

**GigaDevice Semiconductor Inc.**

**GD32F30x**  
**Arm<sup>®</sup> Cortex<sup>®</sup>-M4 32-bit MCU**

**User Manual**

Revision 2.8

( Jul. 2022 )

# Table of Contents

<b>Table of Contents .....</b>	<b>2</b>
<b>List of Figures.....</b>	<b>19</b>
<b>List of Table.....</b>	<b>26</b>
<b>1. System and memory architecture .....</b>	<b>30</b>
1.1. Arm® Cortex®-M4 processor .....	30
1.2. System architecture .....	31
1.3. Memory map .....	34
1.3.1. Bit-banding.....	38
1.3.2. On-chip SRAM memory.....	39
1.3.3. On-chip flash memory overview .....	39
1.4. Boot configuration .....	39
1.5. Device electronic signature.....	40
1.5.1. Memory density information .....	41
1.5.2. Unique device ID (96 bits) .....	41
1.6. System configuration registers .....	42
<b>2. Flash memory controller (FMC) .....</b>	<b>43</b>
2.1. Introduction.....	43
2.2. Main features .....	43
2.3. Function description.....	43
2.3.1. Flash memory architecture.....	43
2.3.2. Read operations.....	44
2.3.3. Unlock the FMC_CTLx registers .....	44
2.3.4. Page erase .....	45
2.3.5. Mass erase .....	46
2.3.6. Main flash programming .....	48
2.3.7. Option bytes Erase .....	49
2.3.8. Option bytes modify.....	50
2.3.9. Option bytes description .....	50
2.3.10. Page erase/program protection .....	51
2.3.11. Security protection .....	52
2.4. FMC registers.....	53
2.4.1. Wait state register (FMC_WS).....	53
2.4.2. Unlock key register 0(FMC_KEY0) .....	53
2.4.3. Option byte unlock key register (FMC_OBKEY) .....	54
2.4.4. Status register 0 (FMC_STAT0).....	54

2.4.5.	Control register 0(FMC_CTL0) .....	55
2.4.6.	Address register 0 (FMC_ADDR0).....	56
2.4.7.	Option byte status register (FMC_OBSTAT).....	57
2.4.8.	Erase/Program Protection register (FMC_WP).....	57
2.4.9.	Unlock key register 1(FMC_KEY1) .....	58
2.4.10.	Status register 1 (FMC_STAT1).....	58
2.4.11.	Control register 1(FMC_CTL1) .....	59
2.4.12.	Address register 1 (FMC_ADDR1).....	60
2.4.13.	Wait state enable register (FMC_WSEN) .....	60
2.4.14.	Product ID register (FMC_PID) .....	61
<b>3.</b>	<b>Power management unit (PMU) .....</b>	<b>62</b>
3.1.	Overview .....	62
3.2.	Characteristics .....	62
3.3.	Function overview.....	62
3.3.1.	Backup domain .....	63
3.3.2.	V <sub>DD</sub> / V <sub>DDA</sub> power domain .....	64
3.3.3.	1.2V power domain.....	66
3.3.4.	Power saving modes .....	66
3.4.	PMU registers .....	70
3.4.1.	Control register (PMU_CTL) .....	70
3.4.2.	Control and status register (PMU_CS).....	72
<b>4.</b>	<b>Backup registers (BKP) .....</b>	<b>74</b>
4.1.	Introduction.....	74
4.2.	Main features .....	74
4.3.	Function description.....	74
4.3.1.	RTC clock calibration .....	74
4.3.2.	Tamper detection .....	74
4.4.	BKP registers.....	76
4.4.1.	Backup data register x (BKP_DATAx) (x= 0..41) .....	76
4.4.2.	RTC signal output control register (BKP_OCTL) .....	76
4.4.3.	Tamper pin control register (BKP_TPCTL) .....	77
4.4.4.	Tamper control and status register (BKP_TPCS) .....	77
<b>5.</b>	<b>Reset and clock unit (RCU) .....</b>	<b>79</b>
	High- and extra-density eset and clock control unit (RCU) .....	79
5.1.	Reset control unit (RCTL) .....	79
5.1.1.	Overview .....	79
5.1.2.	Function overview.....	79
5.2.	Clock control unit (CCTL) .....	80

---

5.2.1.	Overview .....	80
5.2.2.	Characteristics.....	82
5.2.3.	Function overview.....	82
<b>5.3. Register definition.....</b>		<b>86</b>
5.3.1.	Control register (RCU_CTL).....	86
5.3.2.	Clock configuration register 0 (RCU_CFG0) .....	87
5.3.3.	Clock interrupt register (RCU_INT).....	91
5.3.4.	APB2 reset register (RCU_APB2RST).....	94
5.3.5.	APB1 reset register (RCU_APB1RST).....	96
5.3.6.	AHB enable register (RCU_AHBEN) .....	99
5.3.7.	APB2 enable register (RCU_APB2EN) .....	100
5.3.8.	APB1 enable register (RCU_APB1EN) .....	102
5.3.9.	Backup domain control register (RCU_BDCTL) .....	105
5.3.10.	Reset source/clock register (RCU_RSTSCK) .....	107
5.3.11.	Clock configuration register 1 (RCU_CFG1) .....	108
5.3.12.	Deep-sleep mode voltage register (RCU_DSV) .....	109
5.3.13.	Additional clock control register (RCU_ADDCTL) .....	110
5.3.14.	Additional clock interrupt register (RCU_ADDINT) .....	110
5.3.15.	APB1 additional reset register (RCU_ADDAPB1RST) .....	111
5.3.16.	APB1 additional enable register (RCU_ADDAPB1EN).....	112
<b>Connectivity line devices: reset and clock control unit (RCU) .....</b>		<b>113</b>
<b>5.4. Reset control unit (RCTL) .....</b>		<b>113</b>
5.4.1.	Overview .....	113
5.4.2.	Function overview.....	113
<b>5.5. Clock control unit (CCTL) .....</b>		<b>114</b>
5.5.1.	Overview .....	114
5.5.2.	Characteristics.....	116
5.5.3.	Function overview.....	116
<b>5.6. Register definition.....</b>		<b>121</b>
5.6.1.	Control register (RCU_CTL).....	121
5.6.2.	Clock configuration register 0 (RCU_CFG0) .....	123
5.6.3.	Clock interrupt register (RCU_INT).....	126
5.6.4.	APB2 reset register (RCU_APB2RST).....	130
5.6.5.	APB1 reset register (RCU_APB1RST).....	132
5.6.6.	AHB enable register (RCU_AHBEN) .....	135
5.6.7.	APB2 enable register (RCU_APB2EN) .....	136
5.6.8.	APB1 enable register (RCU_APB1EN) .....	139
5.6.9.	Backup domain control register (RCU_BDCTL) .....	142
5.6.10.	Reset source/clock register (RCU_RSTSCK) .....	143
5.6.11.	AHB reset register (RCU_AHBRST).....	145
5.6.12.	Clock configuration register 1 (RCU_CFG1) .....	145
5.6.13.	Deep-sleep mode voltage register (RCU_DSV) .....	148

---

5.6.14.	Additional clock control register (RCU_ADDCTL) .....	149
5.6.15.	Additional clock interrupt register (RCU_ADDINT).....	150
5.6.16.	APB1 additional reset register (RCU_ADDAPB1RST) .....	150
5.6.17.	APB1 additional enable register (RCU_ADDAPB1EN).....	151
<b>6.</b>	<b>Clock trim controller (CTC) .....</b>	<b>152</b>
6.1.	Overview .....	152
6.2.	Characteristics .....	152
6.3.	Function overview.....	152
6.3.1.	REF sync pulse generator.....	153
6.3.2.	CTC trim counter.....	153
6.3.3.	Frequency evaluation and automatically trim process .....	154
6.3.4.	Software program guide .....	155
6.4.	Register definition.....	157
6.4.1.	Control register 0 (CTC_CTL0).....	157
6.4.2.	Control register 1 (CTC_CTL1).....	158
6.4.3.	Status register (CTC_STAT).....	159
6.4.4.	Interrupt clear register (CTC_INTC).....	161
<b>7.</b>	<b>Interrupt/event controller (EXTI) .....</b>	<b>163</b>
7.1.	Overview .....	163
7.2.	Characteristics.....	163
7.3.	Interrupts function overview .....	163
7.4.	External interrupt and event (EXTI) block diagram.....	167
7.5.	External Interrupt and Event function overview .....	167
7.6.	EXTI Register .....	169
7.6.1.	Interrupt enable register (EXTI_INTEN).....	169
7.6.2.	Event enable register (EXTI EVEN).....	169
7.6.3.	Rising edge trigger enable register (EXTI_RTEN).....	170
7.6.4.	Falling edge trigger enable register (EXTI_FTEN) .....	170
7.6.5.	Software interrupt event register (EXTI_SWIEV).....	170
7.6.6.	Pending register (EXTI_PD).....	171
<b>8.</b>	<b>General-purpose and alternate-function I/Os (GPIO and AFIO) .....</b>	<b>172</b>
8.1.	Overview .....	172
8.2.	Characteristics.....	172
8.3.	Function overview.....	172
8.3.1.	GPIO pin configuration.....	173
8.3.2.	External interrupt/event lines .....	174
8.3.3.	Alternate functions (AF) .....	174
8.3.4.	Input configuration .....	174

8.3.5.	Output configuration .....	175
8.3.6.	Analog configuration.....	176
8.3.7.	Alternate function (AF) configuration .....	176
8.3.8.	GPIO locking function .....	177
8.3.9.	GPIO I/O compensation cell.....	177
<b>8.4.</b>	<b>Remapping function I/O and debug configuration .....</b>	<b>177</b>
8.4.1.	Introduction .....	177
8.4.2.	Main features.....	178
8.4.3.	JTAG/SWD alternate function remapping .....	178
8.4.4.	ADC AF remapping.....	179
8.4.5.	TIMER AF remapping.....	179
8.4.6.	USART AF remapping.....	181
8.4.7.	I2C0 AF remapping.....	181
8.4.8.	SPI/I2S AF remapping .....	182
8.4.9.	CAN AF remapping.....	182
8.4.10.	Ethernet AF remapping .....	183
8.4.11.	CTC AF remapping.....	183
8.4.12.	CLK pins AF remapping.....	183
<b>8.5.</b>	<b>Register definition.....</b>	<b>185</b>
8.5.1.	Port control register 0 (GPIO <sub>x</sub> _CTL0, x=A..G).....	185
8.5.2.	Port control register 1 (GPIO <sub>x</sub> _CTL1, x=A..G).....	187
8.5.3.	Port input status register (GPIO <sub>x</sub> _ISTAT, x=A..G) .....	188
8.5.4.	Port output control register (GPIO <sub>x</sub> _OCTL, x=A..G) .....	189
8.5.5.	Port bit operate register (GPIO <sub>x</sub> _BOP, x=A..G) .....	189
8.5.6.	Port bit clear register (GPIO <sub>x</sub> _BC, x=A..G) .....	190
8.5.7.	Port configuration lock register (GPIO <sub>x</sub> _LOCK, x=A..G) .....	190
8.5.8.	Port bit speed register (GPIO <sub>x</sub> _SPD, x=A..G) .....	191
8.5.9.	Event control register (AFIO_EC).....	192
8.5.10.	AFIO port configuration register 0 (AFIO_PCF0).....	192
8.5.11.	EXTI sources selection register 0 (AFIO_EXTISSL0).....	199
8.5.12.	EXTI sources selection register 1 (AFIO_EXTISSL1).....	201
8.5.13.	EXTI sources selection register 2 (AFIO_EXTISSL2).....	202
8.5.14.	EXTI sources selection register 3 (AFIO_EXTISSL3).....	203
8.5.15.	AFIO port configuration register 1 (AFIO_PCF1).....	204
8.5.16.	IO compensation control register (AFIO_CPSCTL).....	206
<b>9.</b>	<b>Cyclic redundancy checks management unit (CRC) .....</b>	<b>207</b>
9.1.	Overview .....	207
9.2.	Characteristics .....	207
9.3.	Function overview.....	208
<b>9.4.</b>	<b>Register definition.....</b>	<b>209</b>
9.4.1.	Data register (CRC_DATA) .....	209

9.4.2.	Free data register (CRC_FDATA) .....	209
9.4.3.	Control register (CRC_CTL).....	210
<b>10.</b>	<b>Direct memory access controller (DMA) .....</b>	<b>211</b>
<b>10.1.</b>	<b>Overview.....</b>	<b>211</b>
<b>10.2.</b>	<b>Characteristics .....</b>	<b>211</b>
<b>10.3.</b>	<b>Block diagram.....</b>	<b>212</b>
<b>10.4.</b>	<b>Function overview .....</b>	<b>212</b>
10.4.1.	DMA operation.....	212
10.4.2.	Peripheral handshake .....	214
10.4.3.	Arbitration.....	214
10.4.4.	Address generation .....	214
10.4.5.	Circular mode .....	215
10.4.6.	Memory to memory mode.....	215
10.4.7.	Channel configuration .....	215
10.4.8.	Interrupt.....	215
10.4.9.	DMA request mapping .....	216
<b>10.5.</b>	<b>Register definition .....</b>	<b>220</b>
10.5.1.	Interrupt flag register (DMA_INTF).....	220
10.5.2.	Interrupt flag clear register (DMA_INTC) .....	221
10.5.3.	Channel x control register (DMA_CHxCTL).....	221
10.5.4.	Channel x counter register (DMA_CHxCNT).....	223
10.5.5.	Channel x peripheral base address register (DMA_CHxPADDR).....	224
10.5.6.	Channel x memory base address register (DMA_CHxMADDR) .....	224
<b>11.</b>	<b>Debug (DBG) .....</b>	<b>226</b>
<b>11.1.</b>	<b>Introduction .....</b>	<b>226</b>
<b>11.2.</b>	<b>JTAG/SW function description .....</b>	<b>226</b>
11.2.1.	Switch JTAG or SW interface.....	226
11.2.2.	Pin assignment.....	226
11.2.3.	JTAG daisy chained structure .....	227
11.2.4.	Debug reset .....	227
11.2.5.	JEDEC-106 ID code .....	227
<b>11.3.</b>	<b>Debug hold function description .....</b>	<b>227</b>
11.3.1.	Debug support for power saving mode .....	227
11.3.2.	Debug support for TIMER, I2C, WWDGT, FWDGT and CAN .....	228
<b>11.4.</b>	<b>DBG registers.....</b>	<b>229</b>
11.4.1.	ID code register (DBG_ID) .....	229
11.4.2.	Control register 0 (DBG_CTL0) .....	229
<b>12.</b>	<b>Analog-to-digital converter (ADC).....</b>	<b>233</b>
<b>12.1.</b>	<b>Overview.....</b>	<b>233</b>

<b>12.2. Characteristics .....</b>	<b>233</b>
<b>12.3. Pins and internal signals .....</b>	<b>234</b>
<b>12.4. Functional overview .....</b>	<b>235</b>
12.4.1. Foreground calibration function .....	235
12.4.2. ADC clock .....	236
12.4.3. ADC enable .....	236
12.4.4. Routine sequence .....	236
12.4.5. Operation modes .....	236
12.4.6. Conversion result threshold monitor function .....	239
12.4.7. Data storage mode .....	239
12.4.8. Sample time configuration .....	240
12.4.9. External trigger configuration .....	240
12.4.10. DMA request .....	241
12.4.11. ADC internal channels .....	241
12.4.12. Programmable resolution (DRES) .....	242
12.4.13. On-chip hardware oversampling .....	242
<b>12.5. ADC sync mode.....</b>	<b>244</b>
12.5.1. Free mode.....	245
12.5.2. Routine parallel mode .....	245
12.5.3. Routine follow-up fast mode.....	246
12.5.4. Routine follow-up slow mode .....	246
<b>12.6. ADC interrupts.....</b>	<b>247</b>
<b>12.7. ADC registers.....</b>	<b>248</b>
12.7.1. Status register (ADC_STAT) .....	248
12.7.2. Control register 0 (ADC_CTL0) .....	249
12.7.3. Control register 1 (ADC_CTL1) .....	250
12.7.4. Sample time register 0 (ADC_SAMPT0).....	252
12.7.5. Sample time register 1 (ADC_SAMPT1).....	253
12.7.6. Watchdog high threshold register (ADC_WDHT) .....	254
12.7.7. Watchdog low threshold register (ADC_WDLT).....	254
12.7.8. Routine sequence register 0 (ADC_RSQ0) .....	255
12.7.9. Routine sequence register 1 (ADC_RSQ1) .....	255
12.7.10. Routine sequence register 2 (ADC_RSQ2).....	256
12.7.11. Routine data register (ADC_RDATA).....	257
12.7.12. Oversample control register (ADC_OVSAMPCTL) .....	257
<b>13. Digital-to-analog converter (DAC).....</b>	<b>260</b>
<b>13.1. Introduction .....</b>	<b>260</b>
<b>13.2. Characteristics .....</b>	<b>260</b>
<b>13.3. Function description .....</b>	<b>261</b>
13.3.1. DAC enable .....	261

13.3.2.	DAC output buffer .....	261
13.3.3.	DAC data configuration.....	262
13.3.4.	DAC trigger .....	262
13.3.5.	DAC workflow.....	262
13.3.6.	DAC noise wave .....	262
13.3.7.	DAC output calculate .....	263
13.3.8.	DMA function .....	264
13.3.9.	DAC concurrent conversion .....	264
<b>13.4.</b>	<b>DAC registers .....</b>	<b>265</b>
13.4.1.	Control register (DAC_CTL).....	265
13.4.2.	Software trigger register (DAC_SWT).....	267
13.4.3.	DAC0 12-bit right-aligned data holding register (DAC0_R12DH).....	268
13.4.4.	DAC0 12-bit left-aligned data holding register (DAC0_L12DH).....	268
13.4.5.	DAC0 8-bit right-aligned data holding register (DAC0_R8DH) .....	269
13.4.6.	DAC1 12-bit right-aligned data holding register (DAC1_R12DH).....	269
13.4.7.	DAC1 12-bit left-aligned data holding register (DAC1_L12DH).....	270
13.4.8.	DAC1 8-bit right-aligned data holding register (DAC1_R8DH) .....	270
13.4.9.	DAC concurrent mode 12-bit right-aligned data holding register (DACC_R12DH) .....	271
13.4.10.	DAC concurrent mode 12-bit left-aligned data holding register (DACC_L12DH) .....	271
13.4.11.	DAC concurrent mode 8-bit right-aligned data holding register (DACC_R8DH) .....	272
13.4.12.	DAC0 data output register (DAC0_DO) .....	272
13.4.13.	DAC1 data output register (DAC1_DO) .....	273
<b>14.</b>	<b>Watchdog timer (WDGT) .....</b>	<b>274</b>
<b>14.1.</b>	<b>Free watchdog timer (FWDGT) .....</b>	<b>274</b>
14.1.1.	Overview .....	274
14.1.2.	Characteristics.....	274
14.1.3.	Function overview.....	274
14.1.4.	Register definition .....	277
<b>14.2.</b>	<b>Window watchdog timer (WWDT) .....</b>	<b>280</b>
14.2.1.	Overview .....	280
14.2.2.	Characteristics.....	280
14.2.3.	Function overview.....	280
14.2.4.	Register definition .....	283
<b>15.</b>	<b>Real-time Clock(RTC).....</b>	<b>285</b>
<b>15.1.</b>	<b>Overview.....</b>	<b>285</b>
<b>15.2.</b>	<b>Characteristics .....</b>	<b>285</b>
<b>15.3.</b>	<b>Function overview .....</b>	<b>285</b>
15.3.1.	RTC reset.....	286
15.3.2.	RTC reading .....	286
15.3.3.	RTC configuration.....	286
15.3.4.	RTC flag assertion .....	287

<b>15.4. RTC Register .....</b>	<b>289</b>
15.4.1. RTC interrupt enable register(RTC_INTEN).....	289
15.4.2. RTC control register(RTC_CTL).....	289
15.4.3. RTC prescaler high register (RTC_PSCH) .....	290
15.4.4. RTC prescaler low register(RTC_PSCL).....	291
15.4.5. RTC divider high register (RTC_DIVH) .....	291
15.4.6. RTC divider low register (RTC_DIVL).....	291
15.4.7. RTC counter high register(RTC_CNTH) .....	292
15.4.8. RTC counter low register (RTC_CNTL).....	292
15.4.9. RTC alarm high register(RTC_ALRMH) .....	293
15.4.10. RTC alarm low register (RTC_ALRML).....	293
<b>16. Timer(TIMERx) .....</b>	<b>294</b>
<b>16.1. Advanced timer (TIMERx, x=0, 7).....</b>	<b>295</b>
16.1.1. Overview .....	295
16.1.2. Characteristics.....	295
16.1.3. Block diagram.....	296
16.1.4. Function overview.....	297
16.1.5. TIMERx registers(x=0, 7).....	323
<b>16.2. General level0 timer (TIMERx, x=1, 2, 3, 4) .....</b>	<b>349</b>
16.2.1. Overview .....	349
16.2.2. Characteristics.....	349
16.2.3. Block diagram.....	349
16.2.4. Function overview.....	351
16.2.5. TIMERx registers(x=1, 2, 3, 4).....	366
<b>16.3. General level1 timer (TIMERx, x=8, 11).....</b>	<b>387</b>
16.3.1. Overview .....	387
16.3.2. Characteristics.....	387
16.3.3. Block diagram.....	388
16.3.4. Function overview.....	389
16.3.5. TIMERx registers(x=8, 11).....	400
<b>16.4. General level2 timer (TIMERx, x=9, 10, 12, 13).....</b>	<b>412</b>
16.4.1. Overview .....	412
16.4.2. Characteristics.....	412
16.4.3. Block diagram.....	412
16.4.4. Function overview.....	414
16.4.5. TIMERx registers(x=9, 10, 12, 13).....	421
<b>16.5. Basic timer (TIMERx, x=5, 6).....</b>	<b>431</b>
16.5.1. Overview .....	431
16.5.2. Characteristics.....	431
16.5.3. Block diagram.....	431
16.5.4. Function overview.....	431

16.5.5. TIMERx registers(x=5, 6).....	435
<b>17. Universal synchronous/asynchronous receiver /transmitter (USART) .....</b>	<b>440</b>
17.1. Overview.....	440
17.2. Characteristics.....	440
17.3. Function overview .....	441
17.3.1. USART frame format.....	442
17.3.2. Baud rate generation.....	443
17.3.3. USART transmitter.....	443
17.3.4. USART receiver .....	444
17.3.5. Use DMA for data buffer access.....	446
17.3.6. Hardware flow control .....	447
17.3.7. Multi-processor communication .....	448
17.3.8. LIN mode.....	449
17.3.9. Synchronous mode.....	450
17.3.10. IrDA SIR ENDEC mode .....	451
17.3.11. Half-duplex communication mode.....	452
17.3.12. Smartcard (ISO7816-3) mode .....	452
17.3.13. USART interrupts.....	454
17.4. Register definition .....	456
17.4.1. Status register 0 (USART_STAT0).....	456
17.4.2. Data register (USART_DATA).....	458
17.4.3. Baud rate register (USART_BAUD).....	458
17.4.4. Control register 0 (USART_CTL0) .....	459
17.4.5. Control register 1 (USART_CTL1) .....	461
17.4.6. Control register 2 (USART_CTL2) .....	462
17.4.7. Guard time and prescaler register (USART_GP).....	464
17.4.8. Control register 3 (USART_CTL3) .....	465
17.4.9. Receiver timeout register (USART_RT) .....	466
17.4.10. Status register 1 (USART_STAT1).....	467
<b>18. Inter-integrated circuit interface (I2C) .....</b>	<b>469</b>
18.1. Overview.....	469
18.2. Characteristics .....	469
18.3. Function overview .....	469
18.3.1. SDA and SCL lines .....	470
18.3.2. Data validation.....	471
18.3.3. START and STOP signal .....	471
18.3.4. Clock synchronization .....	471
18.3.5. Arbitration.....	472
18.3.6. I2C communication flow.....	473
18.3.7. Programming model .....	473

18.3.8.	SCL line stretching.....	482
18.3.9.	Use DMA for data transfer.....	483
18.3.10.	Packet error checking .....	483
18.3.11.	SMBus support .....	484
18.3.12.	Status, errors and interrupts .....	485
<b>18.4.</b>	<b>Register definition .....</b>	<b>487</b>
18.4.1.	Control register 0 (I2C_CTL0).....	487
18.4.2.	Control register 1 (I2C_CTL1).....	489
18.4.3.	Slave address register 0 (I2C_SADDR0).....	490
18.4.4.	Slave address register 1 (I2C_SADDR1).....	490
18.4.5.	Transfer buffer register (I2C_DATA).....	491
18.4.6.	Transfer status register 0 (I2C_STAT0).....	491
18.4.7.	Transfer status register 1 (I2C_STAT1).....	494
18.4.8.	Clock configure register (I2C_CKCFG) .....	495
18.4.9.	Rise time register (I2C_RT).....	496
18.4.10.	Fast-mode-plus configure register (I2C_FMPCFG) .....	496
<b>19.</b>	<b>Serial peripheral interface/Inter-IC sound (SPI/I2S).....</b>	<b>498</b>
<b>19.1.</b>	<b>Overview.....</b>	<b>498</b>
<b>19.2.</b>	<b>Characteristics .....</b>	<b>498</b>
19.2.1.	SPI characteristics .....	498
19.2.2.	I2S characteristics .....	498
<b>19.3.</b>	<b>SPI function overview .....</b>	<b>499</b>
19.3.1.	SPI block diagram.....	499
19.3.2.	SPI signal description.....	499
19.3.3.	SPI clock timing and data format.....	500
19.3.4.	NSS function.....	501
19.3.5.	SPI operation modes.....	502
19.3.6.	DMA function .....	511
19.3.7.	CRC function .....	511
19.3.8.	SPI interrupts.....	511
<b>19.4.</b>	<b>I2S function overview .....</b>	<b>513</b>
19.4.1.	I2S block diagram .....	513
19.4.2.	I2S signal description .....	513
19.4.3.	I2S audio standards .....	514
19.4.4.	I2S clock.....	522
19.4.5.	Operation .....	523
19.4.6.	DMA function .....	527
19.4.7.	I2S interrupts .....	527
<b>19.5.</b>	<b>Register definition .....</b>	<b>529</b>
19.5.1.	Control register 0 (SPI_CTL0) .....	529
19.5.2.	Control register 1 (SPI_CTL1) .....	531

19.5.3.	Status register (SPI_STAT) .....	532
19.5.4.	Data register (SPI_DATA) .....	533
19.5.5.	CRC polynomial register (SPI_CRCPOLY) .....	534
19.5.6.	RX CRC register (SPI_RCRC) .....	534
19.5.7.	TX CRC register (SPI_TCRC) .....	535
19.5.8.	I2S control register (SPI_I2SCTL) .....	536
19.5.9.	I2S clock prescaler register (SPI_I2SPSC) .....	537
19.5.10.	Quad-SPI mode control register (SPI_QCTL) of SPI0 .....	538
<b>20.</b>	<b>Secure digital input/output interface (SDIO) .....</b>	<b>539</b>
20.1.	Introduction .....	539
20.2.	Main features .....	539
20.3.	SDIO bus topology .....	540
20.4.	SDIO functional description .....	542
20.4.1.	SDIO adapter .....	543
20.4.2.	AHB interface .....	547
20.5.	Card functional description .....	548
20.5.1.	Card registers .....	548
20.5.2.	Commands .....	549
20.5.3.	Responses .....	561
20.5.4.	Data packets format .....	564
20.5.5.	Two status fields of the card .....	566
20.6.	Programming sequence .....	573
20.6.1.	Card identification .....	573
20.6.2.	No data commands .....	574
20.6.3.	Single block or multiple block write .....	575
20.6.4.	Single block or multiple block read .....	576
20.6.5.	Stream write and stream read (MMC only) .....	577
20.6.6.	Erase .....	579
20.6.7.	Bus width selection .....	579
20.6.8.	Protection management .....	580
20.6.9.	Card Lock/Unlock operation .....	580
20.7.	Specific operations .....	583
20.7.1.	SD I/O specific operations .....	583
20.7.2.	CE-ATA specific operations .....	586
20.8.	SDIO registers .....	588
20.8.1.	Power control register (SDIO_PWRCTL) .....	588
20.8.2.	Clock control register (SDIO_CLKCTL) .....	588
20.8.3.	Command argument register (SDIO_CMDAGMT) .....	590
20.8.4.	Command control register (SDIO_CMDCTL) .....	590
20.8.5.	Command index response register (SDIO_RSPCMDIDX) .....	592

20.8.6.	Response register (SDIO_RESPx x=0..3) .....	592
20.8.7.	Data timeout register (SDIO_DATATO) .....	593
20.8.8.	Data length register (SDIO_DATALEN) .....	593
20.8.9.	Data control register (SDIO_DATACTL) .....	594
20.8.10.	Data counter register (SDIO_DATACNT) .....	595
20.8.11.	Status register (SDIO_STAT) .....	596
20.8.12.	Interrupt clear register (SDIO_INTC) .....	597
20.8.13.	Interrupt enable register (SDIO_INTEN) .....	598
20.8.14.	FIFO counter register (SDIO_FIFOCNT) .....	600
20.8.15.	FIFO data register (SDIO_FIFO) .....	601
<b>21.</b>	<b>External memory controller (EXMC) .....</b>	<b>602</b>
21.1.	Overview .....	602
21.2.	Characteristics .....	602
21.3.	Function overview .....	602
21.3.1.	Block diagram .....	602
21.3.2.	Basic regulation of EXMC access .....	603
21.3.3.	External device address mapping .....	604
21.3.4.	NOR/PSRAM controller .....	607
21.3.5.	NAND Flash or PC Card controller .....	626
21.4.	Registers definition .....	632
21.4.1.	NOR/PSRAM controller registers .....	632
21.4.2.	NAND Flash/PC Card controller registers .....	636
<b>22.</b>	<b>Controller area network (CAN) .....</b>	<b>643</b>
22.1.	Overview .....	643
22.2.	Characteristics .....	643
22.3.	Function overview .....	644
22.3.1.	Working mode .....	644
22.3.2.	Communication modes .....	645
22.3.3.	Data transmission .....	646
22.3.4.	Data reception .....	648
22.3.5.	Filtering function .....	649
22.3.6.	Time-triggered communication .....	653
22.3.7.	Communication parameters .....	653
22.3.8.	Error flags .....	655
22.3.9.	CAN interrupts .....	655
22.4.	CAN registers .....	657
22.4.1.	Control register (CAN_CTL) .....	657
22.4.2.	Status register (CAN_STAT) .....	658
22.4.3.	Transmit status register (CAN_TSTAT) .....	660
22.4.4.	Receive message FIFO0 register (CAN_RFIFO0) .....	663

