kept at reset value
30:29	MCPF[1:0]	<p>Multi packet count per frame</p> <p>This field indicates the packet count that must be transmitted per frame for periodic IN endpoints on the USB. It is used to calculate the data PID for isochronous IN endpoints by the core.</p> <p>01: 1 packet</p>

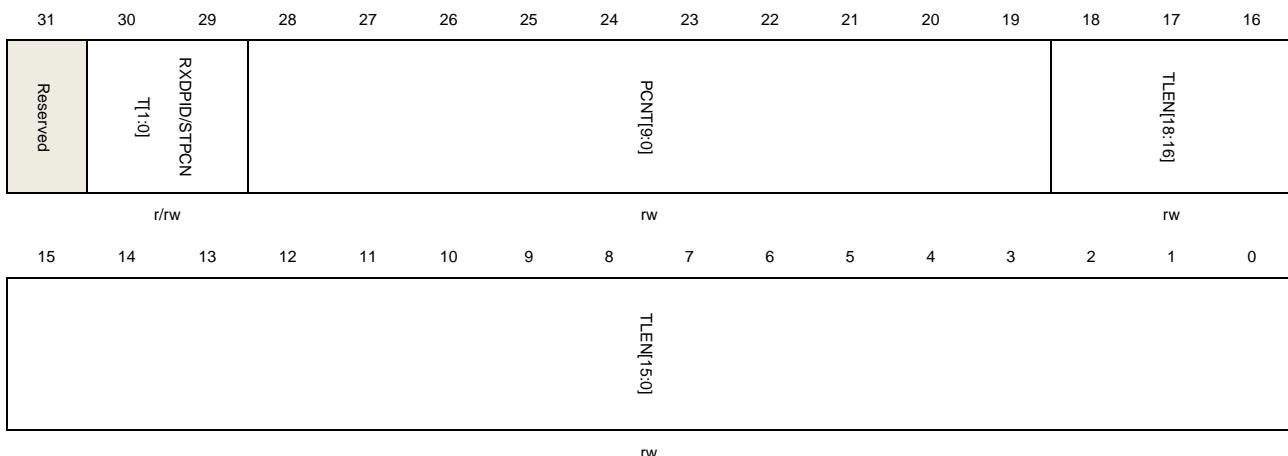
		10: 2 packets 11: 3 packets
28:19	PCNT[9:0]	<p>Packet count</p> <p>The number of data packets desired to be transmitted in a transfer.</p> <p>Program this field before the endpoint is enabled. After the transfer starts, this field is decreased automatically by USBFS after each successful data packet transmission.</p>
18:0	TLEN[18:0]	<p>Transfer length</p> <p>The total data bytes number of a transfer.</p> <p>This field is the total data bytes of all the data packets desired to be transmitted in an IN transfer. Program this field before the endpoint is enabled. When software successfully writes a packet into the endpoint's Tx FIFO, this field is decreased by the byte size of the packet.</p>

**Device OUT endpoint-x transfer length register (USBFS\_DOEPxLEN) (x = 1..3,  
where x = endpoint\_number)**

Address offset: 0x0B10 + (endpoint\_number × 0x20)

Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31	Reserved	Must be kept at reset value
30:29	RXDPID[1:0]	<p>Received data PID (For isochronous OUT endpoints)</p> <p>This field saves the PID of the latest received data packet on this endpoint.</p> <p>00: DATA0 10: DATA1 Others: Reserved</p>
	STPCNT[1:0]	SETUP packet count (For control OUT Endpoints.)

This field defines the maximum number of back-to-back SETUP packets this endpoint can accept.

Program this field before setup transfers. Each time a back-to-back setup packet is received, USBFS decrease this field by one. When this field reaches zero, the BTBSTP flag in USBFS\_DOEPxINTF register will be triggered.

00: 0 packet

01: 1 packet

10: 2 packets

11: 3 packets

28:19	PCNT[9:0]	Packet count The number of data packets desired to receive in a transfer. Program this field before the endpoint is enabled. After the transfer starts, this field is decreased automatically by USBFS after each successful data packet reception on bus.
18:0	TLEN[18:0]	Transfer length The total data bytes number of a transfer. This field is the total data bytes of all the data packets desired to receive in an OUT transfer. Program this field before the endpoint is enabled. Each time after software reads out a packet from the RxFIFO, this field is decreased by the byte size of the packet.

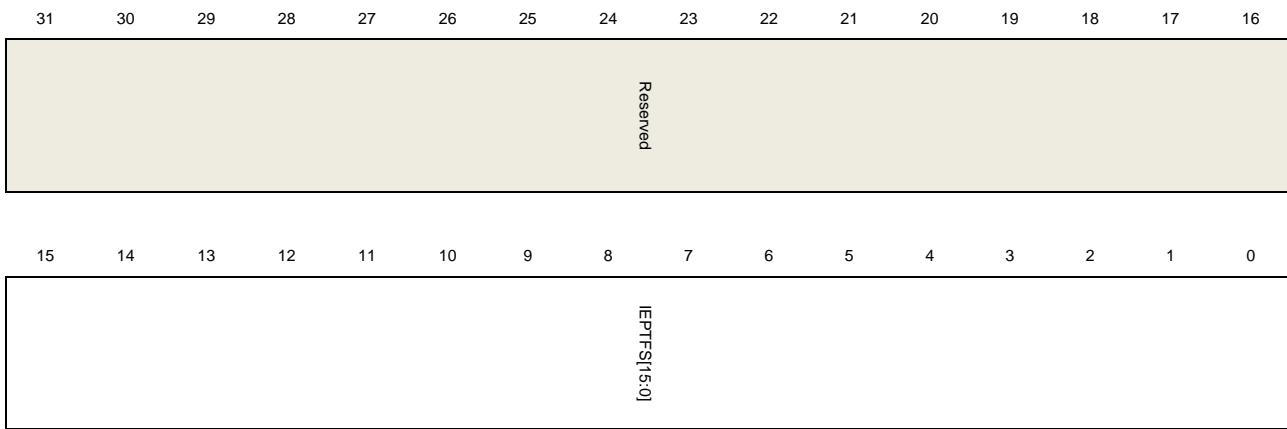
### **Device IN endpoint-x transmit FIFO status register (USBFS\_DIEPxTFSTAT) (x = 0..3, where x = endpoint\_number)**