22.4.5.	Receive message FIFO1 register (CAN_RFIFO1) .....	663
22.4.6.	Interrupt enable register (CAN_INTEN) .....	664
22.4.7.	Error register (CAN_ERR).....	666
22.4.8.	Bit timing register (CAN_BT).....	667
22.4.9.	Transmit mailbox identifier register (CAN_TMI $x$ ) ( $x=0..2$ ) .....	668
22.4.10.	Transmit mailbox property register (CAN_TMP $x$ ) ( $x=0..2$ ) .....	668
22.4.11.	Transmit mailbox data0 register (CAN_TMDATA0 $x$ ) ( $x=0..2$ ) .....	669
22.4.12.	Transmit mailbox data1 register (CAN_TMDATA1 $x$ ) ( $x=0..2$ ) .....	670
22.4.13.	Receive FIFO mailbox identifier register (CAN_RFIFOMI $x$ ) ( $x=0,1$ ).....	670
22.4.14.	Receive FIFO mailbox property register (CAN_RFIFOMP $x$ ) ( $x=0,1$ ).....	671
22.4.15.	Receive FIFO mailbox data0 register (CAN_RFIFOMDATA0 $x$ ) ( $x=0,1$ ).....	672
22.4.16.	Receive FIFO mailbox data1 register (CAN_RFIFOMDATA1 $x$ ) ( $x=0,1$ ).....	672
22.4.17.	Filter control register (CAN_FCTL) (Just for CAN0).....	673
22.4.18.	Filter mode configuration register (CAN_FMCFG) (Just for CAN0).....	673
22.4.19.	Filter scale configuration register (CAN_FSCFG) (Just for CAN0).....	674
22.4.20.	Filter associated FIFO register (CAN_FA FIFO) (Just for CAN0).....	674
22.4.21.	Filter working register (CAN_FW) (Just for CAN0).....	675
22.4.22.	Filter x data y register (CAN_FxDATAy) ( $x=0..27$ , $y=0,1$ ) (Just for CAN0) .....	675
<b>23.</b>	<b>Ethernet (ENET) .....</b>	<b>677</b>
23.1.	Overview.....	677
23.2.	Characteristics .....	677
23.2.1.	Block diagram.....	678
23.2.2.	MAC 802.3 Ethernet packet description.....	679
23.2.3.	Ethernet signal description .....	680
23.3.	Function overview .....	681
23.3.1.	Interface configuration .....	681
23.3.2.	MAC function overview .....	685
23.3.3.	MAC statistics counters: MSC .....	697
23.3.4.	Wake up management: WUM .....	697
23.3.5.	Precision time protocol: PTP .....	701
23.3.6.	DMA controller description .....	705
23.3.7.	Example for a typical configuration flow of Ethernet .....	730
23.3.8.	Ethernet interrupts .....	732
23.4.	Register definition .....	734
23.4.1.	MAC configuration register (ENET_MAC_CFG).....	734
23.4.2.	MAC frame filter register (ENET_MAC_FRMF) .....	736
23.4.3.	MAC hash list high register (ENET_MAC_HLH).....	738
23.4.4.	MAC hash list low register (ENET_MAC_HLL).....	738
23.4.5.	MAC PHY control register (ENET_MAC_PHY_CTL).....	739
23.4.6.	MAC PHY data register (ENET_MAC_PHY_DATA).....	740
23.4.7.	MAC flow control register (ENET_MAC_FCTL).....	740
23.4.8.	MAC VLAN tag register (ENET_MAC_VLT) .....	742

23.4.9.	MAC remote wakeup frame filter register (ENET_MAC_RWFF).....	742
23.4.10.	MAC wakeup management register (ENET_MAC_WUM).....	743
23.4.11.	MAC debug register (ENET_MAC_DBG) .....	744
23.4.12.	MAC interrupt flag register (ENET_MAC_INTF).....	746
23.4.13.	MAC interrupt mask register (ENET_MAC_INTMSK).....	747
23.4.14.	MAC address 0 high register (ENET_MAC_ADDR0H).....	747
23.4.15.	MAC address 0 low register (ENET_MAC_ADDR0L).....	748
23.4.16.	MAC address 1 high register (ENET_MAC_ADDR1H) .....	748
23.4.17.	MAC address 1 low register (ENET_MAC_ADDR1L).....	749
23.4.18.	MAC address 2 high register (ENET_MAC_ADDR2H) .....	749
23.4.19.	MAC address 2 low register (ENET_MAC_ADDR2L).....	750
23.4.20.	MAC address 3 high register (ENET_MAC_ADDR3H) .....	751
23.4.21.	MAC address 3 low register (ENET_MAC_ADDR3L).....	752
23.4.22.	MAC flow control threshold register (ENET_MAC_FCTH).....	752
23.4.23.	MSC control register (ENET_MSC_CTL) .....	753
23.4.24.	MSC receive interrupt flag register (ENET_MSC_RINTF) .....	754
23.4.25.	MSC transmit interrupt flag register (ENET_MSC_TINTF) .....	754
23.4.26.	MSC receive interrupt mask register (ENET_MSC_RINTMSK).....	755
23.4.27.	MSC transmit interrupt mask register (ENET_MSC_TINTMSK).....	756
23.4.28.	MSC transmitted good frames after a single collision counter register (ENET_MSC_SCCNT) .....	757
23.4.29.	MSC transmitted good frames after more than a single collision counter register (ENET_MSC_MSCCNT) .....	757
23.4.30.	MSC transmitted good frames counter register (ENET_MSC_TGFCNT).....	758
23.4.31.	MSC received frames with CRC error counter register (ENET_MSC_RFCECNT).....	758
23.4.32.	MSC received frames with alignment error counter register (ENET_MSC_RFAECNT) ..	758
23.4.33.	MSC received good unicast frames counter register (ENET_MSC_RGUFCNT) .....	759
23.4.34.	PTP time stamp control register (ENET_PTP_TSCTL).....	759
23.4.35.	PTP subsecond increment register (ENET_PTP_SSINC) .....	762
23.4.36.	PTP time stamp high register (ENET_PTP_TSH) .....	762
23.4.37.	PTP time stamp low register (ENET_PTP_TSL) .....	763
23.4.38.	PTP time stamp update high register (ENET_PTP_TSUH) .....	763
23.4.39.	PTP time stamp update low register (ENET_PTP_TSUL) .....	764
23.4.40.	PTP time stamp addend register (ENET_PTP_TSADDEND) .....	764
23.4.41.	PTP expected time high register (ENET_PTP_ETH).....	765
23.4.42.	PTP expected time low register (ENET_PTP_ETL) .....	765
23.4.43.	PTP time stamp flag register (ENET_PTP_TSFLAG).....	765
23.4.44.	PTP PPS control register (ENET_PTP_PPSCTL).....	766
23.4.45.	DMA bus control register (ENET_DMA_BCTL) .....	767
23.4.46.	DMA transmit poll enable register (ENET_DMA_TPEN) .....	769
23.4.47.	DMA receive poll enable register (ENET_DMA_RPEN) .....	769
23.4.48.	DMA receive descriptor table address register (ENET_DMA_RDTADDR).....	770
23.4.49.	DMA transmit descriptor table address register (ENET_DMA_TDTADDR).....	770
23.4.50.	DMA status register (ENET_DMA_STAT).....	771

23.4.51.	DMA control register (ENET_DMA_CTL).....	774
23.4.52.	DMA interrupt enable register (ENET_DMA_INTEN) .....	777
23.4.53.	DMA missed frame and buffer overflow counter register (ENET_DMA_MFBOCNT) .....	779
23.4.54.	DMA receive state watchdog counter register (ENET_DMA_RSWDC) .....	780
23.4.55.	DMA current transmit descriptor address register (ENET_DMA_CTDADDR) .....	780
23.4.56.	DMA current receive descriptor address register (ENET_DMA_CRDADDR) .....	781
23.4.57.	DMA current transmit buffer address register (ENET_DMA_CTBADDR).....	781
23.4.58.	DMA current receive buffer address register (ENET_DMA_CRBADDR).....	782
<b>24.</b>	<b>Universal Serial Bus full-speed device interface (USBD) .....</b>	<b>783</b>
24.1.	<b>Overview.....</b>	<b>783</b>
24.2.	<b>Main features.....</b>	<b>783</b>
24.3.	<b>Block diagram.....</b>	<b>783</b>
24.4.	<b>Signal description .....</b>	<b>784</b>
24.5.	<b>Clock configuration.....</b>	<b>784</b>
24.6.	<b>Function overview .....</b>	<b>785</b>
24.6.1.	USB endpoints .....	785
24.6.2.	Operation procedure .....	787
24.6.3.	USB events and interrupts .....	790
24.6.4.	Operation guide.....	791
24.7.	<b>Registers definition .....</b>	<b>794</b>
24.7.1.	USBD control register (USBD_CTL).....	794
24.7.2.	USBD interrupt flag register (USBD_INTF) .....	796
24.7.3.	USBD status register (USBD_STAT) .....	797
24.7.4.	USBD device address register (USBD_DADDR).....	797
24.7.5.	USBD buffer address register (USBD_BADDR) .....	798
24.7.6.	USBD endpoint x control and status register (USBD_EPxCS), x=[0..7] .....	798
24.7.7.	USBD endpoint x transmission buffer address register (USBD_EPxTBADDR), x can be in [0..7]	800
24.7.8.	USBD endpoint x transmission buffer byte count register (USBD_EPxTBCNT), x can be in [0..7]	800
24.7.9.	USBD endpoint x reception buffer address register (USBD_EPxRBADDR), x can be in [0..7]	801
24.7.10.	USBD endpoint x reception buffer byte count register (USBD_EPxRBCNT), x can be in [0..7]	801
24.7.11.	USBD LPM control and status register (USBD_LPMCS).....	802
<b>25.</b>	<b>Universal serial bus full-speed interface (USBFS) .....</b>	<b>803</b>
25.1.	<b>Overview.....</b>	<b>803</b>
25.2.	<b>Characteristics.....</b>	<b>803</b>
25.3.	<b>Block diagram.....</b>	<b>804</b>

<b>25.4. Signal description .....</b>	<b>804</b>
<b>25.5. Function overview .....</b>	<b>804</b>
25.5.1. USBFS clocks and working modes.....	804
25.5.2. USB host function.....	806
25.5.3. USB device function.....	808
25.5.4. OTG function overview .....	809
25.5.5. Data FIFO .....	810
25.5.6. Operation guide.....	812
<b>25.6. Interrupts.....</b>	<b>817</b>
<b>25.7. Register definition .....</b>	<b>819</b>
25.7.1. Global control and status registers.....	819
25.7.2. Host control and status registers .....	840
25.7.3. Device control and status registers .....	852
25.7.4. Power and clock control register (USBFS_PWRCLKCTL).....	876
<b>26. Revision history.....</b>	<b>877</b>

# List of Figures

Figure 1-1. The structure of the Cortex®-M4 processor .....	31
Figure 1-2. GD32F303 series system architecture .....	33
Figure 1-3. GD32F305 and GD32F307 series system architecture .....	34
Figure 2-1. Process of page erase operation .....	46
Figure 2-2. Process of mass erase operation .....	47
Figure 2-3. Process of word program operation .....	49
Figure 3-1. Power supply overview .....	63
Figure 3-2. Waveform of the POR / PDR.....	65
Figure 3-3. Waveform of the LVD threshold .....	65
Figure 5-1. The system reset circuit .....	80
Figure 5-2. Clock tree .....	81
Figure 5-3. HXTAL clock source .....	82
Figure 5-4. HXTAL clock source in bypass mode.....	83
Figure 5-4. The system reset circuit .....	114
Figure 5-5. Clock tree .....	115
Figure 5-6. HXTAL clock source .....	117
Figure 5-4. HXTAL clock source in bypass mode.....	117
Figure 6-1. CTC overview.....	153
Figure 6-2. CTC trim counter.....	154
Figure 7-1. Block diagram of EXTI .....	167
Figure 8-1. Basic structure of a standard I/O port bit .....	173
Figure 8-2. Input configuration .....	175
Figure 8-3. Output configuration .....	175
Figure 8-4. Analog configuration.....	176
Figure 8-5. Alternate function configuration .....	177
Figure 9-1. Block diagram of CRC calculation unit.....	208
Figure 10-1. Block diagram of DMA .....	212
Figure 10-2. Handshake mechanism .....	214
Figure 10-3. DMA interrupt logic .....	216
Figure 10-4. DMA0 request mapping .....	217
Figure 10-5. DMA1 request mapping .....	218
Figure 12-1. ADC module block diagram .....	235
Figure 12-2. Single operation mode.....	236
Figure 12-3. Continuous operation mode .....	237
Figure 12-4. Scan operation mode, continuous disable .....	238
Figure 12-5. Scan operation mode, continuous enable.....	238
Figure 12-6. Discontinuous operation mode .....	239
Figure 12-9. 12-bit Data storage mode .....	240
Figure 12-10. 6-bit Data storage mode .....	240
Figure 12-11. 20-bit to 16-bit result truncation.....	243

Figure 12-12. Numerical example with 5-bits shift and rounding .....	243
Figure 12-13. ADC sync block diagram.....	245
Figure 12-14. Routine parallel mode on 10 channels.....	246
Figure 12-16. Routine follow-up fast mode (the CTN bit of ADCs are set) .....	246
Figure 12-17. Routine follow-up slow mode.....	247
Figure 13-1. DAC block diagram.....	261
Figure 13-2. DAC LFSR algorithm .....	263
Figure 13-3. DAC triangle noise wave .....	263
Figure 14-1. Free watchdog block diagram .....	275
Figure 14-2. Window watchdog timer block diagram .....	280
Figure 14-3. Window watchdog timing diagram .....	281
Figure 15-1. Block diagram of RTC .....	286
Figure 16-1. Advanced timer block diagram .....	296
Figure 16-2. Timing chart of internal clock divided by 1.....	297
Figure 16-3. Timing chart of PSC value change from 0 to 2 .....	298
Figure 16-4. Timing chart of up counting mode, PSC=0/2 .....	299
Figure 16-5. Timing chart of up counting mode, change TIMERx_CAR on the go.....	300
Figure 16-6. Timing chart of down counting mode, PSC=0/2.....	301
Figure 16-7. Timing chart of down counting mode, change TIMERx_CAR on the go .....	302
Figure 16-8. Center-aligned counter timechart .....	303
Figure 16-9. Repetition timechart for center-aligned counter .....	304
Figure 16-10. Repetition timechart for up-counter .....	304
Figure 16-11. Repetition timechart for down-counter .....	305
Figure 16-12. Channel input capture principle.....	306
Figure 16-13. Output-compare under three modes .....	308
Figure 16-14. EAPWM timechart.....	309
Figure 16-15. CAPWM timechart.....	309
Figure 16-16. Complementary output with dead-time insertion. ....	312
Figure 16-17. Output behavior in response to a break(The break high active).....	313
Figure 16-18. Counter behavior with CI0FE0 polarity non-inverted in mode 2 .....	314
Figure 16-19. Counter behavior with CI0FE0 polarity inverted in mode 2 .....	314
Figure 16-20. Hall sensor is used to BLDC motor .....	315
Figure 16-21. Hall sensor timing between two timers .....	316
Figure 16-22. Restart mode .....	317
Figure 16-23. Pause mode .....	318
Figure 16-24. Event mode .....	318
Figure 16-25. Single pulse mode TIMERx_CHxCV = 4 TIMERx_CAR=99 .....	319
Figure 16-26. Timer0 master/slave mode timer example .....	320
Figure 16-27. Triggering TIMER0 with enable signal of TIMER2.....	321
Figure 16-28. Triggering TIMER0 and TIMER2 with TIMER2's CI0 input .....	322
Figure 16-29. General Level 0 timer block diagram .....	350
Figure 16-30. Timing chart of internal clock divided by 1 .....	351
Figure 16-31. Timing chart of PSC value change from 0 to 2 .....	352
Figure 16-32. Timing chart of up counting mode, PSC=0/2 .....	353

---

Figure 16-33. Timing chart of up counting mode, change TIMERx_CAR ongoing.....	354
Figure 16-34. Timing chart of down counting mode, PSC=0/2.....	355
Figure 16-35. Timing chart of down counting mode, change TIMERx_CAR ongoing .....	355
Figure 16-36. Timing chart of center-aligned counting mode .....	357
Figure 16-37. Channel input capture principle.....	358
Figure 16-38. Output-compare under three modes .....	360
Figure 16-39. EAPWM timechart .....	361
Figure 16-40. CAPWM timechart.....	361
Figure 16-41. Restart mode .....	363
Figure 16-42. Pause mode .....	363
Figure 16-43. Event mode .....	364
Figure 16-44. General level1 timer block diagram.....	388
Figure 16-45. Timing chart of internal clock divided by 1 .....	389
Figure 16-46. Timing chart of PSC value change from 0 to 2 .....	390
Figure 16-47. Timing chart of up counting mode, PSC=0/2.....	391
Figure 16-48. Timing chart of up counting mode, change TIMERx_CAR ongoing .....	391
Figure 16-49. Channel input capture principle.....	392
Figure 16-50. Output-compare under three modes .....	394
Figure 16-51. EAPWM timechart .....	395
Figure 16-52. CAPWM timechart.....	395
Figure 16-53. Restart mode .....	397
Figure 16-54. Pause mode .....	397
Figure 16-55. Event mode .....	398
Figure 16-56. Single pulse mode TIMERx_CHxCV = 4 TIMERx_CAR=99 .....	399
Figure 16-57. General level2 timer block diagram.....	413
Figure 16-58. Timing chart of internal clock divided by 1 .....	414
Figure 16-59. Timing chart of PSC value change from 0 to 2 .....	415
Figure 16-60. Timing chart of up counting mode, PSC=0/2 .....	416
Figure 16-61. Timing chart of up counting mode, change TIMERx_CAR on the go.....	416
Figure 16-62. Channel input capture principle.....	417
Figure 16-63. Output-compare under three modes .....	419
Figure 16-64. Basic timer block diagram .....	431
Figure 16-65. Timing chart of internal clock divided by 1.....	432
Figure 16-66. Timing chart of PSC value change from 0 to 2.....	432
Figure 16-67. Timing chart of up counting mode, PSC=0/2.....	433
Figure 16-68. Timing chart of up counting mode, change TIMERx_CAR ongoing .....	434
Figure 17-1. USART module block diagram .....	442
Figure 17-2. USART character frame (8 bits data and 1 stop bit) .....	442
Figure 17-3. USART transmit procedure .....	444
Figure 17-4. Receiving a frame bit by oversampling method.....	445
Figure 17-5. Configuration step when using DMA for USART transmission.....	446
Figure 17-6. Configuration steps when using DMA for USART reception .....	447
Figure 17-7. Hardware flow control between two USARTs .....	448
Figure 17-8. Hardware flow control.....	448

Figure 17-9. Break frame occurs during idle state .....	450
Figure 17-10. Break frame occurs during a frame .....	450
Figure 17-11. Example of USART in synchronous mode .....	450
Figure 17-12. 8-bit format USART synchronous waveform (CLEN=1) .....	451
Figure 17-13. IrDA SIR ENDEC module .....	451
Figure 17-14. IrDA data modulation .....	452
Figure 17-15. ISO7816-3 frame format .....	453
Figure 17-16. USART interrupt mapping diagram .....	455
Figure 18-1. I2C module block diagram .....	470
Figure 18-2. Data validation .....	471
Figure 18-3. START and STOP condition .....	471
Figure 18-4. Clock synchronization .....	472
Figure 18-5. SDA line arbitration .....	472
Figure 18-6. I2C communication flow with 7-bit address .....	473
Figure 18-7. I2C communication flow with 10-bit address (Master Transmit) .....	473
Figure 18-8. I2C communication flow with 10-bit address (Master Receive) .....	473
Figure 18-9. Programming model for slave transmitting mode (10-bit address mode) .....	475
Figure 18-10. Programming model for slave receiving (10-bit address mode) .....	476
Figure 18-11. Programming model for master transmitting mode (10-bit address mode) .....	478
Figure 18-12. Programming model for master receiving using Solution A (10-bit address mode) ..	479
Figure 18-13. Programming model for master receiving mode using solution B (10-bit address mode) .....	481
Figure 19-1. Block diagram of SPI .....	499
Figure 19-2. SPI timing diagram in normal mode .....	500
Figure 19-3. SPI timing diagram in Quad-SPI mode (CKPL=1, CKPH=1, LF=0) .....	501
Figure 19-4. A typical Full-duplex connection .....	504
Figure 19-5. A typical simplex connection (Master: Receive, Slave: Transmit) .....	504
Figure 19-6. A typical simplex connection (Master: Transmit only, Slave: Receive) .....	504
Figure 19-7. A typical bidirectional connection .....	504
Figure 19-8. Timing diagram of TI master mode with discontinuous transfer .....	506
Figure 19-9. Timing diagram of TI master mode with continuous transfer .....	507
Figure 19-10. Timing diagram of TI slave mode .....	507
Figure 19-11. Timing diagram of NSS pulse with continuous transmit .....	508
Figure 19-12. Timing diagram of quad write operation in Quad-SPI mode .....	509
Figure 19-13. Timing diagram of quad read operation in Quad-SPI mode .....	510
Figure 19-14. Block diagram of I2S .....	513
Figure 19-15. I2S Phillips standard timing diagram (DTLEN=00, CHLEN=0, CKPL=0) .....	514
Figure 19-16. I2S Phillips standard timing diagram (DTLEN=00, CHLEN=0, CKPL=1) .....	514
Figure 19-17. I2S Phillips standard timing diagram (DTLEN=10, CHLEN=1, CKPL=0) .....	515
Figure 19-18. I2S Phillips standard timing diagram (DTLEN=10, CHLEN=1, CKPL=1) .....	515
Figure 19-19. I2S Phillips standard timing diagram (DTLEN=01, CHLEN=1, CKPL=0) .....	515
Figure 19-20. I2S Phillips standard timing diagram (DTLEN=01, CHLEN=1, CKPL=1) .....	515
Figure 19-21. I2S Phillips standard timing diagram (DTLEN=00, CHLEN=1, CKPL=0) .....	516
Figure 19-22. I2S Phillips standard timing diagram (DTLEN=00, CHLEN=1, CKPL=1) .....	516

Figure 19-23. MSB justified standard timing diagram (DTLEN=00, CHLEN=0, CKPL=0) .....	516
Figure 19-24. MSB justified standard timing diagram (DTLEN=00, CHLEN=0, CKPL=1) .....	516
Figure 19-25. MSB justified standard timing diagram (DTLEN=10, CHLEN=1, CKPL=0) .....	516
Figure 19-26. MSB justified standard timing diagram (DTLEN=10, CHLEN=1, CKPL=1) .....	517
Figure 19-27. MSB justified standard timing diagram (DTLEN=01, CHLEN=1, CKPL=0) .....	517
Figure 19-28. MSB justified standard timing diagram (DTLEN=01, CHLEN=1, CKPL=1) .....	517
Figure 19-29. MSB justified standard timing diagram (DTLEN=00, CHLEN=1, CKPL=0) .....	517
Figure 19-30. MSB justified standard timing diagram (DTLEN=00, CHLEN=1, CKPL=1) .....	517
Figure 19-31. LSB justified standard timing diagram (DTLEN=01, CHLEN=1, CKPL=0) .....	518
Figure 19-32. LSB justified standard timing diagram (DTLEN=01, CHLEN=1, CKPL=1) .....	518
Figure 19-33. LSB justified standard timing diagram (DTLEN=00, CHLEN=1, CKPL=0) .....	518
Figure 19-34. LSB justified standard timing diagram (DTLEN=00, CHLEN=1, CKPL=1) .....	518
Figure 19-35. PCM standard short frame synchronization mode timing diagram (DTLEN=00, CHLEN=0, CKPL=0).....	519
Figure 19-36. PCM standard short frame synchronization mode timing diagram (DTLEN=00, CHLEN=0, CKPL=1).....	519
Figure 19-37. PCM standard short frame synchronization mode timing diagram (DTLEN=10, CHLEN=1, CKPL=0).....	519
Figure 19-38. PCM standard short frame synchronization mode timing diagram (DTLEN=10, CHLEN=1, CKPL=1).....	519
Figure 19-39. PCM standard short frame synchronization mode timing diagram (DTLEN=01, CHLEN=1, CKPL=0).....	519
Figure19-40. PCM standard short frame synchronization mode timing diagram (DTLEN=01, CHLEN=1, CKPL=1).....	520
Figure 19-41. PCM standard short frame synchronization mode timing diagram (DTLEN=00, CHLEN=1, CKPL=0).....	520
Figure 19-42. PCM standard short frame synchronization mode timing diagram (DTLEN=00, CHLEN=1, CKPL=1).....	520
Figure 19-43. PCM standard long frame synchronization mode timing diagram (DTLEN=00, CHLEN=0, CKPL=0).....	520
Figure19-44. PCM standard long frame synchronization mode timing diagram (DTLEN=00, CHLEN=0, CKPL=1).....	520
Figure 19-45. PCM standard long frame synchronization mode timing diagram (DTLEN=10, CHLEN=1, CKPL=0).....	520
Figure 19-46. PCM standard long frame synchronization mode timing diagram (DTLEN=10, CHLEN=1, CKPL=1).....	521
Figure 19-47. PCM standard long frame synchronization mode timing diagram (DTLEN=01, CHLEN=1, CKPL=0).....	521
Figure 19-48. PCM standard long frame synchronization mode timing diagram (DTLEN=01, CHLEN=1, CKPL=1).....	521
Figure 19-49. PCM standard long frame synchronization mode timing diagram (DTLEN=00, CHLEN=1, CKPL=0).....	521
Figure 19-50. PCM standard long frame synchronization mode timing diagram (DTLEN=00, CHLEN=1, CKPL=1).....	521

<b>Figure 19-51. Block diagram of I2S clock generator .....</b>	<b>522</b>
<b>Figure 19-52. I2S initialization sequence .....</b>	<b>524</b>
<b>Figure 19-53. I2S master reception disabling sequence .....</b>	<b>526</b>
<b>Figure 20-1. SDIO “no response” and “no data” operations .....</b>	<b>541</b>
<b>Figure 20-2. SDIO multiple blocks read operation .....</b>	<b>541</b>
<b>Figure 20-3. SDIO multiple blocks write operation .....</b>	<b>541</b>
<b>Figure 20-4. SDIO sequential read operation .....</b>	<b>542</b>
<b>Figure 20-5. SDIO sequential write operation .....</b>	<b>542</b>
<b>Figure 20-6. SDIO block diagram.....</b>	<b>543</b>
<b>Figure 20-7. Command Token Format .....</b>	<b>550</b>
<b>Figure 20-8. Response Token Format.....</b>	<b>561</b>
<b>Figure 20-9. 1-bit data bus width .....</b>	<b>565</b>
<b>Figure 20-10. 4-bit data bus width.....</b>	<b>565</b>
<b>Figure 20-11. 8-bit data bus width .....</b>	<b>565</b>
<b>Figure 20-12. Read wait control by stopping SDIO_CLK .....</b>	<b>583</b>
<b>Figure 20-13. Read wait operation using SDIO_DAT[2] .....</b>	<b>584</b>
<b>Figure 20-14. Function2 read cycle inserted during function1 multiple read cycle .....</b>	<b>585</b>
<b>Figure 20-15. Read Interrupt cycle timing.....</b>	<b>585</b>
<b>Figure 20-16. Write interrupt cycle timing.....</b>	<b>586</b>
<b>Figure 20-17. Multiple block 4-Bit read interrupt cycle timing .....</b>	<b>586</b>
<b>Figure 20-18. Multiple block 4-Bit write interrupt cycle timing .....</b>	<b>586</b>
<b>Figure 20-19. The operation for command completion disable signal .....</b>	<b>587</b>
<b>Figure 21-1. The EXMC block diagram .....</b>	<b>603</b>
<b>Figure 21-2. EXMC memory banks .....</b>	<b>604</b>
<b>Figure 21-3. Four regions of bank0 address mapping .....</b>	<b>605</b>
<b>Figure 21-4. NAND/PC Card address mapping .....</b>	<b>606</b>
<b>Figure 21-5. Diagram of bank1 common space .....</b>	<b>606</b>
<b>Figure 21-6. Mode 1 read access.....</b>	<b>611</b>
<b>Figure 21-7. Mode 1 write access .....</b>	<b>611</b>
<b>Figure 21-8. Mode A read access .....</b>	<b>612</b>
<b>Figure 21-9. Mode A write access .....</b>	<b>613</b>
<b>Figure 21-10. Mode 2/B read access.....</b>	<b>614</b>
<b>Figure 21-11. Mode 2 write access .....</b>	<b>615</b>
<b>Figure 21-12. Mode B write access .....</b>	<b>615</b>
<b>Figure 21-13. Mode C read access .....</b>	<b>616</b>
<b>Figure 21-14. Mode C write access .....</b>	<b>617</b>
<b>Figure 21-15. Mode D read access .....</b>	<b>618</b>
<b>Figure 21-16. Mode D write access .....</b>	<b>619</b>
<b>Figure 21-17. Multiplex mode read access .....</b>	<b>620</b>
<b>Figure 21-18. Multiplex mode write access .....</b>	<b>620</b>
<b>Figure 21-19. Read access timing diagram under async-wait signal assertion .....</b>	<b>622</b>
<b>Figure 21-20. Write access timing diagram under async-wait signal assertion .....</b>	<b>622</b>
<b>Figure 21-21. Read timing of synchronous multiplexed burst mode.....</b>	<b>624</b>
<b>Figure 21-22. Write timing of synchronous multiplexed burst mode .....</b>	<b>625</b>

---

<b>Figure 21-23. Access timing of common memory space of PC Card Controller .....</b>	<b>628</b>
<b>Figure 21-24. Access to none "NCE don't care" NAND Flash.....</b>	<b>629</b>
<b>Figure 22-1. CAN module block diagram .....</b>	<b>644</b>
<b>Figure 22-2. Transmission register .....</b>	<b>646</b>
<b>Figure 22-3. State of transmit mailbox .....</b>	<b>647</b>
<b>Figure 22-4. Reception register .....</b>	<b>648</b>
<b>Figure 22-5. 32-bit filter .....</b>	<b>650</b>
<b>Figure 22-6. 16-bit filter .....</b>	<b>650</b>
<b>Figure 22-7. 32-bit mask mode filter .....</b>	<b>650</b>
<b>Figure 22-8. 16-bit mask mode filter .....</b>	<b>650</b>
<b>Figure 22-9. 32-bit list mode filter.....</b>	<b>650</b>
<b>Figure 22-10. 16-bit list mode filter .....</b>	<b>650</b>
<b>Figure 22-11. The bit time .....</b>	<b>654</b>
<b>Figure 23-1. ENET module block diagram .....</b>	<b>678</b>
<b>Figure 23-2. MAC/Tagged MAC frame format.....</b>	<b>679</b>
<b>Figure 23-3. Station management interface signals .....</b>	<b>681</b>
<b>Figure 23-4. Media independent interface signals .....</b>	<b>683</b>
<b>Figure 23-5. Reduced media-independent interface signals .....</b>	<b>685</b>
<b>Figure 23-6. Wakeup frame filter register .....</b>	<b>698</b>
<b>Figure 23-7. System time update using the fine correction method.....</b>	<b>702</b>
<b>Figure 23-8. Descriptor ring and chain structure .....</b>	<b>706</b>
<b>Figure 23-9. Transmit descriptor in normal mode.....</b>	<b>711</b>
<b>Figure 23-10. Transmit descriptor in enhanced mode.....</b>	<b>717</b>
<b>Figure 23-11. Receive descriptor in normal mode .....</b>	<b>721</b>
<b>Figure 23-12. Receive descriptor in enhanced mode .....</b>	<b>728</b>
<b>Figure 23-13. MAC interrupt scheme .....</b>	<b>732</b>
<b>Figure 23-14. Ethernet interrupt scheme .....</b>	<b>733</b>
<b>Figure 23-15. Wakeup frame filter register .....</b>	<b>743</b>
<b>Figure 24-1. USBD block diagram .....</b>	<b>783</b>
<b>Figure 24-2. an example with buffer descriptor table usage (USBD_BADDR = 0).....</b>	<b>786</b>
<b>Figure 25-1. USBFS block diagram .....</b>	<b>804</b>
<b>Figure 25-2. Connection with host or device mode .....</b>	<b>805</b>
<b>Figure 25-3. Connection with OTG mode.....</b>	<b>806</b>
<b>Figure 25-4. State transition diagram of host port .....</b>	<b>806</b>
<b>Figure 25-5. HOST mode FIFO space in SRAM .....</b>	<b>811</b>
<b>Figure 25-6. Host mode FIFO access register map .....</b>	<b>811</b>
<b>Figure 25-7. Device mode FIFO space in SRAM .....</b>	<b>812</b>
<b>Figure 25-8. Device mode FIFO access register map .....</b>	<b>812</b>

# List of Table

<b>Table 1-1. The interconnection relationship of the AHB interconnect matrix .....</b>	<b>31</b>
<b>Table 1-2. Memory map of GD32F30x devices .....</b>	<b>35</b>
<b>Table 1-3. Boot modes .....</b>	<b>39</b>
<b>Table 1-4. Bootloader supported peripherals .....</b>	<b>40</b>
<b>Table 2-1. GD32F30x_CL and GD32F30x_HD, GD32F30x_XD base address and size for flash memory .....</b>	<b>43</b>
<b>Table 2-2. Option byte.....</b>	<b>50</b>
<b>Table 3-1. Power saving mode summary .....</b>	<b>68</b>
<b>Table 5-1. Clock output 0 source select .....</b>	<b>85</b>
<b>Table 5-2. 1.2V domain voltage selected in deep-sleep mode .....</b>	<b>85</b>
<b>Table 5-3. Clock output 0 source select .....</b>	<b>120</b>
<b>Table 5-4. 1.2V domain voltage selected in deep-sleep mode .....</b>	<b>120</b>
<b>Table 7-1. NVIC exception types in Cortex-M4.....</b>	<b>164</b>
<b>Table 7-2. Interrupt vector table .....</b>	<b>164</b>
<b>Table 7-3. EXTI source .....</b>	<b>168</b>
<b>Table 8-1. GPIO configuration table.....</b>	<b>173</b>
<b>Table 8-2. Debug interface signals.....</b>	<b>178</b>
<b>Table 8-3. Debug port mapping and Pin availability .....</b>	<b>178</b>
<b>Table 8-4. ADC0/ADC1 external trigger routine conversion AF remapping <sup>(1)</sup> .....</b>	<b>179</b>
<b>Table 8-5. TIMERx alternate function remapping .....</b>	<b>179</b>
<b>Table 8-6. TIMER4 alternate function remapping <sup>(1)</sup>.....</b>	<b>180</b>
<b>Table 8-7. USART alternate function remapping .....</b>	<b>181</b>
<b>Table 8-8. I2C0 alternate function remapping .....</b>	<b>182</b>
<b>Table 8-9. SPI/I2S alternate function remapping .....</b>	<b>182</b>
<b>Table 8-10. CAN alternate function remapping.....</b>	<b>182</b>
<b>Table 8-11. ENET alternate function remapping .....</b>	<b>183</b>
<b>Table 8-12. CTC alternate function remapping .....</b>	<b>183</b>
<b>Table 8-13. OSC32 pins configuration .....</b>	<b>184</b>
<b>Table 8-14. OSC pins configuration .....</b>	<b>184</b>
<b>Table 10-1. DMA transfer operation .....</b>	<b>213</b>
<b>Table 10-2. interrupt events.....</b>	<b>216</b>
<b>Table 10-3. DMA0 requests for each channel.....</b>	<b>217</b>
<b>Table 10-4. DMA1 requests for each channel.....</b>	<b>218</b>
<b>Table 12-1. ADC internal input signals .....</b>	<b>234</b>
<b>Table 12-2. ADC input pins definition.....</b>	<b>234</b>
<b>Table 12-3. External trigger source for ADC0 and ADC1.....</b>	<b>240</b>
<b>Table 12-4. External trigger source for ADC2.....</b>	<b>241</b>
<b>Table 12-5. t<sub>CONV</sub> timings depending on resolution .....</b>	<b>242</b>
<b>Table 12-6. Maximum output results vs N and M Grayed values indicates truncation .....</b>	<b>243</b>
<b>Table 12-7. ADC sync mode table .....</b>	<b>244</b>

<b>Table 13-1. DAC I/O description.....</b>	<b>261</b>
<b>Table 13-2. External triggers of DAC.....</b>	<b>262</b>
<b>Table 14-1. Min/max FWDGT timeout period at 40 kHz (IRC40K) .....</b>	<b>275</b>
<b>Table 14-2. Min/max timeout value at 60 MHz (<math>f_{PCLK1}</math>) .....</b>	<b>282</b>
<b>Table 16-1. Timers (TIMERx) are divided into five sorts .....</b>	<b>294</b>
<b>Table 16-2. Complementary outputs controlled by parameters .....</b>	<b>311</b>
<b>Table 16-3. Counting direction in different quadrature decoder mode .....</b>	<b>314</b>
<b>Table 16-4. Examples of slave mode .....</b>	<b>317</b>
<b>Table 16-5. Examples of slave mode .....</b>	<b>362</b>
<b>Table 16-6. Examples of slave mode.....</b>	<b>396</b>
<b>Table 17-1. Description of USART important pins.....</b>	<b>441</b>
<b>Table 17-2. Configuration of stop bits .....</b>	<b>442</b>
<b>Table 17-3. USART interrupt requests .....</b>	<b>455</b>
<b>Table 18-1. Definition of I2C-bus terminology (refer to the I2C specification of Philips semiconductors) .....</b>	<b>470</b>
<b>Table18-2. Event status flags .....</b>	<b>485</b>
<b>Table18-3. I2C error flags .....</b>	<b>485</b>
<b>Table 19-1. SPI signal description.....</b>	<b>499</b>
<b>Table 19-2. Quad-SPI signal description.....</b>	<b>500</b>
<b>Table 19-3. NSS function in slave mode .....</b>	<b>501</b>
<b>Table 19-4. NSS function in master mode .....</b>	<b>502</b>
<b>Table 19-5. SPI operation modes.....</b>	<b>502</b>
<b>Table 19-6. SPI interrupt requests.....</b>	<b>512</b>
<b>Table 19-7. I2S bitrate calculation formulas .....</b>	<b>522</b>
<b>Table 19-8. Audio sampling frequency calculation formulas .....</b>	<b>523</b>
<b>Table 19-9. Direction of I2S interface signals for each operation mode .....</b>	<b>523</b>
<b>Table 19-10. I2S interrupt.....</b>	<b>528</b>
<b>Table 20-1. SDIO I/O definitions .....</b>	<b>544</b>
<b>Table 20-2. Command format .....</b>	<b>550</b>
<b>Table 20-3. Card command classes (CCCs) .....</b>	<b>551</b>
<b>Table 20-4. Basic commands (class 0) .....</b>	<b>553</b>
<b>Table 20-5. Block-Oriented read commands (class 2) .....</b>	<b>554</b>
<b>Table 20-6. Stream read commands (class 1) and stream write commands (class 3) .....</b>	<b>555</b>
<b>Table 20-7. Block-Oriented write commands (class 4) .....</b>	<b>555</b>
<b>Table 20-8. Erase commands (class 5) .....</b>	<b>556</b>
<b>Table 20-9. Block oriented write protection commands (class 6) .....</b>	<b>557</b>
<b>Table 20-10. Lock card (class 7) .....</b>	<b>558</b>
<b>Table 20-11. Application-specific commands (class 8).....</b>	<b>558</b>
<b>Table 20-12. I/O mode commands (class 9) .....</b>	<b>559</b>
<b>Table 20-13. Switch function commands (class 10) .....</b>	<b>560</b>
<b>Table 20-14. Response R1 .....</b>	<b>562</b>
<b>Table 20-15. Response R2 .....</b>	<b>562</b>
<b>Table 20-16. Response R3 .....</b>	<b>563</b>
<b>Table 20-17. Response R4 for MMC .....</b>	<b>563</b>

<b>Table 20-18. Response R4 for SD I/O .....</b>	<b>563</b>
<b>Table 20-19. Response R5 for MMC .....</b>	<b>563</b>
<b>Table 20-20. Response R5 for SD I/O .....</b>	<b>564</b>
<b>Table 20-21. Response R6 .....</b>	<b>564</b>
<b>Table 20-22. Response R7 .....</b>	<b>564</b>
<b>Table 20-23. Card status.....</b>	<b>566</b>
<b>Table 20-24. SD status.....</b>	<b>569</b>
<b>Table 20-25. Performance move field .....</b>	<b>571</b>
<b>Table 20-26. AU_SIZE field.....</b>	<b>571</b>
<b>Table 20-27. Maximum AU size .....</b>	<b>571</b>
<b>Table 20-28. Erase size field .....</b>	<b>572</b>
<b>Table 20-29. Erase timeout field.....</b>	<b>572</b>
<b>Table 20-30. Erase offset field .....</b>	<b>572</b>
<b>Table 20-31. Lock card data structure .....</b>	<b>581</b>
<b>Table 20-32. SDIO_RESPx register at different response type .....</b>	<b>592</b>
<b>Table 21-1. NOR Flash interface signals description .....</b>	<b>607</b>
<b>Table 21-2. PSRAM non-muxed signal description.....</b>	<b>608</b>
<b>Table 21-3. EXMC bank 0 supports all transactions .....</b>	<b>608</b>
<b>Table 21-4. NOR / PSRAM controller timing parameters.....</b>	<b>609</b>
<b>Table 21-5. EXMC_timing models.....</b>	<b>610</b>
<b>Table 21-6. Mode 1 related registers configuration .....</b>	<b>611</b>
<b>Table 21-7. Mode A related registers configuration.....</b>	<b>613</b>
<b>Table 21-8. Mode 2/B related registers configuration.....</b>	<b>615</b>
<b>Table 21-9. Mode C related registers configuration.....</b>	<b>617</b>
<b>Table 21-10. Mode D related registers configuration.....</b>	<b>619</b>
<b>Table 21-11. Multiplex mode related registers configuration .....</b>	<b>621</b>
<b>Table 21-12. Timing configurations of synchronous multiplexed read mode.....</b>	<b>624</b>
<b>Table 21-13. Timing configurations of synchronous multiplexed write mode.....</b>	<b>625</b>
<b>Table 21-14. 8-bit or 16-bit NAND interface signal .....</b>	<b>626</b>
<b>Table 21-15. 16-bit PC Card interface signal .....</b>	<b>627</b>
<b>Table 21-16. Bank1/2/3 of EXMC support the memory and access mode.....</b>	<b>627</b>
<b>Table 21-17. NAND Flash or PC Card programmable parameters.....</b>	<b>628</b>
<b>Table 22-1. 32-bit filter number .....</b>	<b>651</b>
<b>Table 22-2. Filtering index.....</b>	<b>652</b>
<b>Table 23-1. Ethernet pin configuration .....</b>	<b>680</b>
<b>Table 23-2. Clock range.....</b>	<b>683</b>
<b>Table 23-3. Rx interface signal encoding .....</b>	<b>684</b>
<b>Table 23-4. Destination address filtering table .....</b>	<b>693</b>
<b>Table 23-5. Source address filtering table .....</b>	<b>693</b>
<b>Table 23-6. Error status decoding in RDES0, only used for normal descriptor (DFM=0).....</b>	<b>725</b>
<b>Table 23-7. Supported time stamp snapshot with PTP register configuration .....</b>	<b>761</b>
<b>Table 24-1. USBD signal description .....</b>	<b>784</b>
<b>Table 24-2. Double-buffering buffer flag definition .....</b>	<b>787</b>
<b>Table 24-3. Double buffer usage .....</b>	<b>787</b>

<b>Table 24-4. Reception status encoding .....</b>	<b>799</b>
<b>Table 24-5. Endpoint type encoding .....</b>	<b>800</b>
<b>Table 24-6. Endpoint kind meaning .....</b>	<b>800</b>
<b>Table 24-7. Transmission status encoding.....</b>	<b>800</b>
<b>Table 25-1. USBFS signal description .....</b>	<b>804</b>
<b>Table 25-2. USBFS global interrupt.....</b>	<b>817</b>
<b>Table 26-1. Revision history.....</b>	<b>877</b>

## 1. System and memory architecture

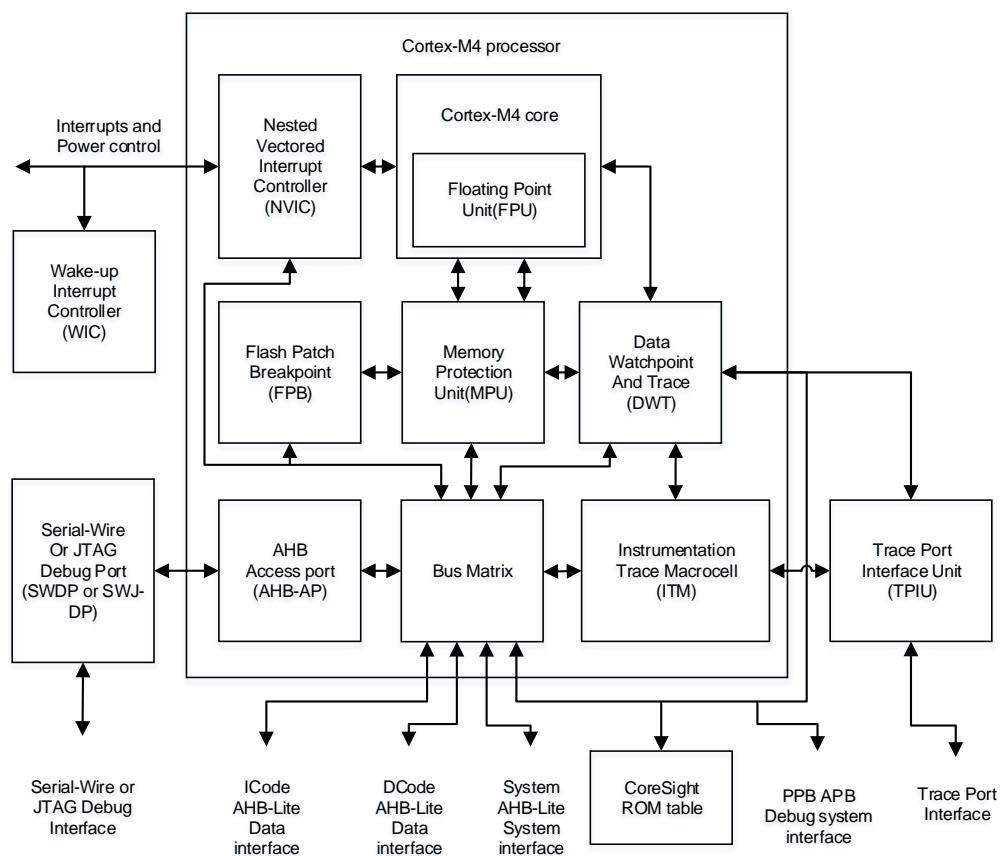
The devices of GD32F30x series are 32-bit general-purpose microcontrollers based on the Arm® Cortex®-M4 processor. The Arm® Cortex®-M4 processor includes three AHB buses known as I-Code, D-Code and System buses. All memory accesses of the Arm® Cortex®-M4 processor are executed on the three buses according to the different purposes and the target memory spaces. The memory organization uses a Harvard architecture, pre-defined memory map and up to 4 GB of memory space, making the system flexible and extendable.

### 1.1. Arm® Cortex®-M4 processor

The Cortex®-M4 processor is a 32-bit processor that possesses low interrupt latency and low-cost debug. The characteristics of integrated and advanced make the Cortex®-M4 processor suitable for market products that require microcontrollers with high performance and low power consumption. The Cortex®-M4 processor is based on the ARMv7 architecture and supports a powerful and scalable instruction set including general data processing I/O control tasks, advanced data processing bit field manipulations and DSP. Some system peripherals listed below are also provided by Cortex®-M4:

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

[Figure 1-1. The structure of the Cortex®-M4 processor](#) shows the Cortex®-M4 processor block diagram. For more information, please refer to the Arm® Cortex®-M4 Technical Reference Manual.

**Figure 1-1. The structure of the Cortex®-M4 processor**


## 1.2. System architecture

A 32-bit multilayer bus is implemented in the GD32F30x devices, which enables parallel access paths between multiple masters and slaves in the system. 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 the following table, “1” indicates the corresponding master is able to access the corresponding slave through the AHB interconnect matrix, while the blank means 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	ENET
FMC-I	1					
FMC-D		1		1	1	
SRAM	1	1	1	1	1	1
EXMC	1	1	1	1	1	1

	<b>IBUS</b>	<b>DBUS</b>	<b>SBUS</b>	<b>DMA0</b>	<b>DMA1</b>	<b>ENET</b>
<b>AHB</b>			1	1	1	
<b>APB1</b>			1	1	1	
<b>APB2</b>			1	1	1	

As is shown above, there are several masters connected with the AHB interconnect matrix, including IBUS, DBUS, SBUS, DMA0, DMA1 and ENET. IBUS is the instruction bus of the Cortex®-M4 core, which is used for instruction/vector fetches from the Code region (0x0000 ~ 0x1FFF FFFF). DBUS is the data bus of the Cortex®-M4 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 Cortex®-M4 core, which is used for instruction/vector fetches, data loading/storing and debugging access of the system regions. The System regions include the internal SRAM region and the Peripheral region. DMA0 and DMA1 are the buses of DMA0 and DMA1 respectively. ENET is the Ethernet.

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, while 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 of the AHB slaves, while APB1 and APB2 are the two APB buses connected with all of the APB slaves. The two APB buses connect with all the APB peripherals. APB1 is limited to 60 MHz, APB2 operates at full speed (up to 120MHz depending on the device).

These are interconnected using a multilayer AHB bus architecture as shown in figure below:

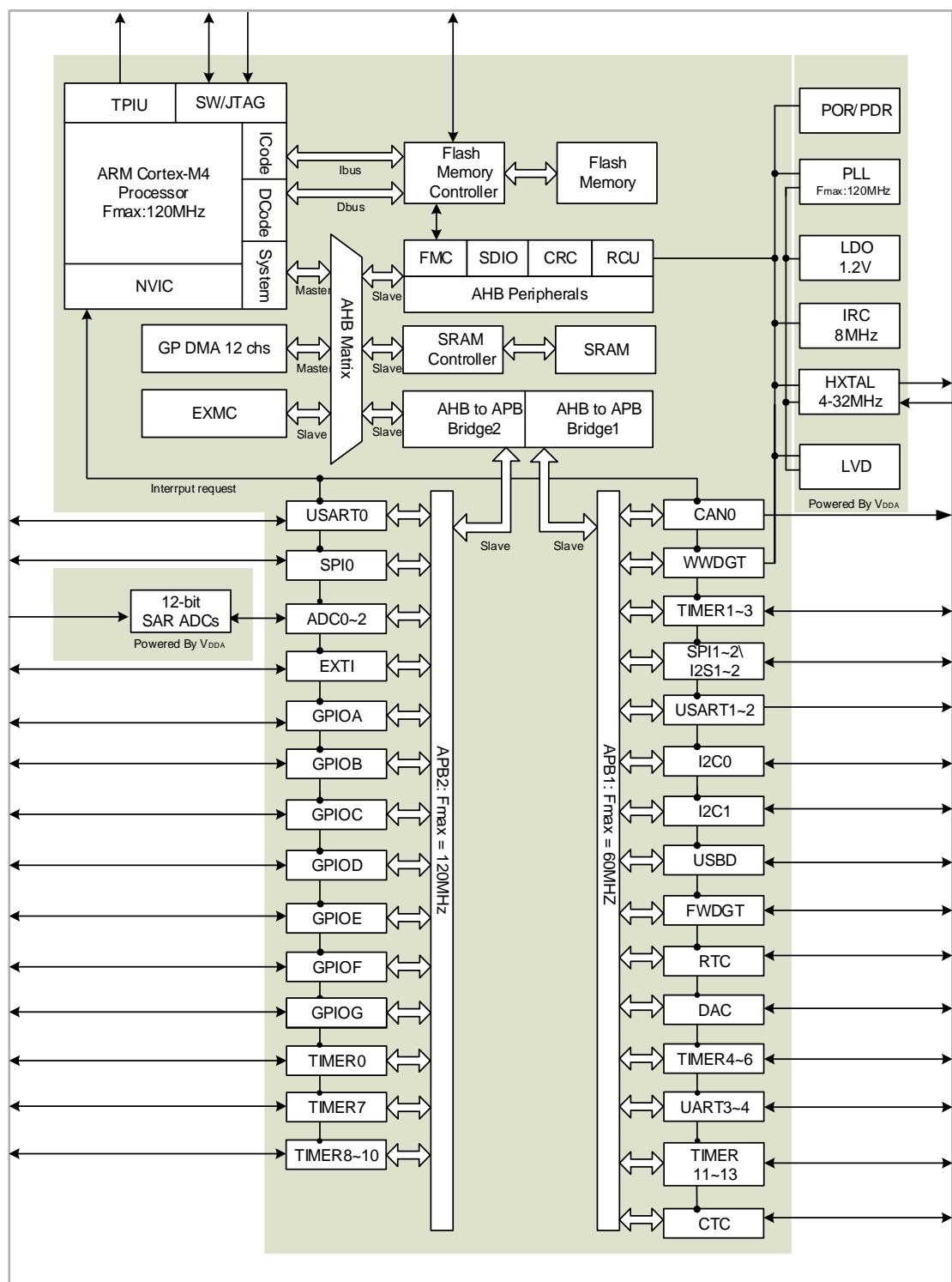
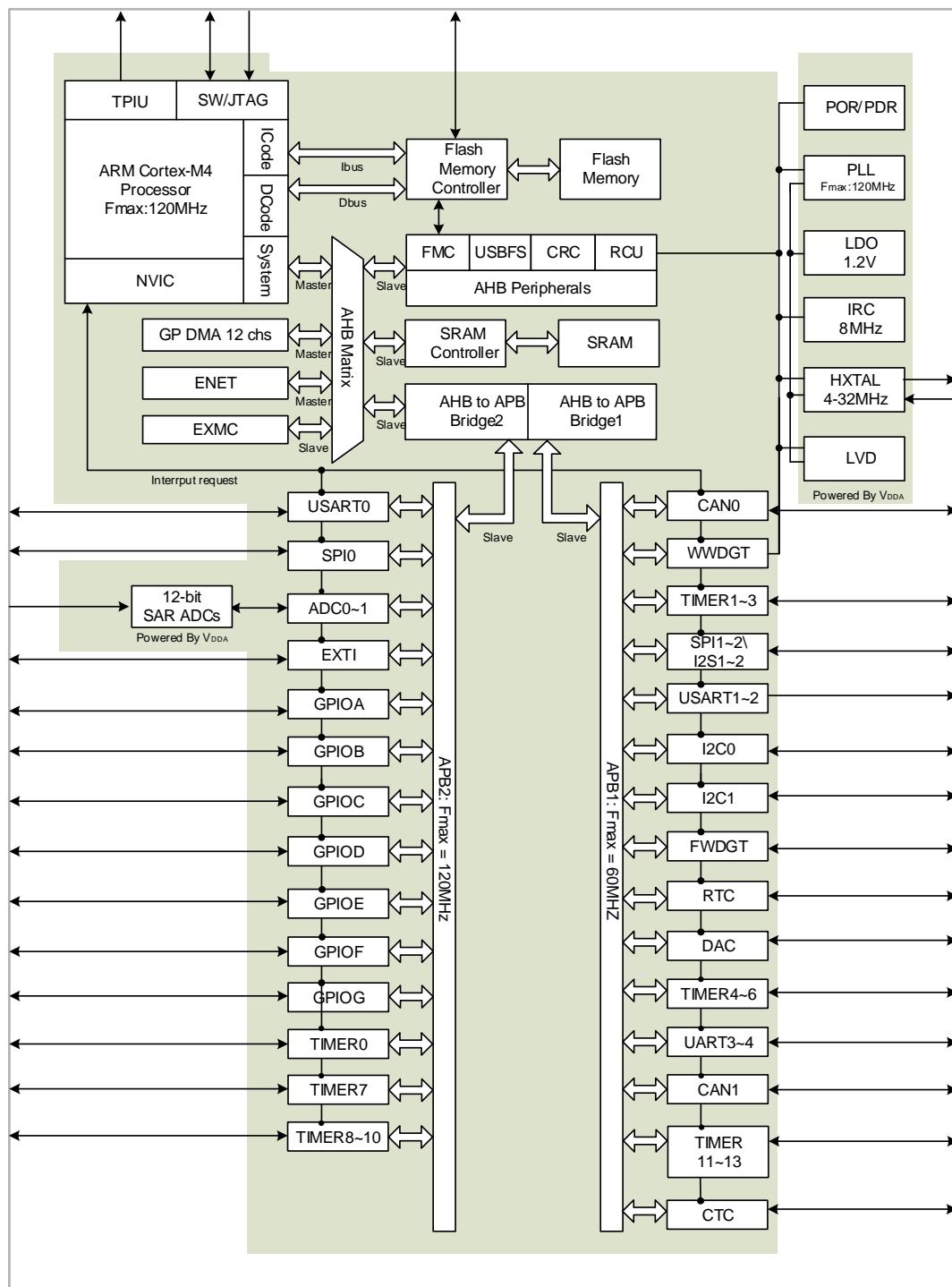
**Figure 1-2. GD32F303 series system architecture**


Figure 1-3. GD32F305 and GD32F307 series system architecture



### 1.3. Memory map

The Arm® Cortex®-M4 processor is structured in Harvard architecture which can use separate buses to fetch instructions and load/store data. The instruction code and data are both located in the same memory address space but in different address ranges. Program memory, data memory, registers and I/O ports are organized within the same linear 4-Gbyte address space

which is the maximum address range of the Cortex®-M4 since the bus address width is 32-bit. Additionally, a pre-defined memory map is provided by the Cortex®-M4 processor to reduce the software complexity of repeated implementation of different device vendors. In the map, some regions are used by the Arm® Cortex®-M4 system peripherals which can not be modified. However, the other regions are available to the vendors. [\*\*Table 1-2. Memory map of GD32F30x devices\*\*](#) shows the memory map of the GD32F30x 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 GD32F30x devices**

Pre-defined Regions	Bus	Address	Peripherals	
External device	AHB3	0xA000 0000 - 0xA000 0FFF	EXMC - SWREG	
External RAM		0x9000 0000 - 0x9FFF FFFF	EXMC - PC CARD	
		0x7000 0000 - 0x8FFF FFFF	EXMC - NAND	
		0x6000 0000 - 0x6FFF FFFF	EXMC - NOR/PSRAM/SRAM	
Peripheral	AHB1	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	ENET	
		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	
		0x4002 1400 - 0x4002 17FF	Reserved	
		0x4002 1000 - 0x4002 13FF	RCU	

<b>Pre-defined Regions</b>	<b>Bus</b>	<b>Address</b>	<b>Peripherals</b>
APB2		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	SDIO
		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	TIMER10
		0x4001 5000 - 0x4001 53FF	TIMER9
		0x4001 4C00 - 0x4001 4FFF	TIMER8
		0x4001 4800 - 0x4001 4BFF	Reserved
		0x4001 4400 - 0x4001 47FF	Reserved
		0x4001 4000 - 0x4001 43FF	Reserved
		0x4001 3C00 - 0x4001 3FFF	ADC2
		0x4001 3800 - 0x4001 3BFF	USART0
	APB1	0x4001 3400 - 0x4001 37FF	TIMER7
		0x4001 3000 - 0x4001 33FF	SPI0
		0x4001 2C00 - 0x4001 2FFF	TIMER0
		0x4001 2800 - 0x4001 2BFF	ADC1
		0x4001 2400 - 0x4001 27FF	ADC0
		0x4001 2000 - 0x4001 23FF	GPIOG
		0x4001 1C00 - 0x4001 1FFF	GPIOF
		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 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 USBD/CAN SRAM 512 bytes
		0x4000 5C00 - 0x4000 5FFF	USBD
		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	TIMER13
		0x4000 1C00 - 0x4000 1FFF	TIMER12
		0x4000 1800 - 0x4000 1BFF	TIMER11
		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
		0x2002 0000 - 0x2002 FFFF	Reserved
		0x2001 8000 - 0x2001 FFFF	Reserved
		0x2000 0000 - 0x2001 7FFF	SRAM

Pre-defined Regions	Bus	Address	Peripherals
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
		0x0830 0000 - 0x083B FFFF	Reserved
		0x0800 0000 - 0x082F 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. Bit-banding

In order to reduce the time of read-modify-write operations, the Cortex®-M4 processor provides a bit-banding function to perform a single atomic bit operation. The memory map includes two bit-band regions. These occupy the SRAM and Peripherals respectively. These bit-band regions map each word in an alias region of memory to a bit in a bit-band region of memory.

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

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

where:

- `bit_word_addr` is the address of the word in the alias memory region that maps to the targeted bit.
- `bit_band_base` is the starting address of the alias region.
- `byte_offset` is the number of the byte in the bit-band region that contains the targeted bit.
- `bit_number` is the bit position (0-7) of the targeted bit.

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

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

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

### 1.3.2. On-chip SRAM memory

The GD32F30x series of devices contain up to 96 KB of on-chip SRAM which starts at the address 0x2000 0000. It supports byte, half-word (16 bits), and word (32 bits) accesses.

### 1.3.3. On-chip flash memory overview

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

- Up to 3072KB of main flash memory.
- Up to 18KB of information blocks for the boot loader.
- Option bytes to configure the device.

GD32F303xx microcontrollers where the flash memory density ranges between 128 and 512 Kbytes are called High-density devices (GD32F30X\_HD).

GD32F303xx microcontrollers where the flash memory density is over 512 Kbytes are called Extra-density devices (GD32F30X\_XD).

GD32F305xx and GD32F307xx microcontrollers are called connectivity line devices (GD32F30X\_CL).

Refer to [\*\*\*Flash Memory Controller \(FMC\)\*\*\*](#) Chapter for more details.

## 1.4. Boot configuration

The GD32F30x 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. It is up to the user to set the BOOT0 and BOOT1 pins after a power-on reset or a system reset to select the required boot source. 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

After power-on sequence or a system reset, the Arm® Cortex®-M4 processor fetches the top-

of-stack value from address 0x0000 0000 and the base address of boot code from 0x0000 0004 in sequence. Then, it starts executing code from the base address of boot code.

Due to the selected boot source, either the main flash memory (original memory space beginning at 0x0800 0000) or the system memory (HD series original memory space beginning at 0x1FFF F000, please refer to [\*\*Table 2-1. GD32F30x CL and GD32F30x HD, GD32F30x XD base address and size for flash memory\*\*](#) for other series addresses) is aliased in the boot memory space which begins at the address 0x0000 0000. When the on-chip SRAM, whose memory space is beginning at 0x2000 0000, is selected as the boot source, in the application initialization code, you have to relocate the vector table in SRAM using the NVIC exception table and offset register.

The embedded boot loader is located in the System memory, which is used to reprogram the Flash memory.

GD32F3 MCU embedded bootloader supports multi interfaces to update the Flash memory. There will be one or two USART ports, and standard USB port can be used on GD32F30x line products. The details are shown in the following [\*\*Table 1-4. Bootloader supported peripherals\*\*](#).

**Table 1-4. Bootloader supported peripherals**

Products line	Products	Supported serial peripherals
HD	GD32F303xx	USART0(PA9 PA10)
XD	GD32F303xx	USART0(PA9 PA10) USART1(PA2 PA3)
CL	GD32F305xx	USART0(PA9 PA10) USART1(PD5 PD6) USB(PA9 PA11 PA12)
	GD32F307xx	USART0(PA9 PA10) USART1(PD5 PD6) USB(PA9 PA11 PA12)

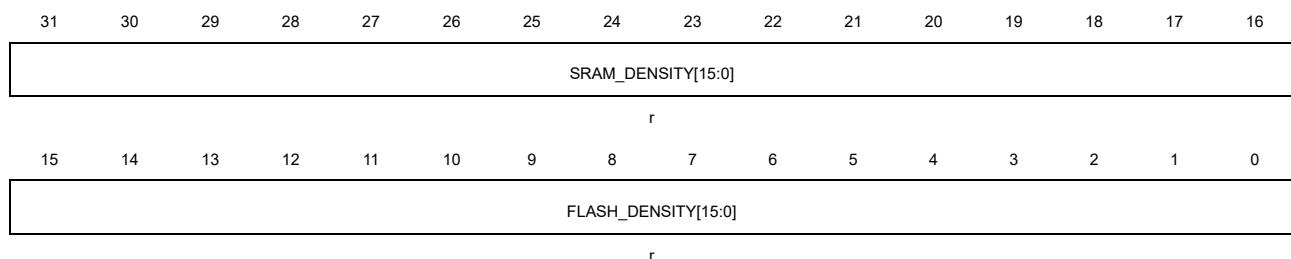
## 1.5. Device electronic signature

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.