Address offset: 0x0918 + (endpoint\_number × 0x20)

Reset value: 0x0000 0200

This register contains the information of each endpoint's Tx FIFO.

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



r

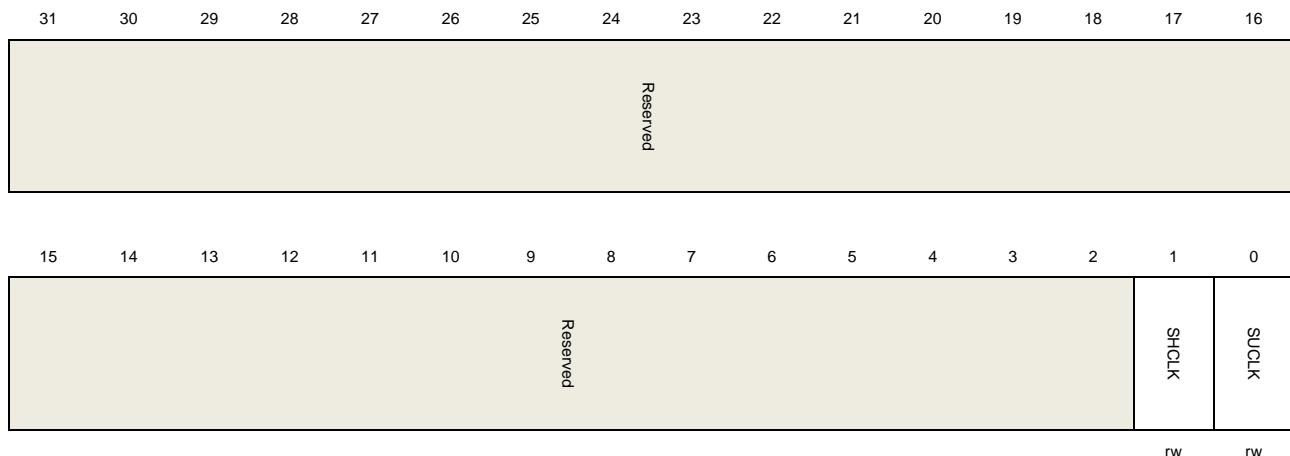
<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:16	Reserved	Must be kept at reset value
15:0	IEPTFS[15:0]	IN endpoint's Tx FIFO space remaining   N endpoint's Tx FIFO space remaining in 32-bit words: 0: FIFO is full 1: 1 word available ... n: n words available

#### 21.7.4. Power and clock control register (USBFS\_PWRCLKCTL)

Address offset: 0x0E00

Reset value: 0x0000 0000

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



<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:2	Reserved	Must be kept at reset value
1	SHCLK	Stop HCLK Stop the HCLK to save power. 0:HCLK is not stopped 1:HCLK is stopped
0	SUCLK	Stop the USB clock Stop the USB clock to save power. 0:USB clock is not stopped 1:USB clock is stopped

## 22. Revision history

**Table 22-1. Revision history**

Revision No.	Description	Date
1.0	Initial Release	Jun.5, 2019

## Important Notice

This document is the property of GigaDevice Semiconductor Inc. and its subsidiaries (the "Company"). This document, including any product of the Company described in this document (the "Product"), is owned by the Company under the intellectual property laws and treaties of the People's Republic of China and other jurisdictions worldwide. The Company reserves all rights under such laws and treaties and does not grant any license under its patents, copyrights, trademarks, or other intellectual property rights. The names and brands of third party referred thereto (if any) are the property of their respective owner and referred to for identification purposes only.

The Company makes no warranty of any kind, express or implied, with regard to this document or any Product, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The Company does not assume any liability arising out of the application or use of any Product described in this document. Any information provided in this document is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Except for customized products which has been expressly identified in the applicable agreement, the Products are designed, developed, and/or manufactured for ordinary business, industrial, personal, and/or household applications only. The Products are not designed, intended, or authorized for use as components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, atomic energy control instruments, combustion control instruments, airplane or spaceship instruments, transportation instruments, traffic signal instruments, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or Product could cause personal injury, death, property or environmental damage ("Unintended Uses"). Customers shall take any and all actions to ensure using and selling the Products in accordance with the applicable laws and regulations. The Company is not liable, in whole or in part, and customers shall and hereby do release the Company as well as its suppliers and/or distributors from any claim, damage, or other liability arising from or related to all Unintended Uses of the Products. Customers shall indemnify and hold the Company as well as its suppliers and/or distributors harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of the Products.

Information in this document is provided solely in connection with the Products. The Company reserves the right to make changes, corrections, modifications or improvements to this document and Products and services described herein at any time, without notice.