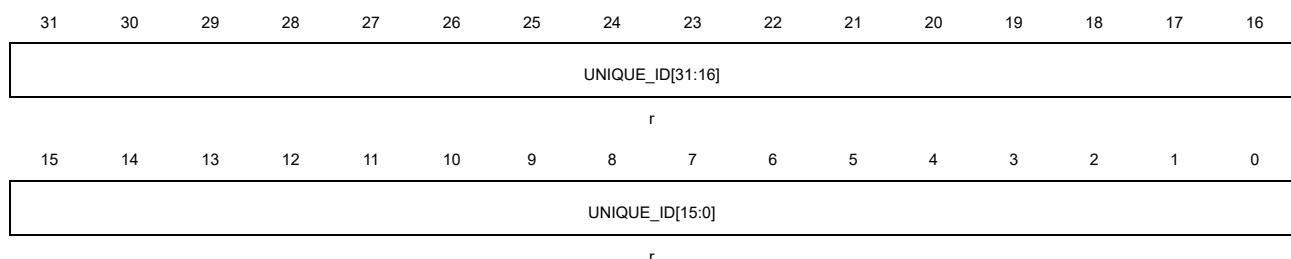


Bits	Fields	Descriptions
31:16	SRAM_DENSITY [15:0]	SRAM density The value indicates the on-chip SRAM density of the device in Kbytes. Example: 0x0008 indicates 8 Kbytes.
15:0	FLASH_DENSITY [15:0]	Flash memory density The value indicates the Flash memory density of the device in Kbytes. Example: 0x0020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

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

Base address: 0xFFFF F7E8

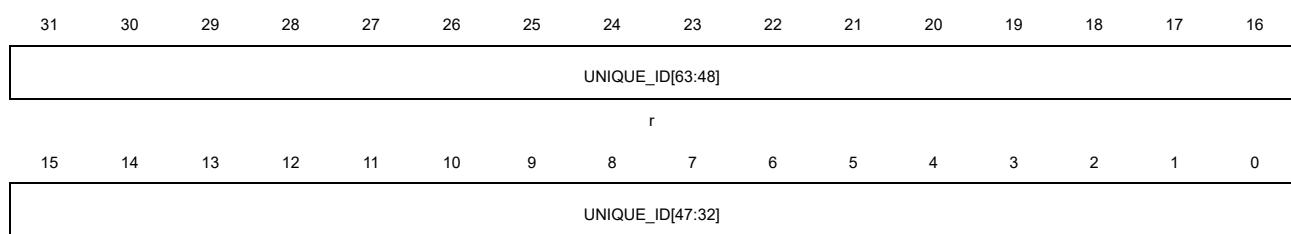
The value is factory programmed and can never be altered by user.



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.													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
r															
UNIQUE_ID[95:80]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
r															
UNIQUE_ID[79:64]															

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:0	UNIQUE_ID[95:64]	Unique device ID

## 1.6. System configuration registers

Base address: 0x4002 103C

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								CEE		Reserved					
rw															

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:8	Reserved	Must be kept at reset value.
7	CEE	Code execution efficiency 0: Default code execution efficiency 1: Code execution efficiency enhancement
6:0	Reserved	Must be kept at reset value.

### NOTE:

- Only bit[7] can be read-modify-write, other bits are not permitted.

## 2. Flash memory controller (FMC)

### 2.1. Introduction

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 first 256K bytes of the flash. It also provides page erase, mass erase, and word/half-word/bit program operations for flash memory.

### 2.2. Main features

- Up to 3072KB of on-chip flash memory for instruction and data;
- No waiting time within first 256K bytes when CPU executes instructions. A long delay when CPU fetches the instructions out of the range;
- 2 banks adopted for GD32F30x\_CL and GD32F30x\_XD. Bank0 is used for the first 512KB and bank1 is for the rest capacity;
- Only bank0 is adopted for GD32F30x\_CL with flash no more than 512KB and GD32F30x\_HD;
- The flash page size is 2KB for bank0, 4KB for bank1;
- Word/half-word/bit 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 description

#### 2.3.1. Flash memory architecture

For GD32F30x\_CL with flash no more than 512KB and GD32F30x\_HD, the page size is 2KB. For GD32F30x\_CL and GD32F30x\_XD, bank0 is used for the first 512KB where the page size is 2KB. Bank1 is used for the rest capacity where the page size is 4KB. Each page can be erased individually.

[Table 2-1. GD32F30x\\_CL and GD32F30x\\_HD, GD32F30x\\_XD base address and size for flash memory](#) shows the details of flash organization.

**Table 2-1. GD32F30x\_CL and GD32F30x\_HD, GD32F30x\_XD base address and size for**

**flash memory**

<b>Block</b>	<b>Name</b>	<b>Address Range</b>	<b>size (bytes)</b>
Main Flash Block	Page 0	0x0800 0000 - 0x0800 07FF	2KB
	Page 1	0x0800 0800 - 0x0800 0FFF	2KB
	Page 2	0x0800 1000 - 0x0800 17FF	2KB
	Page 255	0x0807 F800 - 0x0807 FFFF	2KB
	Page 256	0x0808 0000 - 0x0808 0FFF	4KB
	Page 257	0x0808 1000 - 0x0808 1FFF	4KB
	Page 895	0x082F F000 - 0x082F FFFF	4KB
Information Block	GD32F30x_HD	Boot loader area	0x1FFF F000- 0x1FFF F7FF
	GD32F30x_XD		0x1FFF E000- 0x1FFF F7FF
	GD32F30x_CL		0x1FFF B000- 0x1FFF F7FF
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\_CTLx registers**

After reset, the FMC\_CTLx registers are not accessible in write mode, and the LK bit in FMC\_CTLx 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\_CTL again by setting the LK bit in FMC\_CTL0 register to 1. Any wrong operations to the FMC\_KEY0, 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.

For the GD32F30x\_CL and GD32F30x\_XD, the FMC\_CTL0 register is used to configure the operations to bank0 and the option bytes block, while FMC\_CTL1 register is used to configure the program and erase operations to bank1. The lock/unlock mechanism of FMC\_CTL1 register is similar to FMC\_CTL0 register. The unlock sequence should be written to FMC\_KEY1 when unlocking FMC\_CTL1.

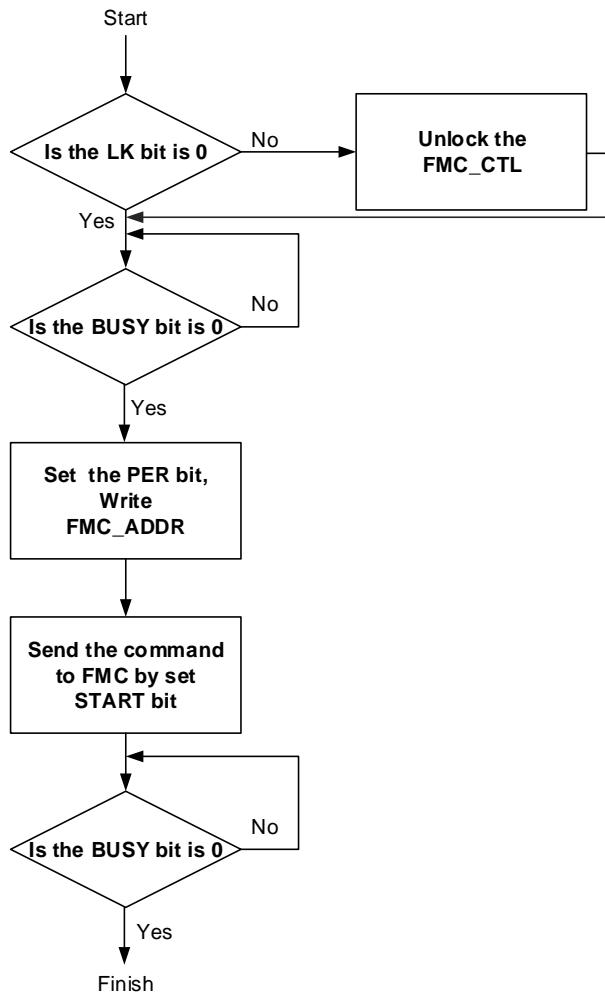
### 2.3.4. Page erase

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.

1. Unlock the FMC\_CTLx registers if necessary;
2. 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;
3. Set the PER bit in FMC\_CTLx registers;
4. Write the page absolute address (0x08XX XXXX) into the FMC\_ADDRx registers;
5. Send the page erase command to the FMC by setting the START bit in FMC\_CTLx registers;
6. Wait until all the operations have finished by checking the value of the BUSY bit in FMC\_STATx registers;
7. Read and verify the page if required using a DBUS access.

When the operation is executed successfully, the ENDF in FMC\_STATx registers is set, and an interrupt will be triggered by FMC if the ENDIE bit in the FMC\_CTLx 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\_CTLx registers is set. The software can check the WPERR bit in the FMC\_STATx 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



For the GD32F30x\_CL and GD32F30x\_XD, FMC\_STAT0 reflects the operation status of bank0, and FMC\_STAT1 reflects the operation status of bank1. The page erase procedure applied to bank1 is similar to the procedure applied to bank0. Especially, when erasing page in bank1 under security protection, the address should not only be written to FMC\_ADDR1 but also to FMC\_ADDR0.

### 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 only on Bank0 by setting MER bit to 1 in the FMC\_CTL0 register, or only on Bank1 by setting MER bit to 1 in the FMC\_CTL1 register, or on entire flash by setting MER bits to 1 in FMC\_CTL0 register and FMC\_CTL1 register. The following steps show the mass erase register access sequence.

1. Unlock the FMC\_CTLx registers if necessary;
2. 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;

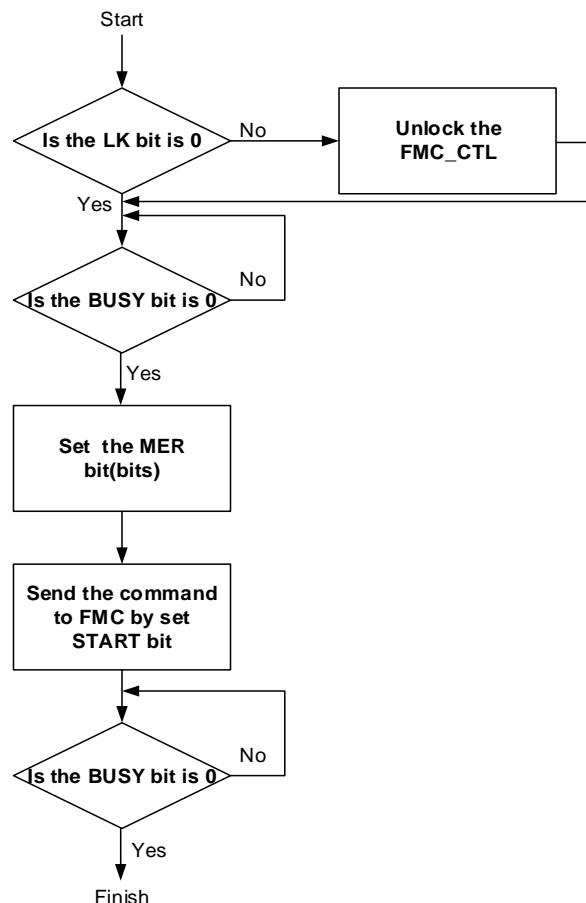
3. Set MER bit in FMC\_CTL0 register if erase Bank0 only. Set MER bit in FMC\_CTL1 register if erase Bank1 only. Set MER bits in in FMC\_CTL0 register and FMC\_CTL1 register if erase entire flash;
4. Send the mass erase command to the FMC by setting the START bit in FMC\_CTL register;
5. Wait until all the operations have been finished by checking the value of the BUSY bit in FMC\_STATx registers;
6. Read and verify the flash memory if required using a DBUS access.

When the operation is executed successfully, the ENDF in FMC\_STATx registers is set, and an interrupt will be triggered by FMC if the ENDIE bit in the FMC\_CTLx 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.

For the GD32F30x\_CL and GD32F30x\_XD, the mass erase procedure applied to bank1 is similar to the procedure applied to bank0.

[\*\*Figure 2-2. Process of mass erase operation\*\*](#) indicates the mass erase operation flow.

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



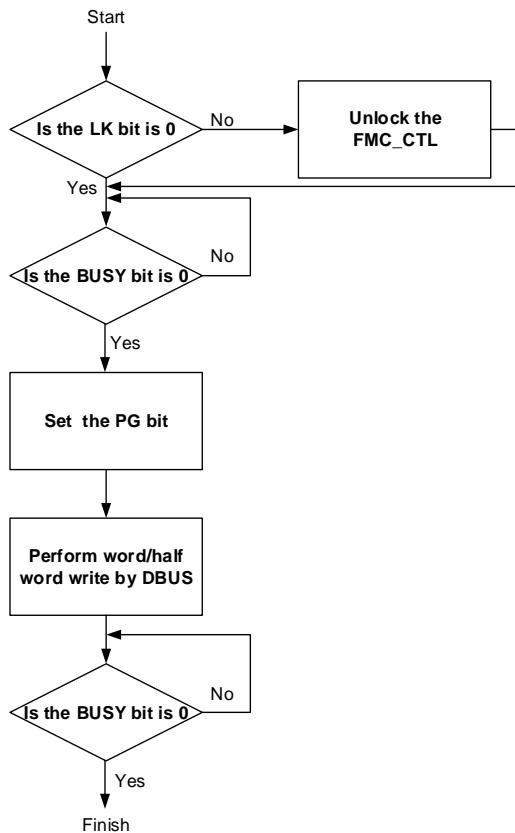
### 2.3.6. Main flash programming

The FMC provides a 32-bit word/16-bit half word/bit 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.

1. Unlock the FMC\_CTLx registers if necessary;
2. 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;
3. Set the PG bit in FMC\_CTLx registers;
4. Write a 32-bit word/16-bit half word to desired absolute address (0x08XX XXXX) by DBUS;
5. Wait until all the operations have been finished by checking the value of the BUSY bit in FMC\_STATx registers;
6. Read and verify the Flash memory if required using a DBUS access.

When the operation is executed successfully, the ENDF in FMC\_STATx registers is set, and an interrupt will be triggered by FMC if the ENDIE bit in the FMC\_CTLx 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\_STATx registers will be set when program the address except programming 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\_STATx is set. In these conditions, a flash operation error interrupt will be triggered by the FMC if the ERRIE bit in the FMC\_CTLx registers is set. The software can check the PGERR bit or WPERR bit in the FMC\_STATx 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



For the GD32F30x\_CL and GD32F30x\_XD, the program procedure applied to bank1 is similar to the procedure applied to bank0.

**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:

1. Unlock the FMC\_CTL0 register if necessary;
2. 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;
3. Unlock the option bytes operation bits in FMC\_CTL0 register if necessary;
4. Wait until OBWEN bit is set in FMC\_CTL0 register;
5. Set OBER bit in FMC\_CTL0 register;
6. Send the option bytes erase command to the FMC by setting the START bit in FMC\_CTL0 register;
7. Wait until all the operations have been finished by checking the value of the BUSY bit in FMC\_STAT0 register;

- 
8. 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.

1. Unlock the FMC\_CTL0 register if necessary;
2. 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;
3. Unlock the option bytes operation bits in FMC\_CTL0 register if necessary;
4. Wait until OBWEN bit is set in FMC\_CTL0 register;
5. Set the OBPG bit in FMC\_CTL0 register;
6. A 32-bit word/16-bit half word write at desired address by DBUS;
7. Wait until all the operations have been finished by checking the value of the BUSY bit in FMC\_STAT0 register;
8. 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](#) is 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

<b>Address</b>	<b>Name</b>	<b>Description</b>
0x1fff f802	USER	[7:4]: reserved [3]: BB 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: generator a reset instead of entering standby mode 1: no reset when entering standby mode [1]: nRST_DPSLP 0: generator 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 2 pages for GD32F30x_HD, GD32F30x_XD and GD32F30x_CL. Bit 0 configures the first 4KB flash protection, and so on. These bits totally controls the first 124KB flash protection. WP[31]: Bit 31 controls the protection of the rest flash memory.
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\_STATx 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/SWD 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\_STATx 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. FMC registers

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 register 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]												w			

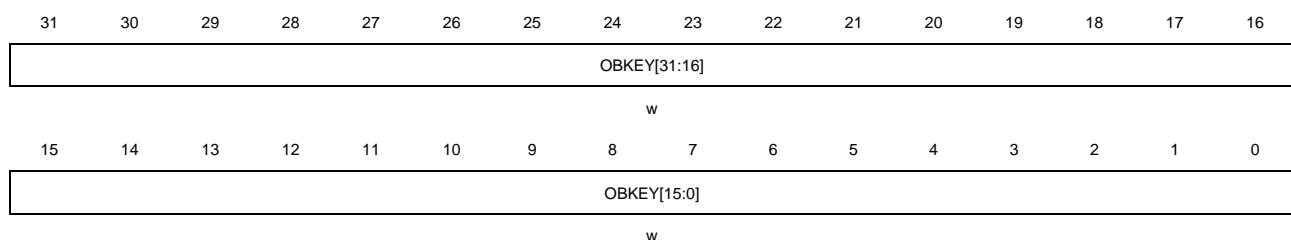
Write KEY[31:0] with keys to unlock FMC\_CTL0 register.

### 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)



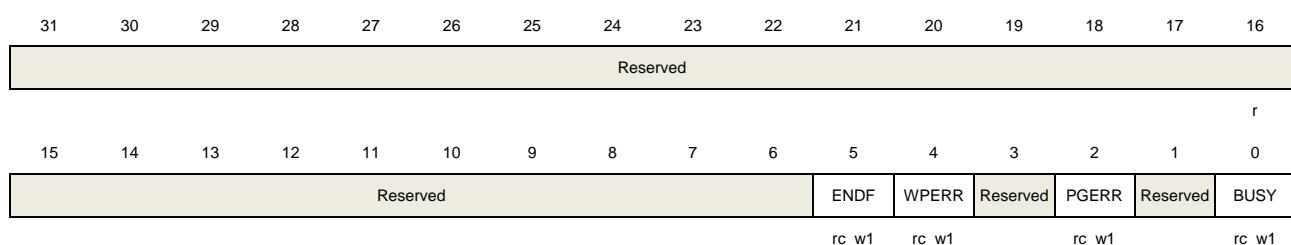
Bits	Fields	Descriptions
31:0	OBKEY[31:0]	FMC_CTL0 option bytes operation unlock register These bits are only be written by software. Write OBKEY[31:0] with keys to unlock option bytes command in FMC_CTL0 register.

### 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)



Bits	Fields	Descriptions
31:6	Reserved	Must be kept at reset value.
5	ENDF	End of operation flag bit When the operation executed successfully, this bit is set by hardware. The software can clear it by writing 1.
4	WPERR	Erase/Program protection error flag bit When 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 is 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 to 0.

### 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)

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

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
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: 0x0XXX XXXX

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



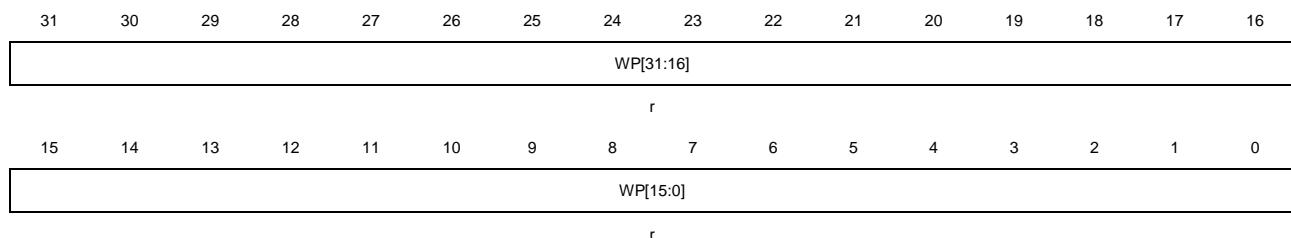
<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:26	Reserved	Must be kept at reset value.
25:10	DATA[15:0]	Store DATA of option bytes block after system reset.
9:2	USER[7:0]	Store USER of option bytes block after system reset.
1	SPC	Option bytes security protection code 0: no protection 1: protection
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. Unlock key register 1(FMC\_KEY1)

Address offset: 0x44

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

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:0	KEY[31:0]	FMC_CTL1 unlock register  These bits are only be written by software.  Write KEY[31:0] with keys to unlock FMC_CTL1 register.

### 2.4.10. Status register 1 (FMC\_STAT1)

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
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															
rc_w1      rc_w1      rc_w1      rc_w1															

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:6	Reserved	Product reserved ID code register.
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 is 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 to 0.

#### 2.4.11. Control register 1(FMC\_CTL1)

Address offset: 0x50

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	Reserved	LK	START	Reserved	Reserved	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
9:8	Reserved	Must be kept at reset value.
7	LK	FMC_CTL1 lock bit This bit is cleared by hardware when right sequence written to FMC_KEY1 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.

5:3	Reserved	Must be kept at reset value
2	MER	Main flash mass erase for bank1 command bit This bit is set or cleared by software. 0: no effect 1: main flash mass erase command for bank1
1	PER	Main flash page erase for bank1 command bit This bit is set or clear by software. 0: no effect 1: main flash page erase command for bank1
0	PG	Main flash program for bank1 command bit This bit is set or clear by software. 0: no effect 1: main flash program command for bank1

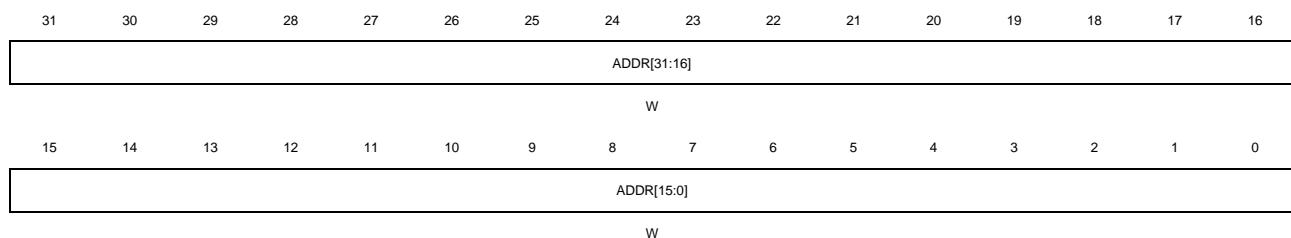
**Note:** This register should be reset after the corresponding flash operation completed.

#### 2.4.12. Address register 1 (FMC\_ADDR1)

Address offset: 0x54

Reset value: 0x0000 0000

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



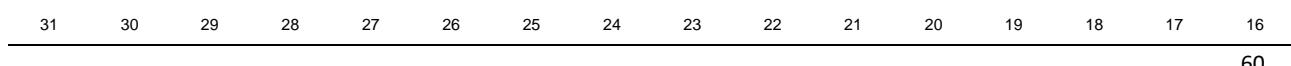
Bits	Fields	Descriptions
31:0	ADDR[31:0]	Flash erase/program command address bits These bits are configured by software. ADDR bits are the address of flash erase/program command

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

Address offset: 0xFC

Reset value: 0x0000 0000

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



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

rw      rw

Bits	Fields	Descriptions
31:2	Reserved	Must be kept at reset value
1	BPEN	FMC bit program enable register This bit set and reset by software. 0: No effect, write page must check if flash is "FF" 1: Write page donot check the flash is FF. The FMC can program each bit
0	WSEN	FMC wait state enable register This bit is set and reset by software. This bit also protected by the FMC_KEYx register. It is necessary to writing 0x45670123 and 0xCDEF89AB to the FMC_KEYx register. 0: no wait state added when fetch flash 1: wait state added when fetch flash

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

Address offset: 0x100

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

Bits	Fields	Descriptions
31:0	PID[31:0]	Product reserved ID code register 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

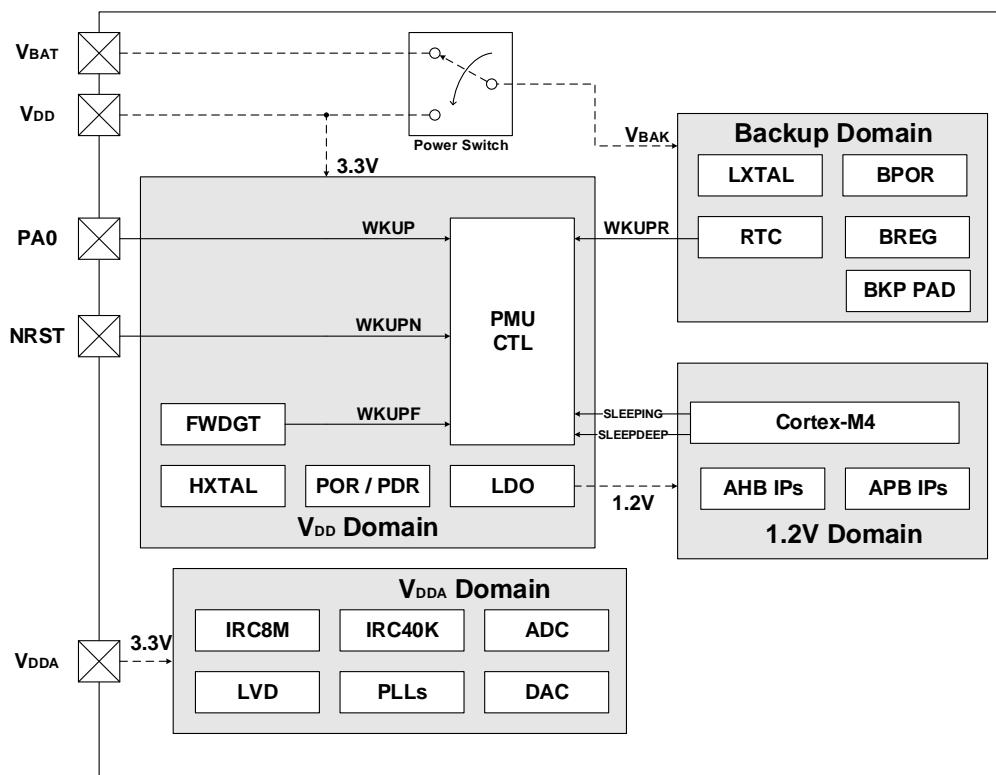
The power consumption is regarded as one of the most important issues for the devices of GD32F30x series. Power management unit (PMU) provides three types of power saving modes, including Sleep, Deep-sleep and Standby mode. These modes reduce the power consumption and allow the application to achieve the best tradeoff among the conflicting demands of CPU operating time, speed and power consumption. For GD32F30x devices, there are three power domains, including  $V_{DD}$  /  $V_{DDA}$  domain, 1.2V domain, and Backup domain, as is shown in the following figure. 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 (LVD) 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.
- Ultra power saving for low-driver mode in Deep-sleep mode. And high-driver mode for high frequency.

#### 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**


LVD: Low Voltage Detector

LDO: Voltage Regulator

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

POR: Power On Reset

PDR: Power Down Reset

BREG: Backup registers

### 3.3.1. 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, power supply for RTC unit, LXTAL oscillator, BPOR and BREG, and three BKP PAD 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 includes 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 Cortex®-M4 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 RTC chapter.
- PC14 and PC15 can be used as either GPIO or LXTAL Crystal oscillator pins.

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

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

**Note:** Since PC13, PC14, 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**

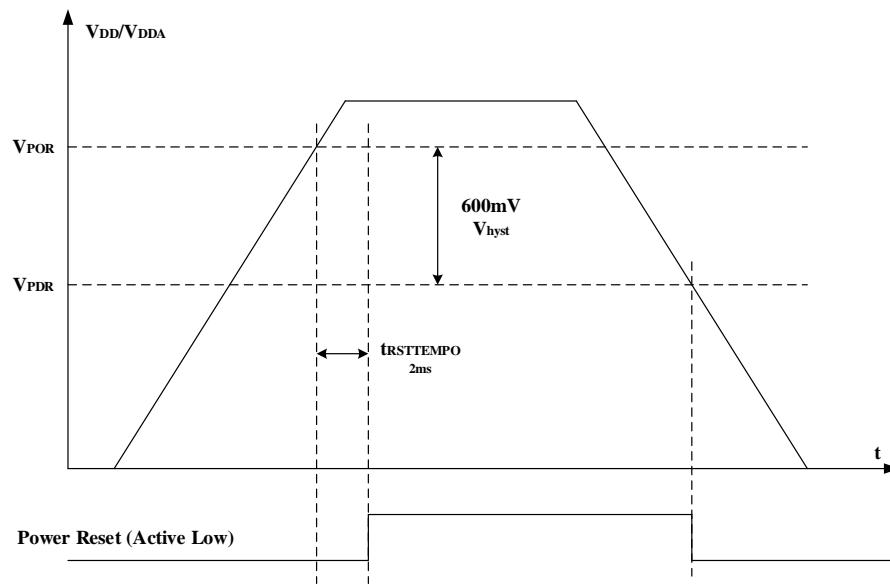
$V_{DD}$  /  $V_{DDA}$  domain includes two parts:  $V_{DD}$  domain and  $V_{DDA}$  domain.  $V_{DD}$  domain includes HXTAL (high speed crystal oscillator), LDO (voltage regulator), POR / PDR (power on / down Reset), FWDGT (free watchdog timer), all pads except PC13 / PC14 / PC15, etc.  $V_{DDA}$  domain includes ADC / DAC (AD / DA Converter), IRC8M (internal 8MHz RC oscillator), IRC48M (internal 48MHz RC oscillator at 48MHz frequency), 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 1.8V, means the threshold of power down reset. The hysteresis voltage ( $V_{hyst}$ ) is around 600mV.

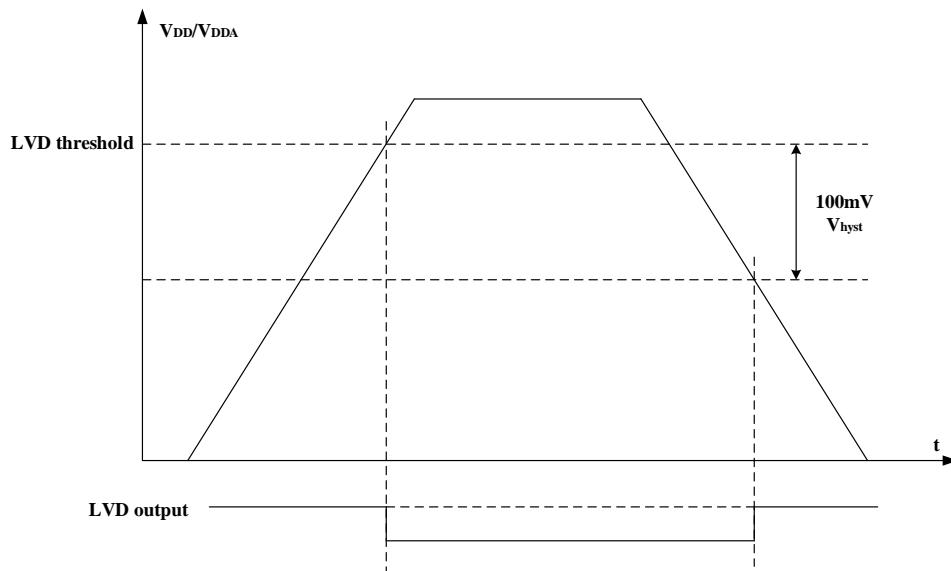
Figure 3-2. Waveform of the POR / PDR



### $V_{DDA}$ domain

The LVD is used to detect whether the  $V_{DD}/V_{DDA}$  supply voltage is lower than a programmed threshold selected by the LVDT[2:0] bits in the Power control register(PMU\_CTL). The LVD is enabled by setting the LVDEN bit, and LVDF bit, which in the Power status register(PMU\_CS), indicates if  $V_{DD}/V_{DDA}$  is higher or lower than the LVD threshold. This event is internally connected to the EXTI line 16 and can generate an interrupt if it is enabled through the EXTI registers. [Figure 3-3. 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 0.3V.

To ensure a high accuracy on ADC and DAC, the ADC/DAC independent external reference voltage should be connected to  $V_{REF+}/V_{REF-}$  pins. According to the different packages,  $V_{REF+}$  pin can be connected to  $V_{DDA}$  pin, or external reference voltage which refers to [Table 12-2. ADC input pins definition](#) and [Table 13-1. DAC I/O description](#),  $V_{REF-}$  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

1.2V power domain supplies power for Cortex®-M4 logic, AHB / APB peripherals, the APB interfaces for the Backup domain and the  $V_{DD} / V_{DDA}$  domain, etc. 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.

#### High-driver mode

If the 1.2V power domain runs with high frequency and opens many functions, it is recommended to enter high-driver mode. The following steps are needed when using high-driver mode.

- IRC8M or HXTAL selected as system clock.
- Set HDEN bit in PMU\_CTL register to 1 to open high-driver mode.
- Wait HDRF bit be set to 1 in PMU\_CS register.
- Set HDS bit in PMU\_CTL register to 1 to switch LDO to high-driver mode.
- Wait HDSRF bit be set to 1 in PMU\_CS register. And enter high-driver mode.
- Running the application at high frequency.

The high-driver mode exit by resetting HDEN and HDS bits in PMU\_CTL register after IRC8M or HXTAL selected as system clock. The high-driver mode exit automatically when exiting from Deep-sleep mode.

### 3.3.4. Power saving modes

After a system reset or a power reset, the GD32F30x 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, and PCLK2) or gating the clocks of the unused peripherals or configuring the LDO output voltage by LDOVS bits in PMU\_CTL register. The LDOVS bits should be configured only when the PLL is off, and the programmed value is selected to drive 1.2V domain after the PLL opened. While the PLL is off, LDO output voltage low mode is selected to drive 1.2V domain. Besides, three power saving modes are provided to achieve even lower power consumption, they are Sleep mode, Deep-sleep mode, and Standby mode.

### Sleep mode

The Sleep mode is corresponding to the SLEEPING mode of the Cortex®-M4. In Sleep mode, only clock of Cortex®-M4 is off. To enter the Sleep mode, it is only necessary to clear the SLEEPDEEP bit in the Cortex®-M4 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 (If SEVONPEND is 1, any interrupt can wake up the system, refer to Cortex-M4 Technical Reference Manual). The mode offers the lowest wakeup time as no time is wasted in interrupt entry or exit.

According to the SLEEPONEXIT bit in the Cortex®-M4 System Control Register, there are two options to select the Sleep mode entry mechanism.

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

### Deep-sleep mode

The Deep-sleep mode is based on the SLEEPDEEP mode of the Cortex®-M4. 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 SLEEPDEEP bit in the Cortex®-M4 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 (If SEVONPEND is 1, any interrupt from EXTI lines can wake up the system, refer to Cortex-M4 Technical Reference Manual). 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.

The low-driver mode in Deep-sleep mode can be entered by configuring the LDEN, LDNP, LDLP, LDOLP bits in the PMU\_CTL register. The Low-driver mode provides lower drive capability, and the Low-power mode take lower power.

**Normal-driver / Normal-power:** The Deep-sleep mode is not in low-driver mode by configure LDEN to 00 in the PMU\_CTL register, and not in low-power mode depending on the LDOLP bit reset in the PMU\_CTL register.

**Normal-driver / Low-power:** The Deep-sleep mode is not in low-driver mode by configure LDEN to 00 in the PMU\_CTL register. The low-power mode enters depending on the LDOLP bit set in the PMU\_CTL register.

**Low-driver / Normal-power:** The low-driver mode in Deep-sleep mode when the LDO in normal-power mode depending on the LDOLP bit reset in the PMU\_CTL register enters by configure LDEN to 0b11 and LDNP to 1 in the PMU\_CTL register.

**Low-driver / Low-power:** The low-driver mode in Deep-sleep mode when the LDO in low-power mode depending on the LDOLP bit set in the PMU\_CTL register enters by configure LDEN to 0b11 and LDLP to 1 in the PMU\_CTL register.

**No Low-driver:** The Deep-sleep mode is not in low-driver mode by configure LDEN to 00 in the PMU\_CTL register.

**Note:** In order to enter Deep-sleep mode smoothly, all EXTI line pending status (in the EXTI\_PD register) and related peripheral flags must be reset, refer to [Table 7-3. EXTI source](#). If not, the program will skip the entry process of Deep-sleep mode to continue to execute the following procedure.

## Standby mode

The Standby mode is based on the SLEEPDEEP mode of the Cortex®-M4, too. In Standby mode, the whole 1.2V domain is power off, the LDO is shut down, and all of IRC8M, HXTAL and PLL are disabled. Before entering the Standby mode, it is necessary to set the SLEEPDEEP bit in the Cortex®-M4 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, 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 Cortex®-M4 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	1. All clocks in the 1.2V domain are off 2. Disable IRC8M, HXTAL and PLL	1. The 1.2V domain is power off 2. Disable IRC8M, HXTAL and PLL
LDO Status	On (normal power mode, normal driver)	On (normal power mode or low power mode ,	Off

<b>Mode</b>	<b>Sleep</b>	<b>Deep-sleep</b>	<b>Standby</b>
	mode)	normal driver mode or low driver mode)	
Configuration	SLEEPDEEP = 0	SLEEPDEEP = 1 STBMOD = 0	SLEEPDEEP = 1 STBMOD = 1, WURST = 1
Entry	WFI or WFE	WFI or WFE	WFI or WFE
Wakeup	Any interrupt for WFI Any event (or interrupt when SEVONPEND is 1) for WFE	Any interrupt from EXTI lines for WFI Any event (or interrupt when SEVONPEND is 1) from EXTI for WFE	<ol style="list-style-type: none"> <li>1. NRST pin</li> <li>2. WKUP pin</li> <li>3. FWDGT reset</li> <li>4. RTC</li> </ol>
Wakeup Latency	None	IRC8M wakeup time, LDO wakeup time added if LDO is in low power mode	Power on sequence

**Note:** In Standby mode, all I / Os are in high-impedance state except NRST pin, PC13 pin when configured for RTC function, PC14 and PC15 pins when used as LXTAL crystal oscillator pins, and WKUP pin if enabled.

## 3.4. PMU registers

PMU base address: 0x4000 7000

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

Address offset: 0x00

Reset value: 0x0000 C000 (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														LDEN[1:0]	HDS	HDEN
rw														rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
LDOVS[1:0]	Reserved	LDNP	LDLP	Reserved	BKPWEN	LVDT[2:0]			LVDEN	STBRST	WURST	STBMOD	LDOLP			
rs		rw	rw		rw	rw			rw	rc_w1	rc_w1	rw	rw			

Bits	Fields	Descriptions
31:20	Reserved	Must be kept at reset value.
19:18	LDEN[1:0]	Low-driver mode enable in Deep-sleep mode 00: Low-driver mode disable in Deep-sleep mode 01: Reserved 10: Reserved 11: Low-driver mode enable in Deep-sleep mode
17	HDS	High-driver mode switch Set this bit by software only when HDRF flag is set and IRC8M or HXTAL used as system clock. After this bit is set, the system enters High-driver mode. This bit can be cleared by software. And cleared by hardware when exit from Deep-sleep mode or when the HDEN bit is clear. 0: No High-driver mode switch 1: High-driver mode switch
16	HDEN	High-driver mode enable This bit is set by software only when IRC8M or HXTAL used as system clock. This bit is cleared by software or by hardware when exit from Deep-sleep mode. 0: High-driver mode disable 1: High-driver mode enable
15:14	LDOVS[1:0]	LDO output voltage select These bits are set by software when the main PLL closed. And the LDO output voltage selected by LDOVS bits takes effect when the main PLL enabled. If the main PLL closed, the LDO output voltage low mode selected. 00: Reserved (LDO output voltage low mode)

		01: LDO output voltage low mode 10: LDO output voltage mid mode 11: LDO output voltage high mode
13:12	Reserved	Must be kept at reset value.
11	LDNP	Low-driver mode when use normal power LDO 0: normal driver when use normal power LDO 1: Low-driver mode enabled when LDEN is 11 and use normal power LDO
10	LDLP	Low-driver mode when use low power LDO. 0: normal driver when use low power LDO 1: Low-driver mode enabled when LDEN is 11 and use low power LDO
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: 2.1V 001: 2.3V 010: 2.4V 011: 2.6V 100: 2.7V 101: 2.9V 110: 3.0V 111: 3.1V
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 Cortex®-M4 enters SLEEPDEEP mode

1: Enter the Standby mode when the Cortex®-M4 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).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved										LDRF[1:0]		HDSRF	HDRF		
										rc_w1		r	r		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	LDOVSRF	Reserved				WUPEN	Reserved				LVDF	STBF	WUF		

Bits	Fields	Descriptions
31:20	Reserved	Must be kept at reset value.
19:18	LDRF[1:0]	<p>Low-driver mode ready flag</p> <p>These bits are set by hardware when enter Deep-sleep mode and the LDO in Low-driver mode. These bits are cleared by software when write 11.</p> <ul style="list-style-type: none"> <li>00: normal driver in Deep-sleep mode</li> <li>01: Reserved</li> <li>10: Reserved</li> <li>11: Low-driver mode in Deep-sleep mode</li> </ul>
17	HDSRF	<p>High-driver switch ready flag</p> <ul style="list-style-type: none"> <li>0: High-driver switch not ready</li> <li>1: High-driver switch ready</li> </ul>
16	HDRF	<p>High-driver ready flag</p> <ul style="list-style-type: none"> <li>0: High-driver not ready</li> <li>1: High-driver ready</li> </ul>
15	Reserved	Must be kept at reset value.
14	LDOVSRF	<p>LDO voltage select ready flag</p> <ul style="list-style-type: none"> <li>0: LDO voltage select not ready</li> <li>1: LDO voltage select ready</li> </ul>
13:9	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 Standby mode, a rising edge on the WKUP pin wakes up the system from the Standby 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 is already high.

7:3	Reserved	Must be kept at reset value.
2	LVDF	<p>Low Voltage Detector Status Flag</p> <p>0: Low Voltage event has not occurred (<math>V_{DD}</math> is higher than the specified LVD threshold)</p> <p>1: Low Voltage event occurred (<math>V_{DD}</math> is equal to or lower than the specified LVD threshold)</p> <p><b>Note:</b> The LVD function is stopped in Standby mode.</p>
1	STBF	<p>Standby Flag</p> <p>0: The device has not entered the Standby mode</p> <p>1: The device has been in the Standby mode</p> <p>This bit is cleared only by a POR / PDR or by setting the STBRST bit in the PMU_CTL register.</p>
0	WUF	<p>Wakeup Flag</p> <p>0: No wakeup event has been received</p> <p>1: Wakeup event occurred from the WKUP pin or the RTC alarm event</p> <p>This bit is cleared only by a POR / PDR or by setting the WURST bit in the PMU_CTL register.</p>

## 4. Backup registers (BKP)

### 4.1. Introduction

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. Main features

- 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 sets the calibration value.
- Tamper control and status register (BKP\_TPCS) can control tamper detection with interrupt or event capability.

### 4.3. Function description

#### 4.3.1. RTC clock calibration

In order to improve the RTC clock accuracy, the MCU provides the RTC output for calibration function. The RTC clock, or a clock with the frequency is  $f_{RTCCLK}/64$ , 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. BKP registers

### 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.

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

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
CALDIR	CCOSEL	Reserved		ROSEL	ASOEN	COEN	RCCV[6:0]								
rw	rw			rw	rw	rw									rw

Bits	Fields	Descriptions
15	CALDIR	RTC clock calibration direction 0: Slowed down 1: Speed up This bit is reset only by a Backup domain reset.
14	CCOSEL	RTC clock output selection 0: RTC clock div 64 1: RTC clock This bit is reset only by a POR.
13:10	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

This bit is reset only by a Backup domain reset.

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. This bit is reset only by a Backup domain reset.
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 the RTC clock or RTC clock divided by 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. This bit is reset only by a POR.
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. This bit is reset only by a Backup domain reset.

#### 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

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 register.

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

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
			Reserved		TIF	TEF		Reserved		TPIE	TIR	TER			

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 This bit is reset only by a system reset and wake-up from Standby mode.
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)

### High- and extra-density reset and clock control unit (RCU)

#### 5.1. Reset control unit (RCTL)

##### 5.1.1. Overview

GD32F30x 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 SW-DP 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\_0004 in the memory map.

###### System reset

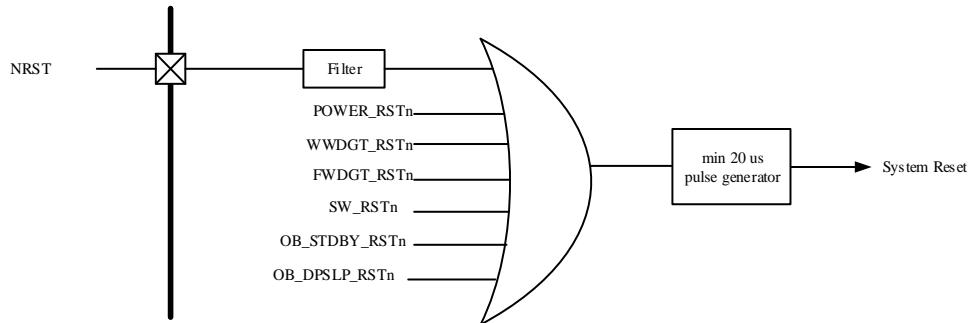
A system reset is generated by the following events:

- A power reset (POWER\_RSTn).
- A external pin reset (NRST).
- A window watchdog timer reset (WWDGT\_RSTn).
- A free watchdog timer reset (FWDGT\_RSTn).
- The SYSRESETREQ bit in Cortex®-M4 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 SW-DP controller and the backup domain.

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

**Figure 5-1. The system reset circuit**



### **Backup domain reset**

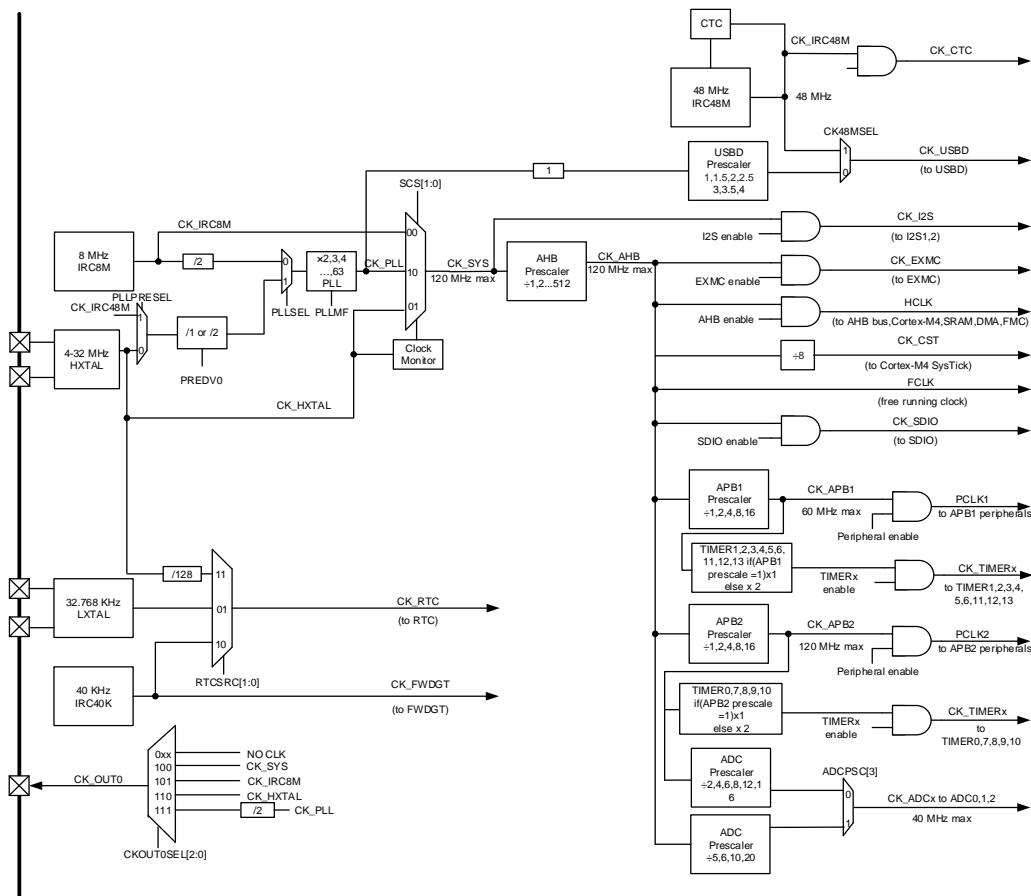
A backup domain reset is generated by setting the BKPRST bit in the backup domain control register or backup domain power on reset ( $V_{DD}$  or  $V_{BAT}$  power on, if both supplies have previously been powered off).

## **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), an Internal 48M RC oscillator (IRC48M), a High Speed crystal oscillator (HXTAL), a Low Speed Internal 40K RC oscillator (IRC40K), a Low Speed crystal oscillator (LXTAL), a Phase Lock Loop (PLL), a HXTAL clock monitor, clock prescalers, clock multiplexers and clock gating circuitry.

The clocks of the AHB, APB and Cortex®-M4 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 120 MHz.

**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 120 MHz / 120 MHz / 60 MHz. The cortex® system timer (systick) external clock is clocked with the AHB clock (HCLK) divided by 8. The systick can work either with this clock or with the 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 or by the clock of AHB divided by 5, 6, 10, 20, which defined by ADCPSC in RCU\_CFG0 and RCU\_CFG1 register.

The SDIO, EXMC are clocked by the clock of CK\_AHB.

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 USBD is clocked by the clock of CK48M. The CK48M is selected from the clock of CK\_PLL or the clock of IRC48M by CK48MSEL bit in RCU\_ADDCTL register.

The CTC is clocked by the clock of IRC48M. The IRC48M can be automatically trimmed by CTC unit.

The I2S is clocked by the clock of CK\_SYS.

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.

### 5.2.2. Characteristics

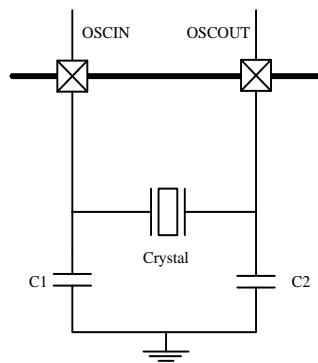
- 4 to 32 MHz high speed crystal oscillator (HXTAL).
- Internal 8 MHz RC oscillator (IRC8M).
- Internal 48 MHz RC oscillator (IRC48M).
- 32,768 Hz low speed crystal oscillator (LXTAL).
- Internal 40KHz RC oscillator (IRC40K).
- PLL clock source can be HXTAL, IRC8M, IRC48M.
- HXTAL clock monitor.

### 5.2.3. Function overview

#### High speed crystal oscillator (HXTAL)

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

**Figure 5-3. HXTAL clock source**

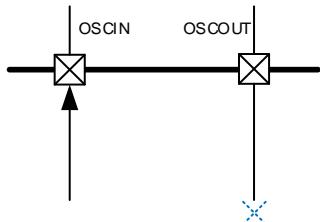


The HXTAL crystal oscillator can be switched on or off using the HXTALEN bit in the control Register RCU\_CTL. The HXTALSTB flag in Control Register RCU\_CTL indicates if the high-speed external crystal oscillator is stable. When the HXTAL is powered up, it will not be released for use until this HXTALSTB 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. During bypass mode, the signal is connected to OSCIN, and OSCOUT remains in the suspended state, as shown in [\*\*Figure 5-4. HXTAL clock source in bypass mode\*\*](#). The CK\_HXTAL is equal to the external clock which drives the OSCIN pin.

**Figure 5-4. HXTAL clock source in bypass mode**



#### 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.

#### Internal 48M RC oscillators (IRC48M)

The internal 48M RC oscillator, IRC48M, has a fixed frequency of 48 MHz. The IRC48M oscillator provides a lower cost type clock source as no external components are required when USBD used. The IRC48M RC oscillator can be switched on or off using the IRC48MEN bit in the RCU\_ADDCTL register. The IRC48MSTB flag in the RCU\_ADDCTL register is used to indicate if the internal 48M RC oscillator is stable. An interrupt can be generated if the related interrupt enable bit, IRC48MSTBIE, in the RCU\_ADDINT register, is set when the IRC48M becomes stable. The IRC48M clock is used for the clocks of USBD.

The frequency accuracy of the IRC48M can be calibrated by the manufacturer, but its operating frequency is still not enough accurate because the USB need the frequency must

between 48MHz with 500ppm accuracy. A hardware automatically dynamic trim performed in CTC unit adjust the IRC48M to the needed frequency.

### Phase locked loop (PLL)

There is one internal Phase Locked Loop, the PLL.

The PLL can be switched on or off by using the PLLLEN 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 PLL is closed by hardware when entering the Deepsleep / Standby mode or HXTAL monitor fail when HXTAL used as the source clock of the PLL.

### 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 fee 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 Cortex-M4. 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 4 MHz to 120 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**

<b>Clock Source 0 Selection bits</b>	<b>Clock Source</b>
0xx	NO CLK
100	CK_SYS
101	CK_IRC8M
110	CK_HXTAL
111	CK_PLL/2

### **Voltage control**

The 1.2V domain voltage in Deep-sleep mode can be controlled by DSLPVS[2:0] bit in the Deep-sleep mode voltage register (RCU\_DSV).

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

<b>DSLPVS[2:0]</b>	<b>Deep-sleep mode voltage(V)</b>
000	1.0
001	0.9
010	0.8
011	0.7

## 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
						PLLSTB	PLL EN					CKMEN	HXTALB PS	HXTALST B	HXTALE N
							r		rw			rw	rw	r	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
													IRC8MST B	IRC8MEN	
													r	rw	

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

---

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

### 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
USBDPSC[2]	PLLMF[5]	Reserved	ADCPSC[2]	PLLMF[4]	CKOUT0SEL[2:0]	USBDPSC[1:0]		PLLMF[3:0]	PREDV0	PLLSEL					
rw	rw		rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADCPSC[1:0]	APB2PSC[2:0]		APB1PSC[2:0]		AHBPSC[3:0]		SCSS[1:0]	SCS[1:0]							
rw	rw		rw		rw		rw	rw	r						

Bits	Fields	Descriptions
31	USBDPSC[2]	Bit 2 of USBDPSC see bits 23:22 of RCU_CFG0
30	PLLMF[5]	Bit 5 of PLLMF see bits 21:18 of RCU_CFG0
29	Reserved	Must be kept at reset value.
28	ADCPSC[2]	Bit 2 of ADCPSC see bits 15:14 of RCU_CFG0
27	PLLMF[4]	Bit 4 of PLLMF see bits 21:18 of RCU_CFG0
26:24	CKOUT0SEL[2:0]	CKOUT0 Clock Source Selection Set and reset by software. 0xx: No clock selected 100: System clock selected 101: Internal 8MHz RC Oscillator clock selected 110: External high speed oscillator clock selected 111: (CK_PLL / 2) clock selected
23:22	USBDPSC[1:0]	USBD clock prescaler selection Set and reset by software to control the USBD clock prescaler value. The USBD clock must be 48MHz. These bits can't be reset if the USBD clock is enabled. 000: CK_USBD = CK_PLL / 1.5 001: CK_USBD = CK_PLL 010: CK_USBD = CK_PLL / 2.5 011: CK_USBD = CK_PLL / 2 100: CK_USBD = CK_PLL / 3 101: CK_USBD = CK_PLL / 3.5 11x: CK_USBD = CK_PLL / 4
21:18	PLLMF[3:0]	The PLL clock multiplication factor Bit 27, bit 30 of RCU_CFG0 and these bits are written by software to define the PLL multiplication factor. <b>Note:</b> The PLL output frequency must not exceed 120 MHz 000000: CK_SYS = CK_PLL x 2

000001: CK\_SYS = CK\_PLL x 3  
 000010: CK\_SYS = CK\_PLL x 4  
 000011: CK\_SYS = CK\_PLL x 5  
 000100: CK\_SYS = CK\_PLL x 6  
 000101: CK\_SYS = CK\_PLL x 7  
 000110: CK\_SYS = CK\_PLL x 8  
 000111: CK\_SYS = CK\_PLL x 9  
 001000: CK\_SYS = CK\_PLL x 10  
 001001: CK\_SYS = CK\_PLL x 11  
 001010: CK\_SYS = CK\_PLL x 12  
 001011: CK\_SYS = CK\_PLL x 13  
 001100: CK\_SYS = CK\_PLL x 14  
 001101: CK\_SYS = CK\_PLL x 15  
 001110: CK\_SYS = CK\_PLL x 16  
 001111: CK\_SYS = CK\_PLL x 16  
 010000: CK\_SYS = CK\_PLL x 17  
 010001: CK\_SYS = CK\_PLL x 18  
 010010: CK\_SYS = CK\_PLL x 19  
 010011: CK\_SYS = CK\_PLL x 20  
 010100: CK\_SYS = CK\_PLL x 21  
 010101: CK\_SYS = CK\_PLL x 22  
 010110: CK\_SYS = CK\_PLL x 23  
 010111: CK\_SYS = CK\_PLL x 24  
 011000: CK\_SYS = CK\_PLL x 25  
 011001: CK\_SYS = CK\_PLL x 26  
 011010: CK\_SYS = CK\_PLL x 27  
 011011: CK\_SYS = CK\_PLL x 28  
 011100: CK\_SYS = CK\_PLL x 29  
 011101: CK\_SYS = CK\_PLL x 30  
 011110: CK\_SYS = CK\_PLL x 31  
 011111: CK\_SYS = CK\_PLL x 32  
 100000: CK\_SYS = CK\_PLL x 33  
 100001: CK\_SYS = CK\_PLL x 34  
 ...  
 111110: CK\_SYS = CK\_PLL x 63  
 111111: CK\_SYS = CK\_PLL x 63

17	PREDV0	PREDV0 division factor  This bit is set and reset by software. These bits can be written when PLL is disable.  0: PREDV0 input source clock not divided 1: PREDV0 input source clock divided by 2
16	PLLSEL	PLL clock source selection  Set and reset by software to control the PLL clock source.

0: (IRC8M / 2) clock selected as source clock of PLL  
 1: HXTAL or IRC48M(PLLPRESEL of RCU\_CFG1 register) selected as source clock of PLL

15:14	ADCPSC[1:0]	ADC clock prescaler selection  These bits, bit 28 of RCU_CFG0 and bit 29 of RCU_CFG1 are written by software to define the ADC prescaler factor. Set and cleared by software.  0000: (CK_APB2 / 2) selected 0001: (CK_APB2 / 4) selected 0010: (CK_APB2 / 6) selected 0011: (CK_APB2 / 8) selected 0100: (CK_APB2 / 2) selected 0101: (CK_APB2 / 12) selected 0110: (CK_APB2 / 8) selected 0111: (CK_APB2 / 16) selected 1x00 : (CK_AHB / 5) selected 1x01 : (CK_AHB / 6) selected 1x10 : (CK_AHB / 10) selected 1x11 : (CK_AHB / 20) selected
13:11	APB2PSC[2:0]	APB2 prescaler selection  Set and reset by software to control the APB2 clock division ratio.  0xx: CK_AHB selected 100: (CK_AHB / 2) selected 101: (CK_AHB / 4) selected 110: (CK_AHB / 8) selected 111: (CK_AHB / 16) selected
10:8	APB1PSC[2:0]	APB1 prescaler selection  Set and reset by software to control the APB1 clock division ratio. Caution: The CK_APB1 output frequency must not exceed 60 MHz.  0xx: CK_AHB selected 100: (CK_AHB / 2) selected 101: (CK_AHB / 4) selected 110: (CK_AHB / 8) selected 111: (CK_AHB / 16) selected
7:4	AHBPSC[3:0]	AHB prescaler selection  Set and reset by software to control the AHB clock division ratio  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	Reserved		PLL STBIC	HXTAL STBIC	IRC8M STBIC	LXTAL STBIC	IRC40K STBIC
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		PLL STBIE	HXTAL STBIE	IRC8M STBIE	LXTAL STBIE	IRC40K STBIE	CKMIF	Reserved		PLL STBIF	HXTAL STBIF	IRC8M STBIF	LXTAL STBIF	IRC40K STBIF	
		rw	rw	rw	rw	rw	r			r	r	r	r	r	

Bits	Fields	Descriptions
31:24	Reserved	Must be kept at reset value.
23	CKMIC	HXTAL Clock Stuck interrupt clear  Write 1 by software to reset the CKMIF flag.  0: Not reset CKMIF flag 1: Reset CKMIF flag
22:21	Reserved	Must be kept at reset value.

20	PLLSTBIC	PLL stabilization interrupt clear Write 1 by software to reset the PLLSTBIF flag. 0: Not reset PLLSTBIF flag 1: Reset PLLSTBIF flag
19	HXTALSTBIC	HXTAL stabilization interrupt clear Write 1 by software to reset the HXTALSTBIF flag. 0: Not reset HXTALSTBIF flag 1: Reset HXTALSTBIF flag
18	IRC8MSTBIC	IRC8M stabilization interrupt clear Write 1 by software to reset the IRC8MSTBIF flag. 0: Not reset IRC8MSTBIF flag 1: Reset IRC8MSTBIF flag
17	LXTALSTBIC	LXTAL stabilization interrupt clear Write 1 by software to reset the LXTALSTBIF flag. 0: Not reset LXTALSTBIF flag 1: Reset LXTALSTBIF flag
16	IRC40KSTBIC	IRC40K stabilization interrupt clear Write 1 by software to reset the IRC40KSTBIF flag. 0: Not reset IRC40KSTBIF flag 1: Reset IRC40KSTBIF flag
15:13	Reserved	Must be kept at reset value.
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	<p>HXTAL clock stuck interrupt flag</p> <p>Set by hardware when the HXTAL clock is stuck.</p> <p>Reset when setting the CKMIC bit by software.</p> <p>0: Clock operating normally 1: HXTAL clock stuck</p>
6:5	Reserved	Must be kept at reset value.
4	PLLSTBIF	<p>PLL stabilization interrupt flag</p> <p>Set by hardware when the PLL is stable and the PLLSTBIE bit is set.</p> <p>Reset when setting the PLLSTBIC bit by software.</p> <p>0: No PLL stabilization interrupt generated 1: PLL stabilization interrupt generated</p>
3	HXTALSTBIF	<p>HXTAL stabilization interrupt flag</p> <p>Set by hardware when the High speed 4 ~ 16 MHz crystal oscillator clock is stable and the HXTALSTBIE bit is set.</p> <p>Reset when setting the HXTALSTBIC bit by software.</p> <p>0: No HXTAL stabilization interrupt generated 1: HXTAL stabilization interrupt generated</p>
2	IRC8MSTBIF	<p>IRC8M stabilization interrupt flag</p> <p>Set by hardware when the Internal 8 MHz RC oscillator clock is stable and the IRC8MSTBIE bit is set.</p> <p>Reset when setting the IRC8MSTBIC bit by software.</p> <p>0: No IRC8M stabilization interrupt generated 1: IRC8M stabilization interrupt generated</p>
1	LXTALSTBIF	<p>LXTAL stabilization interrupt flag</p> <p>Set by hardware when the Low speed 32,768 Hz crystal oscillator clock is stable and the LXTALSTBIE bit is set.</p> <p>Reset when setting the LXTALSTBIC bit by software.</p> <p>0: No LXTAL stabilization interrupt generated 1: LXTAL stabilization interrupt generated</p>
0	IRC40KSTBIF	<p>IRC40K stabilization interrupt flag</p> <p>Set by hardware when the Internal 40kHz RC oscillator clock is stable and the IRC40KSTBIE bit is set.</p> <p>Reset when setting the IRC40KSTBIC bit by software.</p> <p>0: No IRC40K stabilization clock ready interrupt generated 1: IRC40K stabilization interrupt generated</p>

### 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														Reserved	
										TIMER10 RST	TIMER9 RST	TIMER8 RST		Reserved	
										rw	rw	rw			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC2RS T	USART0 RST	TIMER7R ST	SPIORST ST	TIMER0R ST	ADC1RS T	ADC0RS T	PGRST	PFRST	PERST	PDRST	PCRST	PBRST	PARST	Reserved	AFRST
rw	rw	rw	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	TIMER10RST	<p>Timer 10 reset</p> <p>This bit is set and reset by software.</p> <p>0: No reset</p> <p>1: Reset the TIMER10</p>
20	TIMER9RST	<p>Timer 9 reset</p> <p>This bit is set and reset by software.</p> <p>0: No reset</p> <p>1: Reset the TIMER9</p>
19	TIMER8RST	<p>Timer 8 reset</p> <p>This bit is set and reset by software.</p> <p>0: No reset</p> <p>1: Reset the TIMER8</p>
18:16	Reserved	Must be kept at reset value.
15	ADC2RST	<p>ADC2 reset</p> <p>This bit is set and reset by software.</p> <p>0: No reset</p> <p>1: Reset the ADC2</p>
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	TIMER7RST	<p>Timer 7 reset</p> <p>This bit is set and reset by software.</p> <p>0: No reset</p>

		1: Reset the TIMER7
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	PGRST	<p>GPIO port G reset</p> <p>This bit is set and reset by software.</p> <p>0: No reset</p> <p>1: Reset the GPIO port G</p>
7	PFRST	<p>GPIO portF reset</p> <p>This bit is set and reset by software.</p> <p>0: No reset</p> <p>1: Reset the GPIO port F</p>
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> <p>1: Reset the GPIO port C</p>
3	PBRST	<p>GPIO port B reset</p> <p>This bit is set and reset by software.</p>

		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	Reserved	CAN0RS T	Reserved	USBDRS T	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	WWDDGT	Reserved	TIMER13 RST	TIMER12 RST	TIMER11 RST	TIMER6R ST	TIMER5R ST	TIMER4R ST	TIMER3R ST	TIMER2R ST	TIMER1R ST		
	rw	rw		rw	rw	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	Backup interface reset  This bit is set and reset by software. 0: No reset

		1: Reset backup interface
26	Reserved	Must be kept at reset value.
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	Reserved	Must be kept at reset value.
23	USBDRST	<p>USBD reset</p> <p>This bit is set and reset by software.</p> <p>0: No reset</p> <p>1: Reset the USBD</p>
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>

		0: No reset 1: Reset the SPI2
14	SPI1RST	SPI1 reset  This bit is set and reset by software. 0: No reset 1: Reset the SPI1
13:12	Reserved	Must be kept at reset value.
11	WWDGTRST	WWDGT reset  This bit is set and reset by software. 0: No reset 1: Reset the WWDGT
10:9	Reserved	Must be kept at reset value.
8	TIMER13RST	TIMER13 reset  This bit is set and reset by software. 0: No reset 1: Reset the TIMER13
7	TIMER12RST	TIMER12 reset  This bit is set and reset by software. 0: No reset 1: Reset the TIMER12
6	TIMER11RST	TIMER11 reset  This bit is set and reset by software. 0: No reset 1: Reset the TIMER11
5	TIMER6RST	TIMER6 reset  This bit is set and reset by software. 0: No reset 1: Reset the TIMER6
4	TIMER5RST	TIMER5 reset  This bit is set and reset by software. 0: No reset 1: Reset the TIMER5
3	TIMER4RST	TIMER4 reset  This bit is set and reset by software. 0: No reset 1: Reset the TIMER4
2	TIMER3RST	TIMER3 reset  This bit is set and reset by software.

		0: No reset 1: Reset the TIMER3
1	TIMER2RST	TIMER2 reset  This bit is set and reset by software. 0: No reset 1: Reset the TIMER2
0	TIMER1RST	TIMER1 reset  This bit is set and reset by software. 0: No reset 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		SDIOEN	Reserved	EXMCEN	Reserved	CRCEN	Reserved	FMCSPE	N	Reserved	SRAMSP	EN	DMA1EN	DMA0EN
				rw		rw		rw		rw		rw		rw		rw

Bits	Fields	Descriptions
31:11	Reserved	Must be kept at reset value.
10	SDIOEN	SDIO clock enable  This bit is set and reset by software. 0: Disabled SDIO clock 1: Enabled SDIO clock
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	FMCS PEN	<p>FMC clock enable when sleep mode</p> <p>This bit is set and reset by software to enable/disable FMC clock during Sleep mode.</p> <p>0: Disabled FMC clock during Sleep mode</p> <p>1: Enabled FMC clock during Sleep mode</p>
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										TIMER10 EN	TIMER9E N	TIMER8E N	Reserved		
										rw	rw	rw			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC2EN EN	USART0 EN	TIMER7E N	SPI0EN	TIMER0E N	ADC1EN	ADC0EN	PGEN	PFEN	PEEN	PDEN	PCEN	PBEN	PAEN	Reserved	AFEN
rw	rw	rw	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	TIMER10EN	TIMER10 clock enable

		This bit is set and reset by software. 0: Disabled TIMER10 clock 1: Enabled TIMER10 clock
20	TIMER9EN	TIMER9 clock enable  This bit is set and reset by software. 0: Disabled TIMER9 clock 1: Enabled TIMER9 clock
19	TIMER8EN	TIMER8 clock enable  This bit is set and reset by software. 0: Disabled TIMER8 clock 1: Enabled TIMER8 clock
18:16	Reserved	Must be kept at reset value.
15	ADC2EN	ADC2 clock enable  This bit is set and reset by software. 0: Disabled ADC2 clock 1: Enabled ADC2 clock
14	USART0EN	USART0 clock enable  This bit is set and reset by software. 0: Disabled USART0 clock 1: Enabled USART0 clock
13	TIMER7EN	TIMER7 clock enable  This bit is set and reset by software. 0: Disabled TIMER7 clock 1: Enabled TIMER7 clock
12	SPI0EN	SPI0 clock enable  This bit is set and reset by software. 0: Disabled SPI0 clock 1: Enabled SPI0 clock
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	PGEN	<p>GPIO port G clock enable</p> <p>This bit is set and reset by software.</p> <p>0: Disabled GPIO port G clock</p> <p>1: Enabled GPIO port G clock</p>
7	PFEN	<p>GPIO port F clock enable</p> <p>This bit is set and reset by software.</p> <p>0: Disabled GPIO port F clock</p> <p>1: Enabled GPIO port F clock</p>
6	PEEN	<p>GPIO port E clock enable</p> <p>This bit is set and reset by software.</p> <p>0: Disabled GPIO port E clock</p> <p>1: Enabled GPIO port E clock</p>
5	PDEN	<p>GPIO port D clock enable</p> <p>This bit is set and reset by software.</p> <p>0: Disabled GPIO port D clock</p> <p>1: Enabled GPIO port D clock</p>
4	PCEN	<p>GPIO port C clock enable</p> <p>This bit is set and reset by software.</p> <p>0: Disabled GPIO port C clock</p> <p>1: Enabled GPIO port C clock</p>
3	PBEN	<p>GPIO port B clock enable</p> <p>This bit is set and reset by software.</p> <p>0: Disabled GPIO port B clock</p> <p>1: Enabled GPIO port B clock</p>
2	PAEN	<p>GPIO port A clock enable</p> <p>This bit is set and reset by software.</p> <p>0: Disabled GPIO port A clock</p> <p>1: Enabled GPIO port A clock</p>
1	Reserved	Must be kept at reset value.
0	AFEN	<p>Alternate function IO clock enable</p> <p>This bit is set and reset by software.</p> <p>0: Disabled Alternate Function IO clock</p> <p>1: Enabled Alternate Function IO cloc</p>

### 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	Reserved	CAN0EN	Reserved	USBDEN	I2C1EN	I2C0EN	UART4E	UART3E	USART2	USART1	Reserved	
	rw	rw	rw		rw		rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI2EN	SPI1EN	Reserved	WWDGT	Reserved	TIMER13	TIMER12	TIMER11	TIMER6E	TIMER5E	TIMER4E	TIMER3E	TIMER2E	TIMER1E		
	rw	rw	en		rw	en	rw								

Bits	Fields	Descriptions
31:30	Reserved	Must be kept at reset value.
29	DACEN	DAC clock enable This bit is set and reset by software. 0: Disabled DAC clock 1: Enabled DAC clock
28	PMUEN	PMU clock enable This bit is set and reset by software. 0: Disabled PMU clock 1: Enabled PMU clock
27	BKPIEN	Backup interface clock enable This bit is set and reset by software. 0: Disabled backup interface clock 1: Enabled backup interface clock
26	Reserved	Must be kept at reset value.
25	CAN0EN	CAN0 clock enable This bit is set and reset by software. 0: Disabled CAN0 clock 1: Enabled CAN0 clock
24	Reserved	Must be kept at reset value.
23	USBDEN	USBD clock enable This bit is set and reset by software. 0: Disabled USBD clock 1: Enabled USBD clock
22	I2C1EN	I2C1 clock enable This bit is set and reset by software. 0: Disabled I2C1 clock 1: Enabled I2C1 clock
21	I2C0EN	I2C0 clock enable

		This bit is set and reset by software. 0: Disabled I2C0 clock 1: Enabled I2C0 clock
20	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
10:9	Reserved	Must be kept at reset value.
8	TIMER13EN	TIMER13 clock enable  This bit is set and reset by software. 0: Disabled TIMER13 clock 1: Enabled TIMER13 clock

7	TIMER12EN	TIMER12 clock enable This bit is set and reset by software. 0: Disabled TIMER12 clock 1: Enabled TIMER12 clock
6	TIMER11EN	TIMER11 clock enable This bit is set and reset by software. 0: Disabled TIMER11 clock 1: Enabled TIMER11 clock
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 BKPWEN 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				LXTALDRI[1:0]		LXTALBPS		LXTALSTB		LXTALEN	
rw																		

Bits	Fields	Descriptions
31:17	Reserved	Must be kept at reset value.
16	BKPRST	Backup domain reset This bit is set and reset by software. 0: No reset 1: Resets backup domain
15	RTCEN	RTC clock enable This bit is set and reset by software. 0: Disabled RTC clock 1: Enabled RTC clock
14:10	Reserved	Must be kept at reset value.
9:8	RTCSRC[1:0]	RTC 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
7:5	Reserved	Must be kept at reset value.
4:3	LXTALDRI[1:0]	LXTAL drive capability Set and reset by software. Backup domain reset resets this value. 00: Lower driving capability 01: Medium low driving capability 10: Medium high driving capability 11: Higher driving capability (reset value) <b>Note:</b> The LXTALDRI is not in bypass mode.
2	LXTALBPS	LXTAL bypass mode enable Set and reset by software. 0: Disable the LXTAL Bypass mode

1: Enable the LXTAL Bypass mode

1	LXTALSTB	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	IRC40KEN
														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	<p>Software reset flag</p> <p>Set by hardware when a software reset generated.</p> <p>Reset by writing 1 to the RSTFC bit.</p> <p>0: No software reset generated</p> <p>1: Software reset generated</p>
27	PORRSTF	<p>Power reset flag</p> <p>Set by hardware when a power reset generated.</p> <p>Reset by writing 1 to the RSTFC bit.</p> <p>0: No power reset generated</p> <p>1: Power reset generated</p>
26	EPRSTF	<p>External pin reset flag</p> <p>Set by hardware when an external pin reset generated.</p> <p>Reset by writing 1 to the RSTFC bit.</p> <p>0: No external pin reset generated</p> <p>1: External pin reset generated</p>
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. 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	PLL PRES	ADCPSC[ EL 3]													Reserved

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

Bits	Fields	Descriptions
31	Reserved	Must be kept at reset value.
30	PLLPRESEL	PLL clock source preselection 0: HXTAL selected as PLL source clock 1: CK_IRC48M selected as PLL source clock
29	ADCPSC[3]	Bit 3 of ADCPSC see bits 15:14 of RCU_CFG0 and bit 28 of RCU_CFG0
28:0	Reserved	Must be kept at reset value.

### 5.3.12. 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															

rw

Bits	Fields	Descriptions
31:3	Reserved	Must be kept at reset value.
2:0	DSLPVS[2:0]	Deep-sleep mode voltage select These bits are set and reset by software 000: The core voltage is default value in Deep-sleep mode 001: The core voltage is (default value-0.1)V in Deep-sleep mode(customers are not recommended to use it) 010: The core voltage is (default value-0.2)V in Deep-sleep mode(customers are not recommended to use it) 011: The core voltage is (default value-0.3)V in Deep-sleep mode(customers are not recommended to use it) 1xx: Reserved

### 5.3.13. Additional clock control register (RCU\_ADDCTL)

Address offset: 0xC0

Reset value: 0x8000 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
IRC48MCALIB[7:0]								Reserved							
r								r							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								CK48MS EL							
rw															

Bits	Fields	Descriptions
31:24	IRC48MCALIB [7:0]	Internal 48MHz RC oscillator calibration value register  These bits are load automatically at power on.
23:18	Reserved	Must be kept at reset value.
17	IRC48MSTB	Internal 48MHz RC oscillator clock stabilization flag  Set by hardware to indicate if the IRC48M oscillator is stable and ready for use. 0: IRC48M is not stable 1: IRC48M is stable
16	IRC48MEN	Internal 48MHz RC oscillator enable  Set and reset by software. Reset by hardware when entering Deep-sleep or Standby mode. 0: IRC48M disable 1: IRC48M enable
15:2	Reserved	Must be kept at reset value.
0	CK48MSEL	48MHz clock selection  Set and reset by software. This bit used to generate CK48M clock which select IRC48M clock or PLL48M clock. 0: Don't select IRC48M clock(use CK_PLL clock divided by USBDPSC) 1: Select IRC48M clock

### 5.3.14. Additional clock interrupt register (RCU\_ADDINT)

Address offset: 0xCC

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												IRC48MS	Reserved							
w																				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0					
Reserved	IRC48MS	Reserved												IRC48MS	Reserved					
rw																			r	

Bits	Fields	Descriptions
31:23	Reserved	Must be kept at reset value.
22	IRC48MSTBIC	Internal 48 MHz RC oscillator stabilization interrupt clear Write 1 by software to reset the IRC48MSTBIF flag. 0: Not reset IRC48MSTBIF flag 1: Reset IRC48MSTBIF flag
21:15	Reserved	Must be kept at reset value.
14	IRC48MSTBIE	Internal 48 MHz RC oscillator stabilization interrupt enable Set and reset by software to enable/disable the IRC48M stabilization interrupt 0: Disable the IRC48M stabilization interrupt 1: Enable the IRC48M stabilization interrupt
13:7	Reserved	Must be kept at reset value.
6	IRC48MSTBIF	IRC48M stabilization interrupt flag Set by hardware when the Internal 48 MHz RC oscillator clock is stable and the IRC48MSTBIE bit is set. Reset by software when setting the IRC48MSTBIC bit. 0: No IRC48M stabilization interrupt generated 1: IRC48M stabilization interrupt generated
5:0	Reserved	Must be kept at reset value.

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

Address offset: 0xE0

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

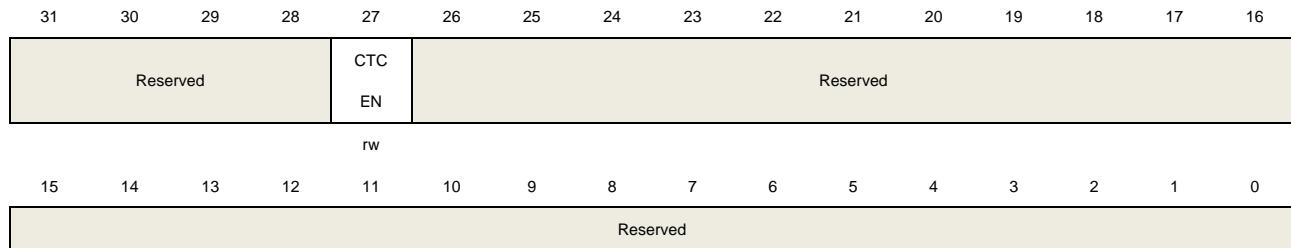
<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:28	Reserved	Must be kept at reset value.
27	CTCRST	CTC reset This bit is set and reset by software. 0: No reset 1: Reset CTC
26:0	Reserved	Must be kept at reset value.

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

Address offset: 0xE4

Reset value: 0x0000 0000

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



<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:28	Reserved	Must be kept at reset value.
27	CTCEN	CTC clock enable This bit is set and reset by software. 0: Disabled CTC clock 1: Enabled CTC clock
26:0	Reserved	Must be kept at reset value.

## Connectivity line devices: reset and clock control unit (RCU)

### 5.4. Reset control unit (RCTL)

#### 5.4.1. Overview

GD32F30x 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 SW-DP controller and the backup domain. The backup domain reset resets the backup domain. These 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.

#### 5.4.2. 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\_0004 in the memory map.

##### System reset

A system reset is generated by the following events:

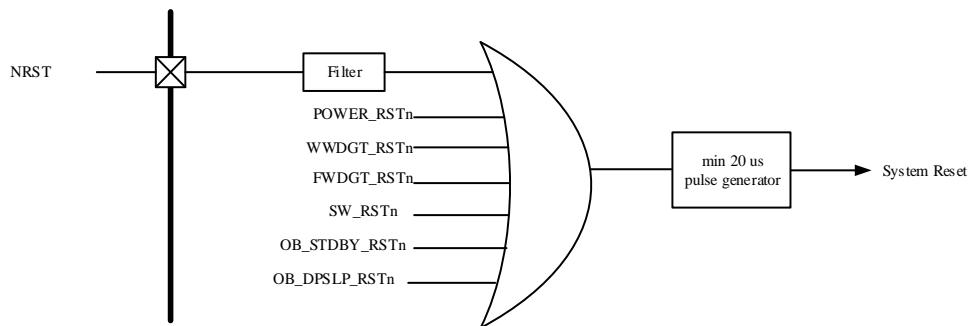
- A power reset (POWER\_RSTn).
- A external pin reset (NRST).
- A window watchdog timer reset (WWDGT\_RSTn).
- A free watchdog timer reset (FWDGT\_RSTn).
- The SYSRESETREQ bit in Cortex®-M4 application interrupt and reset control register is set (SW\_RSTn).
- Reset generated when entering Standby mode 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 SW-DP controller and the backup domain.

A system reset pulse generator guarantees low level pulse duration of 20  $\mu$ s for each reset

source (external or internal reset).

**Figure 5-5. 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).

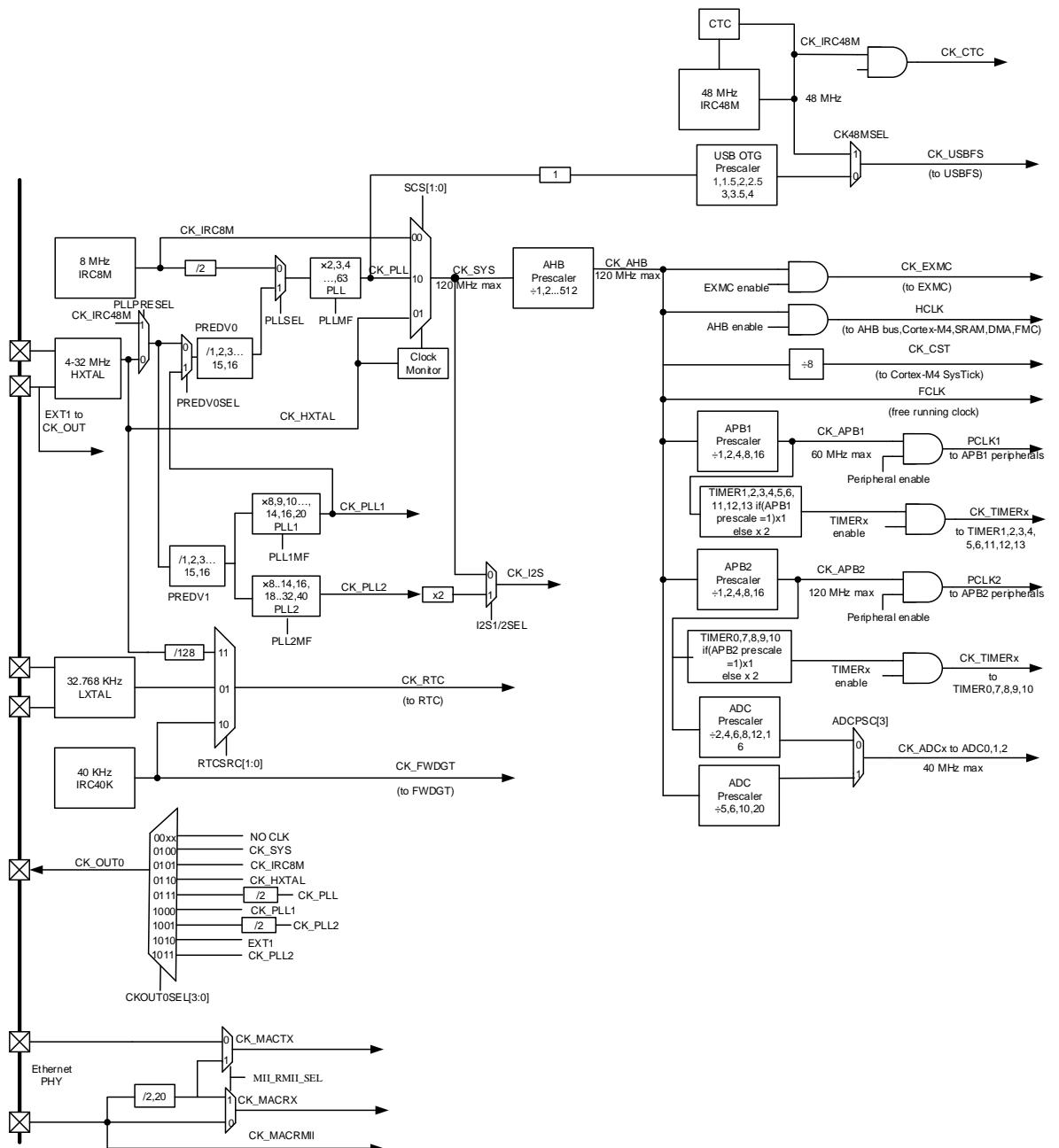
## **5.5. Clock control unit (CCTL)**

### **5.5.1. Overview**

The clock control unit provides a range of frequencies and clock functions. These include a Internal 8M RC oscillator (IRC8M), a Internal 48M RC oscillator (IRC48M), 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 Cortex®-M4 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 120 MHz.

**Figure 5-6. 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 120 MHz / 120 MHz / 60 MHz. The cortex® system timer (systick) external clock is clocked with the AHB clock (HCLK) divided by 8. The systick can work either with this clock or with the 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 or by the clock of AHB divided by 5, 6, 10, 20, which defined by ADCPSC in RCU\_CFG0 and RCU\_CFG1 register.

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 CK48M. The CK48M is selected from the clock of CK\_PLL or the clock of IRC48M by CK48MSEL bit in RCU\_ADDCTL register.

The CTC is clocked by the clock of IRC48M. The IRC48M can be automatically trimmed by CTC unit.

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

The ENET TX / RX are clocked by External PIN (ENET\_TX\_CLK / ENET\_RX\_CLK), which select by ENET\_PHY\_SEL bit in AFIO\_PCF0 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.

### 5.5.2. Characteristics

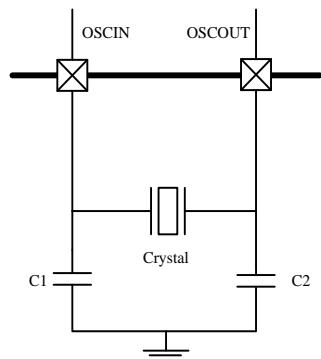
- 4 to 32 MHz high speed crystal oscillator (HXTAL).
- Internal 8 MHz RC oscillator (IRC8M).
- Internal 48 MHz RC oscillator (IRC48M).
- 32,768 Hz Low Speed crystal oscillator (LXTAL).
- Internal 40KHz RC oscillator (IRC40K).
- PLL clock source can be HXTAL, IRC8M orIRC48M.
- HXTAL clock monitor.

### 5.5.3. Function overview

#### High speed crystal oscillator (HXTAL)

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

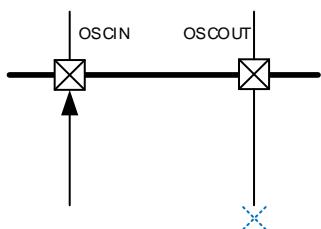
**Figure 5-7. HXTAL clock source**



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. During bypass mode, the signal is connected to OSCIN, and OSCOUT remains in the suspended state, as shown in [Figure 5-4. HXTAL clock source in bypass mode](#). The CK\_HXTAL is equal to the external clock which drives the OSCIN pin.

**Figure 5-8. HXTAL clock source in bypass mode**



#### 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.

### Internal 48M RC oscillators (IRC48M)

The internal 48M RC oscillator, IRC48M, has a fixed frequency of 48 MHz. The IRC48M oscillator provides a lower cost type clock source as no external components are required when USBFS used. The IRC48M RC oscillator can be switched on or off using the IRC48MEN bit in the RCU\_ADDCTL register. The IRC48MSTB flag in the RCU\_ADDCTL register is used to indicate if the internal 48M RC oscillator is stable. An interrupt can be generated if the related interrupt enable bit, IRC48MSTBIE, in the RCU\_ADDINT register, is set when the IRC48M becomes stable. The IRC48M clock is used for the clocks of USBFS.

The frequency accuracy of the IRC48M can be calibrated by the manufacturer, but its operating frequency is still not enough accurate because the USB need the frequency must between 48MHz with 500ppm accuracy. A hardware automatically dynamic trim performed in CTC unit adjust the IRC48M to the needed frequency.

### Phase locked loop (PLL)

There are three internal Phase Locked Loop, the 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 / Standby 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 Cortex®-M4. 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 4 MHz to 120 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 5-3. 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[2:0] bit in the Deep-sleep mode voltage register (RCU\_DSV).

**Table 5-4. 1.2V domain voltage selected in deep-sleep mode**

DSLPVS[2:0]	Deep-sleep mode voltage(V)
000	1.0
001	0.9
010	0.8
011	0.7

## 5.6. Register definition

RCU base address: 0x4002 1000

### 5.6.1. 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	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	
								rw				r	rw		

Bits	Fields	Descriptions
31:30	Reserved	Must be kept at reset value.
29	PLL2STB	<p>PLL2 clock stabilization flag</p> <p>Set by hardware to indicate if the PLL2 output clock is stable and ready for use.</p> <p>0: PLL2 is not stable</p> <p>1: PLL2 is stable</p>
28	PLL2EN	<p>PLL2 enable</p> <p>Set and reset by software. Reset by hardware when entering Deep-sleep or Standby mode.</p> <p>0: PLL2 is switched off</p> <p>1: PLL2 is switched on</p>
27	PLL1STB	<p>PLL1 clock stabilization flag</p> <p>Set by hardware to indicate if the PLL1 output clock is stable and ready for use.</p> <p>0: PLL1 is not stable</p> <p>1: PLL1 is stable</p>
26	PLL1EN	<p>PLL1 enable</p> <p>Set and reset by software. Reset by hardware when entering Deep-sleep or Standby mode.</p> <p>0: PLL1 is switched off</p> <p>1: PLL1 is switched on</p>
25	PLLSTB	<p>PLL clock stabilization flag</p> <p>Set by hardware to indicate if the PLL output clock is stable and ready for use.</p>

		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 ± 1%.</p>

---

2	Reserved	Must be kept at reset value.
1	IRC8MSTB	IRC8M internal 8MHz RC oscillator stabilization flag Set by hardware to indicate if the IRC8M oscillator is stable and ready for use. 0: IRC8M oscillator is not stable 1: IRC8M oscillator is stable
0	IRC8MEN	Internal 8MHz RC oscillator enable Set and reset by software. This bit cannot be reset if the IRC8M clock is used as the system clock. Set by hardware when leaving Deep-sleep or Standby mode or the HXTAL clock is stuck at a low or high state when CKMEN is set. 0: Internal 8 MHz RC oscillator disabled 1: Internal 8 MHz RC oscillator enabled

### 5.6.2. 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
USBFSPSC[2]	PLLMF[5:4]	ADCPSC[2]		CKOUT0SEL[3:0]		USBFSPSC[1:0]		PLLMF[3:0]		PREDV0_LSB		PLLSEL			
rw	rw	rw		rw		rw		rw		rw		rw		rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADCPSC[1:0]		APB2PSC[2:0]		APB1PSC[2:0]		AHPBSC[3:0]		SCSS[1:0]		SCS[1:0]					
rw		rw		rw		rw		r							rw

---

Bits	Fields	Descriptions
31	USBFSPSC[2]	Bit 2 of USBFSPSC see bits 23:22 of RCU_CFG0
30:29	PLLMF[5:4]	Bit 5 and Bit 4 of PLLMF see bits 21:18 of RCU_CFG0
28	ADCPSC[2]	Bit 2 of ADCPSC see bits 15:14 of RCU_CFG0
27:24	CKOUT0SEL[3:0]	CKOUT0 clock source selection Set and reset by software. 00xx: No clock selected 0100: System clock selected 0101: Internal 8MHz RC Oscillator clock selected 0110: External high speed oscillator clock selected 0111: (CK_PLL / 2) clock selected 1000: CK_PLL1 clock selected

		1001: CK_PLL2 clock divided by 2 selected 1010: EXT1 selected, to provide the external clock for ENET 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>000: CK_USBFS = CK_PLL / 1.5 001: CK_USBFS = CK_PLL 010: CK_USBFS = CK_PLL / 2.5 011: CK_USBFS = CK_PLL / 2 100: CK_USBFS = CK_PLL / 3 101: CK_USBFS = CK_PLL / 3.5 11x :CK_USBFS = CK_PLL / 4</p>
21:18	PLLMF[3:0]	<p>The PLL clock multiplication factor</p> <p>Bit 29, bit 30 of RCU_CFG0 and these bits are written by software to define the PLL multiplication factor</p> <p><b>Note:</b> The PLL output frequency must not exceed 120 MHz</p> <p>000000: (PLL source clock x 2) 000001: (PLL source clock x 3) 000010: (PLL source clock x 4) 000011: (PLL source clock x 5) 000100: (PLL source clock x 6) 000101: (PLL source clock x 7) 000110: (PLL source clock x 8) 000111: (PLL source clock x 9) 001000: (PLL source clock x 10) 001001: (PLL source clock x 11) 001010: (PLL source clock x 12) 001011: (PLL source clock x 13) 001100: (PLL source clock x 14) 001101: (PLL source clock x 6.5) 001110: (PLL source clock x 16) 001111: (PLL source clock x 16) 010000: (PLL source clock x 17) 010001: (PLL source clock x 18) 010010: (PLL source clock x 19) 010011: (PLL source clock x 20) 010100: (PLL source clock x 21) 010101: (PLL source clock x 22) 010110: (PLL source clock x 23) 010111: (PLL source clock x 24) 011000: (PLL source clock x 25) 011001: (PLL source clock x 26)</p>

		011010: (PLL source clock x 27) 011011: (PLL source clock x 28) 011100: (PLL source clock x 29) 011101: (PLL source clock x 30) 011110: (PLL source clock x 31) 011111: (PLL source clock x 32) 100000: (PLL source clock x 33) 100001: (PLL source clock x 34)  ... 111110: (PLL source clock x 63) 111111: (PLL source clock x 63)
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 or IRC48M(PLLPRESEL of RCU_CFG1 register) selected as source clock of PLL</p>
15:14	ADCPSC[1:0]	<p>ADC clock prescaler selection</p> <p>These bits, bit 28 of RCU_CFG0 and bit 29 of RCU_CFG1 are written by software to define the ADC prescaler factor. Set and cleared by software.</p> <p>0000: (CK_APB2 / 2) selected 0001: (CK_APB2 / 4) selected 0010: (CK_APB2 / 6) selected 0011: (CK_APB2 / 8) selected 0100: (CK_APB2 / 2) selected 0101: (CK_APB2 / 12) selected 0110: (CK_APB2 / 8) selected 0111: (CK_APB2 / 16) selected 1x00: (CK_AHB / 5) selected 1x01: (CK_AHB / 6) selected 1x10: (CK_AHB / 10) selected 1x11: (CK_AHB / 20) 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</p>

---

		111: (CK_AHB / 16) selected
10:8	APB1PSC[2:0]	APB1 prescaler selection Set and reset by software to control the APB1 clock division ratio. Caution: The CK_APB1 output frequency must not exceed 60 MHz. 0xx: CK_AHB selected 100: (CK_AHB / 2) selected 101: (CK_AHB / 4) selected 110: (CK_AHB / 8) selected 111: (CK_AHB / 16) selected
7:4	AHBPSC[3:0]	AHB prescaler selection Set and reset by software to control the AHB clock division ratio 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

### 5.6.3. 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 STBIC	PLL1 STBIC	PLL STBIC	HXTAL STBIC	IRC8M STBIC	LXTAL STBIC	IRC40K STBIC		
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	HXTAL clock stuck interrupt clear Write 1 by software to reset the CKMIF flag. 0: Not reset CKMIF flag 1: Reset CKMIF flag
22	PLL2STBIC	PLL2 stabilization interrupt clear Write 1 by software to reset the PLL2STBIF flag. 0: Not reset PLL2STBIF flag 1: Reset PLL2STBIF flag
21	PLL1STBIC	PLL1 stabilization interrupt clear Write 1 by software to reset the PLL1STBIF flag. 0: Not reset PLL1STBIF flag 1: Reset PLL1STBIF flag
20	PLLSTBIC	PLL stabilization interrupt clear Write 1 by software to reset the PLLSTBIF flag. 0: Not reset PLLSTBIF flag 1: Reset PLLSTBIF flag
19	HXTALSTBIC	HXTAL stabilization interrupt clear Write 1 by software to reset the HXTALSTBIF flag. 0: Not reset HXTALSTBIF flag 1: Reset HXTALSTBIF flag
18	IRC8MSTBIC	IRC8M stabilization interrupt clear Write 1 by software to reset the IRC8MSTBIF flag. 0: Not reset IRC8MSTBIF flag 1: Reset IRC8MSTBIF flag
17	LXTALSTBIC	LXTAL stabilization interrupt clear Write 1 by software to reset the LXTALSTBIF flag. 0: Not reset LXTALSTBIF flag 1: Reset LXTALSTBIF flag

16	IRC40KSTBIC	IRC40K stabilization interrupt clear Write 1 by software to reset the IRC40KSTBIF flag. 0: Not reset IRC40KSTBIF flag 1: Reset IRC40KSTBIF flag
15	Reserved	Must be kept at reset value.
14	PLL2STBIE	PLL2 stabilization interrupt enable Set and reset by software to enable/disable the PLL2 stabilization interrupt. 0: Disable the PLL2 stabilization interrupt 1: Enable the PLL2 stabilization interrupt
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

### 5.6.4. 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															
										TIMER10 RST	TIMER9 RST	TIMER8 RST	Reserved		
										rw	rw	rw			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	USART0 RST	TIMER7R ST	SPI0RST ST	TIMER0R ST	ADC1RS T	ADC0RS T	PGRST	PFRST	PERST	PDRST	PCRST	PBRST	PARST	Reserved	AFRST
	rw	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	TIMER10RST	<p>Timer 10 reset</p> <p>This bit is set and reset by software.</p> <p>0: No reset</p> <p>1: Reset the TIMER10</p>
20	TIMER9RST	<p>Timer 9 reset</p> <p>This bit is set and reset by software.</p> <p>0: No reset</p> <p>1: Reset the TIMER9</p>
19	TIMER8RST	<p>Timer 8 reset</p> <p>This bit is set and reset by software.</p> <p>0: No reset</p> <p>1: Reset the TIMER8</p>
18:15	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	TIMER7RST	<p>Timer 7 reset</p> <p>This bit is set and reset by software.</p> <p>0: No reset</p> <p>1: Reset the TIMER7</p>
12	SPI0RST	<p>SPI0 reset</p> <p>This bit is set and reset by software.</p> <p>0: No reset</p>

		1: Reset the SPI0
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	PGRST	<p>GPIO port G reset</p> <p>This bit is set and reset by software.</p> <p>0: No reset</p> <p>1: Reset the GPIO port G</p>
7	PFRST	<p>GPIO portF reset</p> <p>This bit is set and reset by software.</p> <p>0: No reset</p> <p>1: Reset the GPIO port F</p>
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> <p>1: Reset the GPIO port C</p>
3	PBRST	<p>GPIO port B reset</p> <p>This bit is set and reset by software.</p> <p>0: No reset</p> <p>1: Reset the GPIO port B</p>
2	PARST	<p>GPIO port A reset</p> <p>This bit is set and reset by software.</p>

		0: No reset 1: Reset the GPIO port A
1	Reserved	Must be kept at reset value.
0	AFRST	<p>Alternate function I/O reset This bit is set and reset by software.</p> <p>0: No reset 1: Reset Alternate Function I/O</p>

### 5.6.5. 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	WWDGT RST	Reserved RST	TIMER13 RST	TIMER12 RST	TIMER11 RST	TIMER6R ST	TIMER5R ST	TIMER4R ST	TIMER3R ST	TIMER2R ST	TIMER1R ST		
	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bits	Fields	Descriptions
31:30	Reserved	Must be kept at reset value.
29	DACRST	<p>DAC reset This bit is set and reset by software.</p> <p>0: No reset 1: Reset DAC unit</p>
28	PMURST	<p>Power control reset This bit is set and reset by software.</p> <p>0: No reset 1: Reset power control unit</p>
27	BKPIRST	<p>Backup interface reset This bit is set and reset by software.</p> <p>0: No reset 1: Reset backup interface</p>
26	CAN1RST	<p>CAN1 reset This bit is set and reset by software.</p> <p>0: No reset</p>

		1: Reset the CAN1
25	CAN0RST	CAN0 reset  This bit is set and reset by software.  0: No reset 1: Reset the CAN0
24:23	Reserved	Must be kept at reset value.
22	I2C1RST	I2C1 reset  This bit is set and reset by software.  0: No reset 1: Reset the I2C1
21	I2C0RST	I2C0 reset  This bit is set and reset by software.  0: No reset 1: Reset the I2C0
20	UART4RST	UART4 reset  This bit is set and reset by software.  0: No reset 1: Reset the UART4
19	UART3RST	UART3 reset  This bit is set and reset by software.  0: No reset 1: Reset the UART3
18	USART2RST	USART2 reset  This bit is set and reset by software.  0: No reset 1: Reset the USART2
17	USART1RST	USART1 reset  This bit is set and reset by software.  0: No reset 1: Reset the USART1
16	Reserved	Must be kept at reset value.
15	SPI2RST	SPI2 reset  This bit is set and reset by software.  0: No reset 1: Reset the SPI2
14	SPI1RST	SPI1 reset  This bit is set and reset by software.  0: No reset

---

		1: Reset the SPI1
13:12	Reserved	Must be kept at reset value.
11	WWDGTRST	<p>WWDGTRST</p> <p>This bit is set and reset by software.</p> <p>0: No reset</p> <p>1: Reset the WWDGT</p>
10:9	Reserved	Must be kept at reset value.
8	TIMER13RST	<p>TIMER13 reset</p> <p>This bit is set and reset by software.</p> <p>0: No reset</p> <p>1: Reset the TIMER13</p>
7	TIMER12RST	<p>TIMER12 reset</p> <p>This bit is set and reset by software.</p> <p>0: No reset</p> <p>1: Reset the TIMER12</p>
6	TIMER11RST	<p>TIMER11 reset</p> <p>This bit is set and reset by software.</p> <p>0: No reset</p> <p>1: Reset the TIMER11</p>
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>

1: Reset the TIMER2

0	TIMER1RST	TIMER1 reset This bit is set and reset by software. 0: No reset 1: Reset the TIMER1
---	-----------	--

### 5.6.6. 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	
																ENETRXE N
																rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
ENETTX EN	ENETEN	Reserved	USBFSE N	Reserved	EXMCEN	Reserved	CRCEN	Reserved	FMCSP N	Reserved	SRAMSP EN	DMA1EN	DMA0EN			
rw	rw		rw		rw		rw		rw		rw		rw		rw	

Bits	Fields	Descriptions
31:17	Reserved	Must be kept at reset value.
16	ENETRXEN	Ethernet RX clock enable This bit is set and reset by software. 0: Disabled Ethernet RX clock 1: Enabled Ethernet RX clock
15	ENETTXEN	Ethernet TX clock enable This bit is set and reset by software. 0: Disabled Ethernet TX clock 1: Enabled Ethernet TX clock
14	ENETEN	Ethernet clock enable This bit is set and reset by software. 0: Disabled Ethernet clock 1: Enabled Ethernet clock
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	<p>EXMC clock enable</p> <p>This bit is set and reset by software.</p> <p>0: Disabled EXMC clock</p> <p>1: Enabled EXMC clock</p>
7	Reserved	Must be kept at reset value.
6	CRCEN	<p>CRC clock enable</p> <p>This bit is set and reset by software.</p> <p>0: Disabled CRC clock</p> <p>1: Enabled CRC clock</p>
5	Reserved	Must be kept at reset value.
4	FMCSPEN	<p>FMC clock enable when sleep mode</p> <p>This bit is set and reset by software to enable/disable FMC clock during Sleep mode.</p> <p>0: Disabled FMC clock during Sleep mode</p> <p>1: Enabled FMC clock during Sleep mode</p>
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>

### 5.6.7. 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															TIMER10 EN	TIMER9E N	TIMER8E N	Reserved	
															rw	rw	rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
Reserved	USART0 EN	TIMER7E N	SPI0EN	TIMER0E N	ADC1EN	ADC0EN	PGEN	PFEN	PEEN	PDEN	PCEN	PBEN	PAEN	Reserved		AFEN			
	rw	rw	rw	rw	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	TIMER10EN	<p>TIMER10 clock enable</p> <p>This bit is set and reset by software.</p> <p>0: Disabled TIMER10 clock</p> <p>1: Enabled TIMER10 clock</p>
20	TIMER9EN	<p>TIMER9 clock enable</p> <p>This bit is set and reset by software.</p> <p>0: Disabled TIMER9 clock</p> <p>1: Enabled TIMER9 clock</p>
19	TIMER8EN	<p>TIMER8 clock enable</p> <p>This bit is set and reset by software.</p> <p>0: Disabled TIMER8 clock</p> <p>1: Enabled TIMER8 clock</p>
18:15	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	TIMER7EN	<p>TIMER7 clock enable</p> <p>This bit is set and reset by software.</p> <p>0: Disabled TIMER7 clock</p> <p>1: Enabled TIMER7 clock</p>
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	<p>TIMER0 clock enable</p> <p>This bit is set and reset by software.</p> <p>0: Disabled TIMER0 clock</p> <p>1: Enabled TIMER0 clock</p>

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	PGEN	GPIO port G clock enable This bit is set and reset by software. 0: Disabled GPIO port G clock 1: Enabled GPIO port G clock
7	PFEN	GPIO port F clock enable This bit is set and reset by software. 0: Disabled GPIO port F clock 1: Enabled GPIO port F clock
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

### 5.6.8. 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	CANOEN	Reserved	I2C1EN	I2C0EN	UART4E	UART3E	USART2	USART1	EN	EN	Reserved
	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	WWDGTE	Reserved	TIMER13	TIMER12	TIMER11	TIMER6E	TIMER5E	TIMER4E	TIMER3E	TIMER2E	TIMER1E		
	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:30	Reserved	Must be kept at reset value.
29	DACEN	DAC clock enable This bit is set and reset by software. 0: Disabled DAC clock 1: Enabled DAC clock
28	PMUEN	PMU clock enable This bit is set and reset by software. 0: Disabled PMU clock 1: Enabled PMU clock
27	BKPIEN	Backup interface clock enable This bit is set and reset by software. 0: Disabled backup interface clock 1: Enabled backup interface clock
26	CAN1EN	CAN1 clock enable This bit is set and reset by software. 0: Disabled CAN1 clock 1: Enabled CAN1 clock
25	CANOEN	CAN0 clock enable This bit is set and reset by software. 0: Disabled CAN0 clock 1: Enabled CAN0 clock

24:23	Reserved	Must be kept at reset value.
22	I2C1EN	<p>I2C1 clock enable</p> <p>This bit is set and reset by software.</p> <p>0: Disabled I2C1 clock</p> <p>1: Enabled I2C1 clock</p>
21	I2C0EN	<p>I2C0 clock enable</p> <p>This bit is set and reset by software.</p> <p>0: Disabled I2C0 clock</p> <p>1: Enabled I2C0 clock</p>
20	UART4EN	<p>UART4 clock enable</p> <p>This bit is set and reset by software.</p> <p>0: Disabled UART4 clock</p> <p>1: Enabled UART4 clock</p>
19	UART3EN	<p>UART3 clock enable</p> <p>This bit is set and reset by software.</p> <p>0: Disabled UART3 clock</p> <p>1: Enabled UART3 clock</p>
18	USART2EN	<p>USART2 clock enable</p> <p>This bit is set and reset by software.</p> <p>0: Disabled USART2 clock</p> <p>1: Enabled USART2 clock</p>
17	USART1EN	<p>USART1 clock enable</p> <p>This bit is set and reset by software.</p> <p>0: Disabled USART1 clock</p> <p>1: Enabled USART1 clock</p>
16	Reserved	Must be kept at reset value.
15	SPI2EN	<p>SPI2 clock enable</p> <p>This bit is set and reset by software.</p> <p>0: Disabled SPI2 clock</p> <p>1: Enabled SPI2 clock</p>
14	SPI1EN	<p>SPI1 clock enable</p> <p>This bit is set and reset by software.</p> <p>0: Disabled SPI1 clock</p> <p>1: Enabled SPI1 clock</p>
13:12	Reserved	Must be kept at reset value.
11	WWDGTEN	<p>WWDGT clock enable</p> <p>This bit is set and reset by software.</p> <p>0: Disabled WWDGT clock</p>

---

		1: Enabled WWDGT clock
10:9	Reserved	Must be kept at reset value.
8	TIMER13EN	<p>TIMER13 clock enable</p> <p>This bit is set and reset by software.</p> <p>0: Disabled TIMER13 clock</p> <p>1: Enabled TIMER13 clock</p>
7	TIMER12EN	<p>TIMER12 clock enable</p> <p>This bit is set and reset by software.</p> <p>0: Disabled TIMER12 clock</p> <p>1: Enabled TIMER12 clock</p>
6	TIMER11EN	<p>TIMER11 clock enable</p> <p>This bit is set and reset by software.</p> <p>0: Disabled TIMER11 clock</p> <p>1: Enabled TIMER11 clock</p>
5	TIMER6EN	<p>TIMER6 clock enable</p> <p>This bit is set and reset by software.</p> <p>0: Disabled TIMER6 clock</p> <p>1: Enabled TIMER6 clock</p>
4	TIMER5EN	<p>TIMER5 clock enable</p> <p>This bit is set and reset by software.</p> <p>0: Disabled TIMER5 clock</p> <p>1: Enabled TIMER5 clock</p>
3	TIMER4EN	<p>TIMER4 clock enable</p> <p>This bit is set and reset by software.</p> <p>0: Disabled TIMER4 clock</p> <p>1: Enabled TIMER4 clock</p>
2	TIMER3EN	<p>TIMER3 clock enable</p> <p>This bit is set and reset by software.</p> <p>0: Disabled TIMER3 clock</p> <p>1: Enabled TIMER3 clock</p>
1	TIMER2EN	<p>TIMER2 clock enable</p> <p>This bit is set and reset by software.</p> <p>0: Disabled TIMER2 clock</p> <p>1: Enabled TIMER2 clock</p>
0	TIMER1EN	<p>TIMER1 clock enable</p> <p>This bit is set and reset by software.</p> <p>0: Disabled TIMER1 clock</p> <p>1: Enabled TIMER1 clock</p>

### 5.6.9. 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 BKPWEN 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			LXTALDRI[1:0]		LXTALBP S	LXTALST B	LXTALEN	
rw					rw				rw			rw	r		

Bits	Fields	Descriptions
31:17	Reserved	Must be kept at reset value.
16	BKPRST	Backup domain reset  This bit is set and reset by software.  0: No reset 1: Resets backup domain
15	RTCEN	RTC clock enable  This bit is set and reset by software.  0: Disabled RTC clock 1: Enabled RTC clock
14:10	Reserved	Must be kept at reset value.
9:8	RTCSRC[1:0]	RTC clock entry selection  Set and reset by software to control the RTC clock source. 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
7:5	Reserved	Must be kept at reset value.
4:3	LXTALDRI[1:0]	LXTAL drive capability  Set and reset by software. Backup domain reset resets this value.  00: Lower driving capability 01: Medium low driving capability

10: Medium high driving capability

11: Higher driving capability (reset value)

**Note:** The LXTALDRI is not in bypass mode.

2	LXTALBPS	LXTAL bypass mode enable Set and reset by software. 0: Disable the LXTAL bypass mode 1: Enable the LXTAL bypass mode
1	LXTALSTB	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

### 5.6.10. 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	
															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	Reset flag clear  This bit is set by software to clear all reset flags.  0: Not clear reset flags 1: Clear reset flags
23:2	Reserved	Must be kept at reset value.
1	IRC40KSTB	IRC40K stabilization flag  Set by hardware to indicate if the IRC40K output clock is stable and ready for use.  0: IRC40K is not stable 1: IRC40K is stable
0	IRC40KEN	IRC40K enable  Set and reset by software.  0: Disable IRC40K 1: Enable IRC40K

### 5.6.11. 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	ENETRS T	Reserved	USBFSR ST									Reserved				

rw                            rw

Bits	Fields	Descriptions
31:15	Reserved	Must be kept at reset value.
14	ENETRST	ENET reset  This bit is set and reset by software. 0: No reset 1: Reset the ENET
13	Reserved	Must be kept at reset value.
12	USBFSRST	USBFS reset  This bit is set and reset by software. 0: No reset 1: Reset the USBFS
11:0	Reserved	Must be kept at reset value.

### 5.6.12. 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
PLL2MF[ 4]	PLLREPS	ADCPSC[ 3]	Reserved								I2S2SEL	I2S1SEL	PREDV0 SEL		
rw	rw	rw									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	PLL2MF[4]	Bit 5 of PLL2MF see bits 15:12 of RCU_CFG1
30	PLLPRESEL	PLL clock source preselection 0: HXTAL selected as PLL source clock 1: CK_IRC48M selected as PLL source clock
29	ADCPSC[3]	Bit 4 of ADCPSC see bits 15:14 of RCU_CFG0 and bit 28 of RCU_CFG0
28:19	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 or IRC48M 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 These bits and bit 31 of RCU_CFG1 are written by software to define the PLL2 multiplication factor. 000xx: reserve 0010x: reserve 00110: (PLL2 source clock x 8) 00111: (PLL2 source clock x 9) 01000: (PLL2 source clock x 10) 01001: (PLL2 source clock x 11) 01010: (PLL2 source clock x 12) 01011: (PLL2 source clock x 13) 01100: (PLL2 source clock x 14) 01101: (PLL2 source clock x 15) 01110: (PLL2 source clock x 16) 01111: (PLL2 source clock x 20) 10000: (PLL2 source clock x 18) 10001: (PLL2 source clock x 19) 10010: (PLL2 source clock x 20) 10011: (PLL2 source clock x 21) 10100: (PLL2 source clock x 22)

		10101: (PLL2 source clock x 23) 10110: (PLL2 source clock x 24) 10111: (PLL2 source clock x 25) 11000 : (PLL2 source clock x 26) 11001: (PLL2 source clock x 27) 11010: (PLL2 source clock x 28) 11011: (PLL2 source clock x 29) 11100: (PLL2 source clock x 30) 11101: (PLL2 source clock x 31) 11110: (PLL2 source clock x 32) 11111: (PLL2 source clock x 40)
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</p>

1111: PREDV2 input source clock divided by 16

3:0	PREDV0[3:0]	PREDV0 division factor This bit is set and reset by software. These bits can be written when PLL is disable.  Note: 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.
		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
		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

### 5.6.13. 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[2:0]	
rw															

Bits	Fields	Descriptions
31:3	Reserved	Must be kept at reset value.
2:0	DSLPVS[2:0]	Deep-sleep mode voltage select These bits are set and reset by software. 000: The core voltage is default value in Deep-sleep mode 001: The core voltage is (default value-0.1)V in Deep-sleep mode(customers are

not recommended to use it)

010: The core voltage is (default value-0.2)V in Deep-sleep mode(customers are not recommended to use it)

011: The core voltage is (default value-0.3)V in Deep-sleep mode(customers are not recommended to use it)

1xx: Reserved

### 5.6.14. Additional clock control register (RCU\_ADDCTL)

Address offset: 0xC0

Reset value: 0x8000 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
IRC48MCALIB[7:0]								Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															

rw

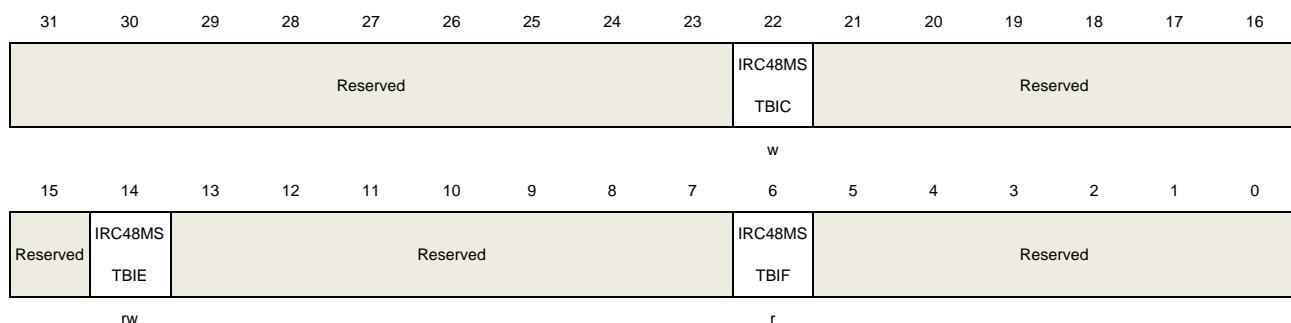
Bits	Fields	Descriptions
31:24	IRC48MCALIB [7:0]	Internal 48MHz RC oscillator calibration value register These bits are load automatically at power on.
23:18	Reserved	Must be kept at reset value.
17	IRC48MSTB	Internal 48MHz RC oscillator clock stabilization flag Set by hardware to indicate if the IRC48M oscillator is stable and ready for use. 0: IRC48M is not stable 1: IRC48M is stable
16	IRC48MEN	Internal 48MHz RC oscillator enable Set and reset by software. Reset by hardware when entering Deep-sleep or Standby mode. 0: IRC48M disable 1: IRC48M enable
15:2	Reserved	Must be kept at reset value.
0	CK48MSEL	48MHz clock selection Set and reset by software. This bit used to generate CK48M clock which select IRC48M clock or PLL48M clock. 0: Don't select IRC48M clock (use CK_PLL clock divided by USBFSPSC) 1: Select IRC48M clock

### 5.6.15. Additional clock interrupt register (RCU\_ADDINT)

Address offset: 0xCC

Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31:23	Reserved	Must be kept at reset value.
22	IRC48MSTBIC	Internal 48 MHz RC oscillator stabilization interrupt clear Write 1 by software to reset the IRC48MSTBIF flag. 0: Not reset IRC48MSTBIF flag 1: Reset IRC48MSTBIF flag
21:15	Reserved	Must be kept at reset value.
14	IRC48MSTBIE	Internal 48 MHz RC oscillator stabilization interrupt enable Set and reset by software to enable/disable the IRC48M stabilization interrupt 0: Disable the IRC48M stabilization interrupt 1: Enable the IRC48M stabilization interrupt
13:7	Reserved	Must be kept at reset value.
6	IRC48MSTBIF	IRC48M stabilization interrupt flag Set by hardware when the Internal 48 MHz RC oscillator clock is stable and the IRC48MSTBIE bit is set. Reset by software when setting the IRC48MSTBIC bit. 0: No IRC48M stabilization interrupt generated 1: IRC48M stabilization interrupt generated
5:0	Reserved	Must be kept at reset value.

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

Address offset: 0xE0

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

Bits	Fields	Descriptions
31:28	Reserved	Must be kept at reset value.
27	CTCRST	CTC reset This bit is set and reset by software. 0: No reset 1: Reset CTC
26:0	Reserved	Must be kept at reset value.

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

Address offset: 0xE4

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

Bits	Fields	Descriptions
31:28	Reserved	Must be kept at reset value.
27	CTCEN	CTC clock enable This bit is set and reset by software. 0: Disabled CTC clock 1: Enabled CTC clock
26:0	Reserved	Must be kept at reset value.

## 6. Clock trim controller (CTC)

### 6.1. Overview

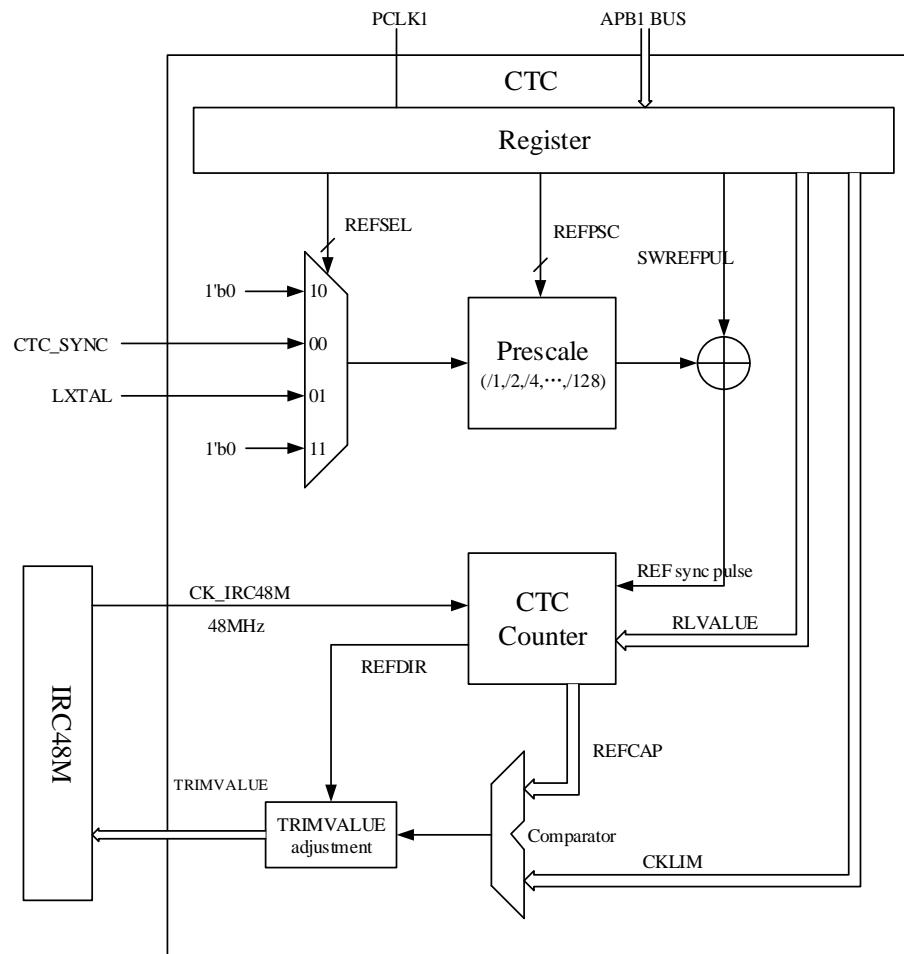
The Clock Trim Controller (CTC) is used to trim internal 48MHz RC oscillator (IRC48M) automatically by hardware. If using IRC48M clock to USBD/USBFS, the IRC48M must be 48 MHz with 500ppm. The internal oscillator without such a high degree of accuracy needs to be trimmed. The CTC unit trim the frequency of the IRC48M based on an external accurate reference signal source. It can automatically adjust the trim value to provide a precise IRC48M clock.

### 6.2. Characteristics

- Two external reference signal sources: GPIO, LXTAL clock
- Provide software reference sync pulse.
- Automatically trimmed by hardware without any software action.
- 16 bits trim counter with reference signal source capture and reload.
- 8 bits clock trim base value to frequency evaluation and automatically trim.
- Enough flag or interrupt to indicate the clock is OK (CKOKIF), warning (CKWARNIF) or error (ERRIF).

### 6.3. Function overview

Figure below provides details on the internal configuration of the CTC.

**Figure 6-1. CTC overview**


### 6.3.1. REF sync pulse generator

Firstly, the reference signal source can select GPIO, LXTAL clock output by setting REFSEL bits in CTC\_CTL1 register.

Secondly, the selected reference signal source uses a configurable polarity by setting REFPOL bit in CTC\_CTL1 register, and can be divided to a suitable frequency with a configurable prescaler by setting REFPSC bits in CTC\_CTL1 register.

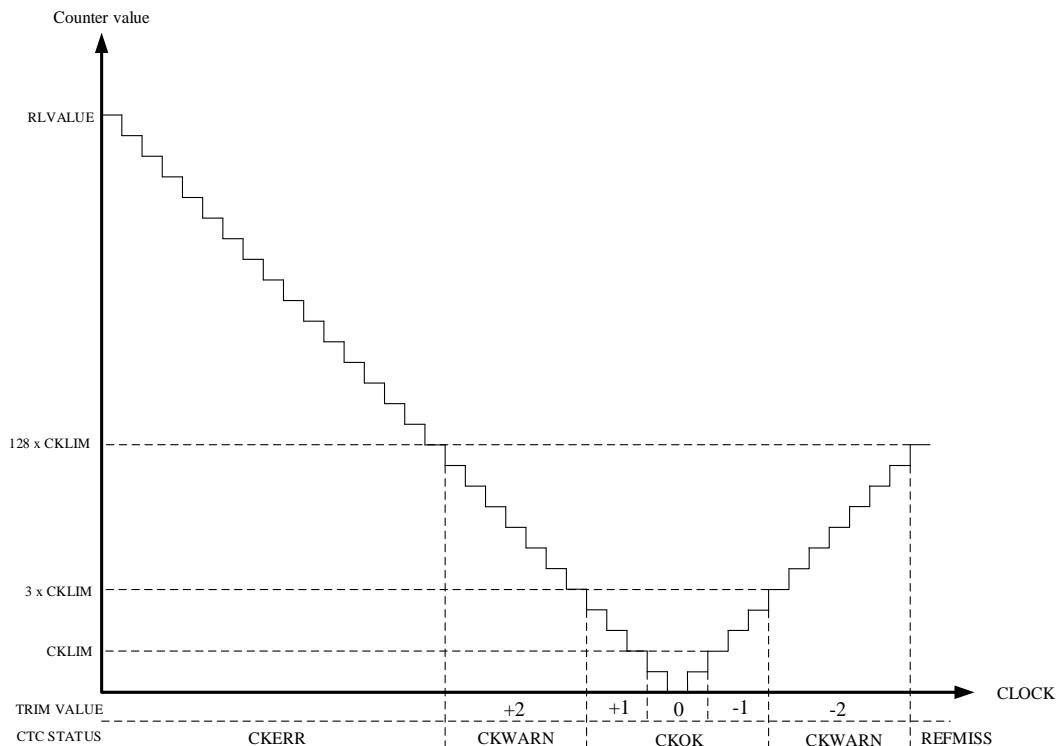
Thirdly, if a software reference pulse needed, write 1 to SWREFPUL bit in CTC\_CTL0 register. The software reference pulse generated in last step is logical OR with the external reference pulse.

### 6.3.2. CTC trim counter

The CTC trim counter is clocked by CK\_IRC48M. After CNTEN bit in CTC\_CTL0 register set, and a first REF sync pulse detected, the counter start down-counting from RLVALUE (defined in CTC\_CTL1 register). If any REF sync pulse detected, the counter reload the RLVALUE and start down-counting again. If no REF sync pulse detected, the counter down-count to

zero, and then up- counting to  $128 \times \text{CKLIM}$  (defined in CTC\_CTL1 register), and then stop until next REF sync pulse detected. If any REF sync pulse detected, the current CTC trim counter value is captured to REFCAP in status register (CTC\_STAT), and the counter direction is captured to REFDIR in status register (CTC\_STAT). The detail is showing as following figure.

**Figure 6-2. CTC trim counter**



### 6.3.3. Frequency evaluation and automatically trim process

The clock frequency evaluation is performed when a REF sync pulse occur. If a REF sync pulse occurs on down-counting, it means the current clock is slower than correct clock (the frequency of 48M). It needs to improve TRIMVALUE in CTC\_CTL0 register. If a REF sync pulse occurs on up-counting, it means the current clock is faster than correct clock (the frequency of 48M). It needs to reduce TRIMVALUE in CTC\_CTL0 register. The CKOKIF, CKWARNIF, CKERR and REFMIS in CTC\_STAT register shows the frequency evaluation scope.

If the AUTOTRIM bit in CTC\_CTL0 register is setting, the automatically hardware trim mode enabled. In this mode, if a REF sync pulse occurs on down-counting, it means the current clock is slower than correct clock, the TRIMVALUE will be increased automatically to raise the clock frequency. Vice versa when it occurs on up-counting, the TRIMVALUE will be reduced automatically to reduce the clock frequency.

- Counter < CKLIM when REF sync pulse is detected.

The CKOKIF in CTC\_STAT register set, and an interrupt generated if CKOKIE bit in CTC\_CTL0 register is 1.

If the AUTOTRIM bit in CTC\_CTL0 register set, the TRIMVALUE in CTC\_CTL0 register is not changed.

- CKLIM  $\leq$  Counter  $< 3 \times$  CKLIM when REF sync pulse is detected.

The CKOKIF in CTC\_STAT register set, and an interrupt generated if CKOKIE bit in CTC\_CTL0 register is 1.

If the AUTOTRIM bit in CTC\_CTL0 register set, the TRIMVALUE in CTC\_CTL0 register add 1 when down-counting or sub 1 when up-counting.

- $3 \times$  CKLIM  $\leq$  Counter  $< 128 \times$  CKLIM when REF sync pulse is detected.

The CKWARNIF in CTC\_STAT register set, and an interrupt generated if CKWARNIE bit in CTC\_CTL0 register is 1.

If the AUTOTRIM bit in CTC\_CTL0 register set, the TRIMVALUE in CTC\_CTL0 register add 2 when down-counting or sub 2 when up-counting.

- Counter  $\geq 128 \times$  CKLIM when down-counting when a REF sync pulse is detected.

The CKERR in CTC\_STAT register set, and an interrupt generated if ERRIE bit in CTC\_CTL0 register is 1.

The TRIMVALUE in CTC\_CTL0 register is not changed

- Counter =  $128 \times$  CKLIM when up-counting.

The REFMISS in CTC\_STAT register set, and an interrupt generated if ERRIE bit in CTC\_CTL0 register is 1.

The TRIMVALUE in CTC\_CTL0 register is not changed.

If adjusting the TRIMVALUE in CTC\_CTL0 register over the value of 63, the overflow will be occurred, while adjusting the TRIMVALUE under the value of 0, the underflow will be occurred. The TRIMVALUE is in the range 0 to 63 (the TRIMVALUE is 63 if overflow, the TRIMVALUE is 0 if underflow). Then, the TRIMERR in CTC\_STAT register will be set, and an interrupt generated if ERRIE bit in CTC\_CTL0 register is 1.

#### 6.3.4. Software program guide

The RLVALUE and CKLIM bits in CTC\_CTL1 register is critical to evaluate the clock frequency and automatically hardware trim. The value is calculated by the correct clock frequency (IRC48M:48 MHz) and the frequency of REF sync pulse. The ideal case is REF sync pulse occur when the CTC counter is zero, so the RLVALUE is:

$$RLVALUE = (F_{clock} \div F_{REF}) - 1 \quad (6-1)$$

The CKLIM is set by user according to the clock accuracy. It is recommended to set to the half of the step size, so the CKLIM is:

$$CKLIM = (F_{clock} \div F_{REF}) \times 0.12\% \div 2 \quad (6-2)$$

The typical step size is 0.12%. Where the  $F_{clock}$  is the frequency of correct clock (IRC48M), the  $F_{REF}$  is the frequency of reference sync pulse.

## 6.4. Register definition

CTC base address: 0x4000 C800

#### 6.4.1. Control register 0 (CTC\_CTL0)

Address offset: 0x00

Reset value: 0x0000 2000

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

31	30	29	28	27	26	25	24	23	22	20	19	18	17	16	15
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	TRIMVALUE[5:0]					SWREF PUL	AUTO TRIM	CNTEN	Reserved	EREFIE	ERRIE	CKWARN IE	CKOKIE		

Bits	Fields	Descriptions
31:14	Reserved	Must be kept at reset value.
13:8	TRIMVALUE[5:0]	<p>IRC48M trim value</p> <p>When AUTOTRIM in CTC_CTL0 register is 0, these bits are set and cleared by software. This mode used to software calibration.</p> <p>When AUTOTRIM in CTC_CTL0 register is 1, these bits are read only. The value automatically modified by hardware. This mode used to hardware trim.</p> <p>The middle value is 32. When increase 1, the IRC48M clock frequency add around 57KHz. When decrease 1, the IRC48M clock frequency sub around 57KHz.</p>
7	SWREFPUL	<p>Software reference source sync pulse</p> <p>This bit is set by software, and generates a reference sync pulse to CTC counter.</p> <p>This bit is cleared by hardware automatically and read as 0.</p> <p>0: No effect</p> <p>1: generates a software reference source sync pulse</p>
6	AUTOTRIM	<p>Hardware automatically trim mode</p> <p>This bit is set and cleared by software. When this bit is set, the hardware automatic trim enabled, the TRIMVALUE bits in CTC_CTL0 register are modified by hardware automatically, until the frequency of IRC48M clock is close to 48MHz.</p> <p>0: Hardware automatic trim disabled</p> <p>1: Hardware automatic trim enabled</p>
5	CNTEN	<p>CTC counter enable</p> <p>This bit is set and cleared by software. This bit used to enable or disable the CTC trim counter. When this bit is set, the CTC_CTL1 register cannot be modified.</p>

		0: CTC trim counter disabled 1: CTC trim counter enabled.
4	Reserved	Must be kept at reset value.
3	EREFIE	EREFIF interrupt enable 0: EREFIF interrupt disable 1: EREFIF interrupt enable
2	ERRIE	Error (ERRIF) interrupt enable 0: ERRIF interrupt disable 1: ERRIF interrupt enable
1	CKWARNIE	Clock trim warning (CKWARNIF) interrupt enable 0: CKWARNIF interrupt disable 1: CKWARNIF interrupt enable
0	CKOKIE	Clock trim OK (CKOKIF) interrupt enable 0: CKOKIF interrupt disable 1: CKOKIF interrupt enable

### 6.4.2. Control register 1 (CTC\_CTL1)

Address offset: 0x04

Reset value: 0x2022 BB7F

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

This register cannot be modified when CNTEN is 1.

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

Bits	Fields	Descriptions
31	REFPOL	Reference signal source polarity This bit is set and cleared by software to select reference signal source polarity 0: rising edge selected 1: falling edge selected
30	Reserved	Must be kept at reset value.
29:28	REFSEL[1:0]	Reference signal source selection These bits are set and cleared by software to select reference signal source.

		00: GPIO selected 01: LXTAL clock selected 10: Reserved, equals 0 selected 11: Reserved, equals 0 selected.
27	Reserved	Must be kept at reset value.
26:24	REFPSC[2:0]	Reference signal source prescaler These bits are set and cleared by software 000: Reference signal not divided 001: Reference signal divided by 2 010: Reference signal divided by 4 011: Reference signal divided by 8 100: Reference signal divided by 16 101: Reference signal divided by 32 110: Reference signal divided by 64 111: Reference signal divided by 128
23:16	CKLIM[7:0]	Clock trim base limit value These bits are set and cleared by software to define the clock trim base limit value. These bits used to frequency evaluation and automatically trim process. Please refer to the <a href="#"><b>Frequency evaluation and automatically trim process</b></a> for detail.
15:0	RLVALUE[15:0]	CTC counter reload value These bits are set and cleared by software to define the CTC counter reload value. These bits reload to CTC trim counter when a reference sync pulse received to start or restart the counter.

#### 6.4.3. Status register (CTC\_STAT)

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
REFCAP[15:0]															
									r						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REFDIR	Reserved			TRIM ERR	REF MISS	CKERR	Reserved			EREFIF	ERRIF	CKWARN IF	CKOK IF		
r				r	r	r				r	r	r	r		

Bits	Fields	Descriptions
31:16	REFCAP[15:0]	CTC counter capture when reference sync pulse.

		When a reference sync pulse occurred, the CTC trim counter value is captured to REFCAP bits.
15	REFDIR	<p>CTC trim counter direction when reference sync pulse</p> <p>When a reference sync pulse occurred during the counter is working, the CTC trim counter direction is captured to REFDIR bit.</p> <p>0: Up-counting 1: Down-counting</p>
14:11	Reserved	Must be kept at reset value.
10	TRIMERR	<p>Trim value error bit</p> <p>This bit is set by hardware when the TRIMVALUE in CTC_CTL0 register overflow or underflow. When the ERRIE in CTC_CTL0 register is set, an interrupt occurs. This bit is cleared by writing 1 to ERRIC bit in CTC_INTC register.</p> <p>0: No trim value error occur 1: Trim value error occur</p>
9	REFMISS	<p>Reference sync pulse miss</p> <p>This bit is set by hardware when the reference sync pulse miss. This is occur when the CTC trim counter reach to 128 x CKLIM during up counting and no reference sync pulse detected. This means the clock is too fast to be trimmed to correct frequency or other error occur. When the ERRIE in CTC_CTL0 register is set, an interrupt occurs. This bit is cleared by writing 1 to ERRIC bit in CTC_INTC register.</p> <p>0: No Reference sync pulse miss occur 1: Reference sync pulse miss occur</p>
8	CKERR	<p>Clock trim error bit</p> <p>This bit is set by hardware when the clock trim error occurs. This is occur when the CTC trim counter greater or equal to 128 x CKLIM during down counting when a reference sync pulse detected. This means the clock is too slow and cannot be trimmed to correct frequency. When the ERRIE in CTC_CTL0 register is set, an interrupt occurs. This bit is cleared by writing 1 to ERRIC bit in CTC_INTC register.</p> <p>0: No Clock trim error occur 1: Clock trim error occur</p>
7:4	Reserved	Must be kept at reset value.
3	EREFIF	<p>Expect reference interrupt flag</p> <p>This bit is set by hardware when the CTC counter reach to 0. When the EREFIE in CTC_CTL0 register is set, an interrupt occurs. This bit is cleared by writing 1 to EREFIC bit in CTC_INTC register.</p> <p>0: No Expect reference occur 1: Expect reference occur</p>
2	ERRIF	<p>Error interrupt flag</p> <p>This bit is set by hardware when an error occurred. If any error of TRIMERR, REFMISS or CKERR occurred, this bit will be set. When the ERRIE in CTC_CTL0</p>

register is set, an interrupt occur. This bit is cleared by writing 1 to ERRIC bit in CTC\_INTC register.

0: No Error occur

1: An error occur

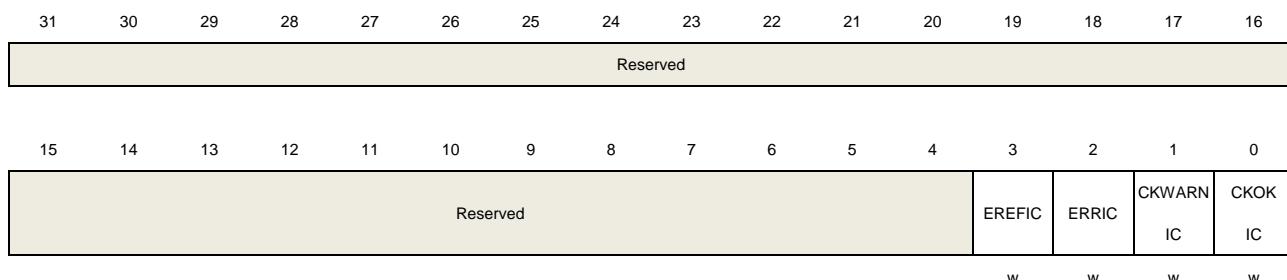
1	CKWARNIF	Clock trim warning interrupt flag  This bit is set by hardware when a clock trim warning occurred. If the CTC trim counter greater or equal to 3 x CKLIM and smaller to 128 x CKLIM when a reference sync pulse detected, this bit will be set. This means the clock is too slow or too fast, but can be trim to correct frequency. The TRIMVALUE add 2 or sub 2 when a clock trim warning occurred. When the CKWARNIE in CTC_CTL0 register is set, an interrupt occurs. This bit is cleared by writing 1 to CKWARNIC bit in CTC_INTC register.  0: No Clock trim warning occur 1: Clock trim warning occur
0	CKOKIF	Clock trim OK interrupt flag  This bit is set by hardware when the clock trim is OK. If the CTC trim counter smaller to 3 x CKLIM when a reference sync pulse detected, this bit will be set. This means the clock is OK to use. The TRIMVALUE need not to adjust or adjust one step. When the CKOKIE in CTC_CTL0 register is, an interrupt occurs. This bit is cleared by writing 1 to CKOKIC bit in CTC_INTC register.  0: No Clock trim OK occur 1: Clock trim OK occur

#### 6.4.4. Interrupt clear register (CTC\_INTC)

Address offset: 0x0C

Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31:4	Reserved	Must be kept at reset value.
3	EREFIC	EREFIF interrupt clear bit  This bit is written by software and read as 0. Write 1 to clear EREFIF bit in

CTC\_STAT register. Write 0 is no effect.

2	ERRIC	ERRIF interrupt clear bit This bit is written by software and read as 0. Write 1 to clear ERRIF, TRIMERR, REFMIS and CKERR bits in CTC_STAT register. Write 0 is no effect.
1	CKWARNIC	CKWARNIF interrupt clear bit This bit is written by software and read as 0. Write 1 to clear CKWARNIF bit in CTC_STAT register. Write 0 is no effect.
0	CKOKIC	CKOKIF interrupt clear bit This bit is written by software and read as 0. Write 1 to clear CKOKIF bit in CTC_STAT register. Write 0 is no effect.

## 7. Interrupt/event controller (EXTI)

### 7.1. Overview

Cortex-M4 integrates the Nested Vectored Interrupt Controller (NVIC) for efficient exception and interrupts processing. NVIC facilitates low-latency exception and interrupt handling and controls power management. It's tightly coupled to the processor core. More details about NVIC could be referred to the technical reference manual of cortex-M4.

EXTI (interrupt/event controller) contains up to 20 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.

### 7.2. Characteristics

- Cortex-M4 system exception.
- Up to 68 maskable peripheral interrupts.
- 4 bits interrupt priority configuration—16 priority levels.
- Efficient interrupt processing.
- Support exception pre-emption and tail-chaining.
- Wake up system from power saving mode.
- Up to 20 independent edge detectors in EXTI.
- Three trigger types: rising, falling and both edges.
- Software interrupt or event trigger.
- Trigger sources configurable.

### 7.3. Interrupts function overview

The Arm® Cortex®-M4 processor and the Nested Vectored Interrupt Controller (NVIC) prioritize and handle all exceptions in Handler Mode. The processor state is automatically stored to the stack on an exception and automatically restored from the stack at the end of the Interrupt Service Routine (ISR).

The vector is fetched in parallel to the state saving, enabling efficient interrupt entry. The processor supports tail-chaining, which enables back-to-back interrupts to be performed without the overhead of state saving and restoration. The following tables list all exception types.

Table 7-1. NVIC exception types in Cortex-M4

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

The SysTick calibration value is 15000 and SysTick clock frequency is fixed to HCLK\*0.125. So this will give a 1ms SysTick interrupt if HCLK is configured to 120MHz.

Table 7-2. Interrupt vector table

Interrupt Number	Vector Number	Non-connectivity devices Interrupt Description	Connectivity devices Interrupt Description	Vector Address
<b>IRQ 0</b>	16	WWDGT interrupt	WWDGT interrupt	0x0000_0040
<b>IRQ 1</b>	17	LVD from EXTI interrupt	LVD from EXTI interrupt	0x0000_0044
<b>IRQ 2</b>	18	Tamper interrupt	Tamper interrupt	0x0000_0048
<b>IRQ 3</b>	19	RTC global interrupt	RTC global interrupt	0x0000_004C
<b>IRQ 4</b>	20	FMC global interrupt	FMC global interrupt	0x0000_0050
<b>IRQ 5</b>	21	RCU and CTC interrupt	RCU and CTC interrupt	0x0000_0054
<b>IRQ 6</b>	22	EXTI Line0 interrupt	EXTI Line0 interrupt	0x0000_0058
<b>IRQ 7</b>	23	EXTI Line1 interrupt	EXTI Line1 interrupt	0x0000_005C
<b>IRQ 8</b>	24	EXTI Line2 interrupt	EXTI Line2 interrupt	0x0000_0060
<b>IRQ 9</b>	25	EXTI Line3 interrupt	EXTI Line3 interrupt	0x0000_0064
<b>IRQ 10</b>	26	EXTI Line4 interrupt	EXTI Line4 interrupt	0x0000_0068
<b>IRQ 11</b>	27	DMA0 channel0 global interrupt	DMA0 channel0 global interrupt	0x0000_006C
<b>IRQ 12</b>	28	DMA0 channel1 global interrupt	DMA0 channel1 global interrupt	0x0000_0070

<b>Interrupt Number</b>	<b>Vector Number</b>	<b>Non-connectivity devices Interrupt Description</b>	<b>Connectivity devices Interrupt Description</b>	<b>Vector Address</b>
<b>IRQ 13</b>	29	DMA0 channel2 global interrupt	DMA0 channel2 global interrupt	0x0000_0074
<b>IRQ 14</b>	30	DMA0 channel3 global interrupt	DMA0 channel3 global interrupt	0x0000_0078
<b>IRQ 15</b>	31	DMA0 channel4 global interrupt	DMA0 channel4 global interrupt	0x0000_007C
<b>IRQ 16</b>	32	DMA0 channel5 global interrupt	DMA0 channel5 global interrupt	0x0000_0080
<b>IRQ 17</b>	33	DMA0 channel6 global interrupt	DMA0 channel6 global interrupt	0x0000_0084
<b>IRQ 18</b>	34	ADC0 and ADC1 global interrupt	ADC0 and ADC1 global interrupt	0x0000_0088
<b>IRQ 19</b>	35	USBD High Priority or CAN0 TX interrupts	CAN0 TX interrupts	0x0000_008C
<b>IRQ 20</b>	36	USBD Low Priority or CAN0 RX0 interrupts	CAN0 RX0 interrupts	0x0000_0090
<b>IRQ 21</b>	37	CAN0 RX1 interrupts	CAN0 RX1 interrupts	0x0000_0094
<b>IRQ 22</b>	38	CAN0 EWMC interrupts	CAN0 EWMC interrupts	0x0000_0098
<b>IRQ 23</b>	39	EXTI line[9:5] interrupts	EXTI line[9:5] interrupts	0x0000_009C
<b>IRQ 24</b>	40	TIMER0 break interrupt and TIMER8 global interrupt	TIMER0 break interrupt and TIMER8 global interrupt	0x0000_00A0
<b>IRQ 25</b>	41	TIMER0 update interrupt and TIMER9 global interrupt	TIMER0 update interrupt and TIMER9 global interrupt	0x0000_00A4
<b>IRQ 26</b>	42	TIMER0 trigger and Channel commutation interrupts and TIMER10 global interrupt	TIMER0 trigger and Channel commutation interrupts and TIMER10 global interrupt	0x0000_00A8
<b>IRQ 27</b>	43	TIMER0 channel capture compare interrupt	TIMER0 channel capture compare interrupt	0x0000_00AC
<b>IRQ 28</b>	44	TIMER1 global interrupt	TIMER1 global interrupt	0x0000_00B0
<b>IRQ 29</b>	45	TIMER2 global interrupt	TIMER2 global interrupt	0x0000_00B4
<b>IRQ 30</b>	46	TIMER3 global interrupt	TIMER3 global interrupt	0x0000_00B8
<b>IRQ 31</b>	47	I2C0 event interrupt	I2C0 event interrupt	0x0000_00BC
<b>IRQ 32</b>	48	I2C0 error interrupt	I2C0 error interrupt	0x0000_00C0
<b>IRQ 33</b>	49	I2C1 event interrupt	I2C1 event interrupt	0x0000_00C4
<b>IRQ 34</b>	50	I2C1 error interrupt	I2C1 error interrupt	0x0000_00C8
<b>IRQ 35</b>	51	SPI0 global interrupt	SPI0 global interrupt	0x0000_00CC
<b>IRQ 36</b>	52	SPI1 global interrupt	SPI1 global interrupt	0x0000_00D0
<b>IRQ 37</b>	53	USART0 global interrupt	USART0 global interrupt	0x0000_00D4
<b>IRQ 38</b>	54	USART1 global interrupt	USART1 global interrupt	0x0000_00D8
<b>IRQ 39</b>	55	USART2 global interrupt	USART2 global interrupt	0x0000_00DC
<b>IRQ 40</b>	56	EXTI line[15:10] interrupts	EXTI line[15:10] interrupts	0x0000_00E0

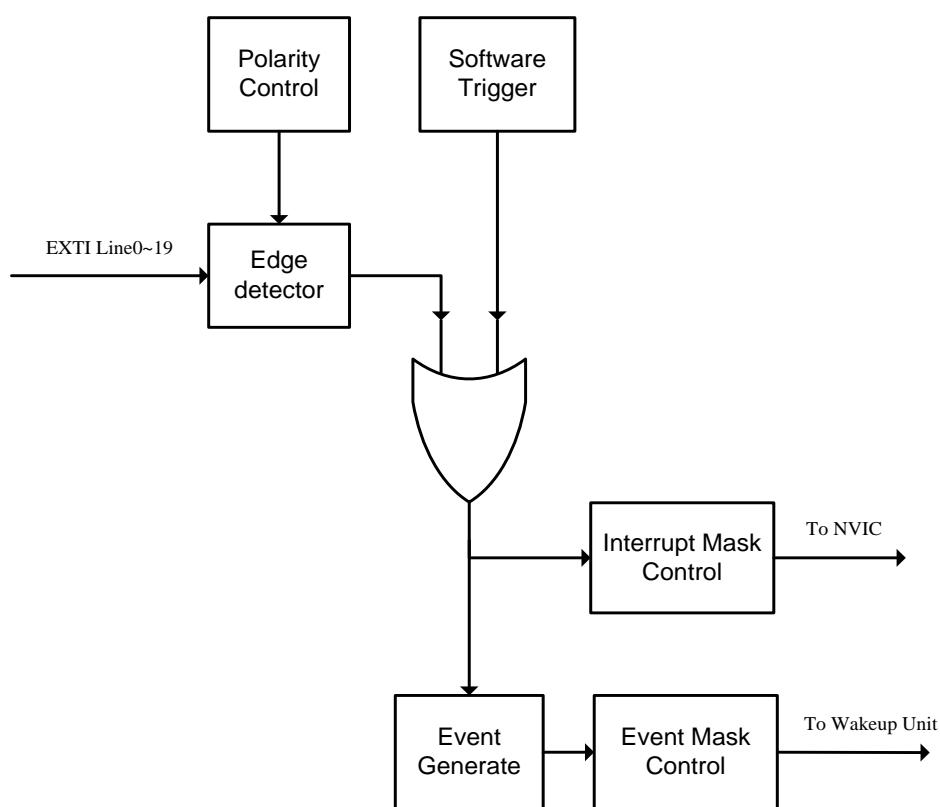
<b>Interrupt Number</b>	<b>Vector Number</b>	<b>Non-connectivity devices Interrupt Description</b>	<b>Connectivity devices Interrupt Description</b>	<b>Vector Address</b>
<b>IRQ 41</b>	57	RTC alarm from EXTI interrupt	RTC alarm from EXTI interrupt	0x0000_00E4
<b>IRQ 42</b>	58	USBD wakeup from EXTI interrupt	USBFS wakeup from EXTI interrupt	0x0000_00E8
<b>IRQ 43</b>	59	TIMER7 break interrupt and TIMER11 global interrupt	TIMER7 break interrupt and TIMER11 global interrupt	0x0000_00EC
<b>IRQ 44</b>	60	TIMER7 update interrupt and TIMER12 global interrupt	TIMER7 update interrupt and TIMER12 global interrupt	0x0000_00F0
<b>IRQ 45</b>	61	TIMER7 trigger and Channel commutation interrupts and TIMER13 global interrupt	TIMER7 trigger and Channel commutation interrupts and TIMER13 global interrupt	0x0000_00F4
<b>IRQ 46</b>	62	TIMER7 channel capture compare interrupt	TIMER7 channel capture compare interrupt	0x0000_00F8
<b>IRQ 47</b>	63	ADC2 global interrupt	reserved	0x0000_00FC
<b>IRQ 48</b>	64	EXMC global interrupt	EXMC global interrupt	0x0000_0100
<b>IRQ 49</b>	65	SDIO global interrupt	reserved	0x0000_0104
<b>IRQ50</b>	66	TIMER4 global interrupt	TIMER4 global interrupt	0x0000_0108
<b>IRQ51</b>	67	SPI2 global interrupt	SPI2 global interrupt	0x0000_010C
<b>IRQ52</b>	68	UART3 global interrupt	UART3 global interrupt	0x0000_0110
<b>IRQ53</b>	69	UART4 global interrupt	UART4 global interrupt	0x0000_0114
<b>IRQ54</b>	70	TIMER5 global interrupt	TIMER5 global interrupt	0x0000_0118
<b>IRQ55</b>	71	TIMER6 global interrupt	TIMER6 global interrupt	0x0000_011C
<b>IRQ56</b>	72	DMA1 channel0 global interrupt	DMA1 channel0 global interrupt	0x0000_0120
<b>IRQ57</b>	73	DMA1 channel1 global interrupt	DMA1 channel1 global interrupt	0x0000_0124
<b>IRQ58</b>	74	DMA1 channel2 global interrupt	DMA1 channel2 global interrupt	0x0000_0128
<b>IRQ59</b>	75	DMA1 channel3 and DMA1 channel4 global interrupt	DMA1 channel3 global interrupt	0x0000_012C
<b>IRQ60</b>	76	reserved	DMA1 channel4 global interrupt	0x0000_0130
<b>IRQ61</b>	77	reserved	ENET global interrupt	0x0000_0134
<b>IRQ62</b>	78	reserved	ENET wakeup from EXTI interrupt	0x0000_0138
<b>IRQ63</b>	79	reserved	CAN1 TX interrupt	0x0000_013C
<b>IRQ64</b>	80	reserved	CAN1 RX0 interrupt	0x0000_0140
<b>IRQ65</b>	81	reserved	CAN1 RX1 interrupt	0x0000_0144
<b>IRQ66</b>	82	reserved	CAN1 EWMC interrupt	0x0000_0148
<b>IRQ67</b>	83	reserved	USBFS global interrupt	0x0000_014C

Note:

1. IRQ0 ~ 59 are available in HD and XD devices, but the TIMER8 to TIMER13 global interrupts(IRQ24、IRQ25、IRQ26、IRQ43、IRQ44、IRQ45)are available only in the XD devices.
2. At non-connectivity devices, USB and CAN (IRQ19, IRQ 20) function cannot be used at the same time

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

**Figure 7-1. Block diagram of EXTI**



## 7.5. External Interrupt and Event function overview

The EXTI contains up to 20 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 4 lines from internal modules (including LVD, RTC Alarm, USB Wakeup, Ethernet Wakeup). All GPIO pins can be selected as an EXTI trigger source by configuring AFIO\_EXTISSx 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 Cortex-M4 processor fully implements the Wait For Interrupt (WFI), Wait For Event (WFE) and the Send Event (SEV) instructions. The Wake-up Interrupt Controller (WIC) enables the processor and NVIC to be put into a very low-power sleep mode leaving the WIC to identify and prioritize interrupts and events. EXTI can be used to wake up processor and the whole system when some expected event occurs, such as a special GPIO pin toggling or RTC alarm.

**Table 7-3. EXTI source**

EXTI Line Number	Source
0	PA0/PB0/PC0/PD0/PE0/PF0/PG0
1	PA1/PB1/PC1/PD1/PE1/PF1/PG1
2	PA2/PB2/PC2/PD2/PE2/PF2/PG2
3	PA3/PB3/PC3/PD3/PE3/PF3/PG3
4	PA4/PB4/PC4/PD4/PE4/PF4/PG4
5	PA5/PB5/PC5/PD5/PE5/PF5/PG5
6	PA6/PB6/PC6/PD6/PE6/PF6/PG6
7	PA7/PB7/PC7/PD7/PE7/PF7/PG7
8	PA8/PB8/PC8/PD8/PE8/PF8/PG8
9	PA9/PB9/PC9/PD9/PE9/PF9/PG9
10	PA10/PB10/PC10/PD10/PE10/PF10/PG10
11	PA11/PB11/PC11/PD11/PE11/PF11/PG11
12	PA12/PB12/PC12/PD12/PE12/PF12/PG12
13	PA13/PB13/PC13/PD13/PE13/PF13/PG13
14	PA14/PB14/PC14/PD14/PE14/PF14/PG14
15	PA15/PB15/PC15/PD15/PE15/PF15/PG15
16	LVD
17	RTC Alarm
18	USB Wakeup
19	Ethernet Wakeup

## 7.6. EXTI Register

EXTI base address: 0x4001 0400

### 7.6.1. 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														INTEN19	INTEN18	INTEN17	INTEN16
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
31:20	Reserved	Must be kept at reset value.
19: 0	INTEN $x$	Interrupt enablebit 0: Interrupt from Linex is disabled. 1: Interrupt from Linex is enabled.

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

Address offset: 0x04

Reset value: 0x0000 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
Reserved														EVEN19	EVEN18	EVEN17	EVEN16
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
31:20	Reserved	Must be kept at reset value.
19: 0	EVEN $x$	Event enable bit 0: Event from Linex is disabled. 1: Event from Linex is enabled.

### 7.6.3. 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														RTEN19	RTEN18	RTEN17	RTEN16
														rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
RTEN15	RTEN14	RTEN13	RTEN12	RTEN11	RTEN10	RTEN9	RTEN8	RTEN7	RTEN6	RTEN5	RTEN4	RTEN3	RTEN2	RTEN1	RTENO		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bits	Fields	Descriptions
31:20	Reserved	Must be kept at reset value.
19:0	RTENx	Rising edge trigger enable 0: Rising edge of Linex is invalid 1: Rising edge of Linex is valid as an interrupt/event request

### 7.6.4. 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														FTEN19	FTEN18	FTEN17	FTEN16
														rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
FTEN15	FTEN14	FTEN13	FTEN12	FTEN11	FTEN10	FTEN9	FTEN8	FTEN7	FTEN6	FTEN5	FTEN4	FTEN3	FTEN2	FTEN1	FTENO		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bits	Fields	Descriptions
31: 20	Reserved	Must be kept at reset value.
19: 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

### 7.6.5. 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														SWIEV19	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		

Bits	Fields	Descriptions
31:20	Reserved	Must be kept at reset value.
19: 0	SWIEV $x$	Interrupt/Event software trigger 0: Deactivate the EXTI $x$ software interrupt/event request 1: Activate the EXTI $x$ software interrupt/event request

### 7.6.6. 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														PD19	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		

Bits	Fields	Descriptions
31: 20	Reserved	Must be kept at reset value.
19: 0	PD $x$	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.

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

### 8.1. Overview

There are up to 112 general purpose I/O pins (GPIO), named PA0 ~ PA15, PB0 ~ PB15, PC0 ~ PC15, PD0 ~ PD15, PE0 ~ PE15, PF0 ~ PF15 and PG0 ~ PG15 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 (EXIT).

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 floating. All GPIOs are high-current capable except for analog mode.

### 8.2. 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.

### 8.3. Function overview

Each of the general-purpose I/O ports can be configured as 8 modes, including 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 (GPIO<sub>x</sub>\_CTL0/GPIO<sub>x</sub>\_CTL1), and a 32-bits register (GPIO<sub>x</sub>\_OCTL). [Table 8-1. GPIO configuration table](#) shows the details.

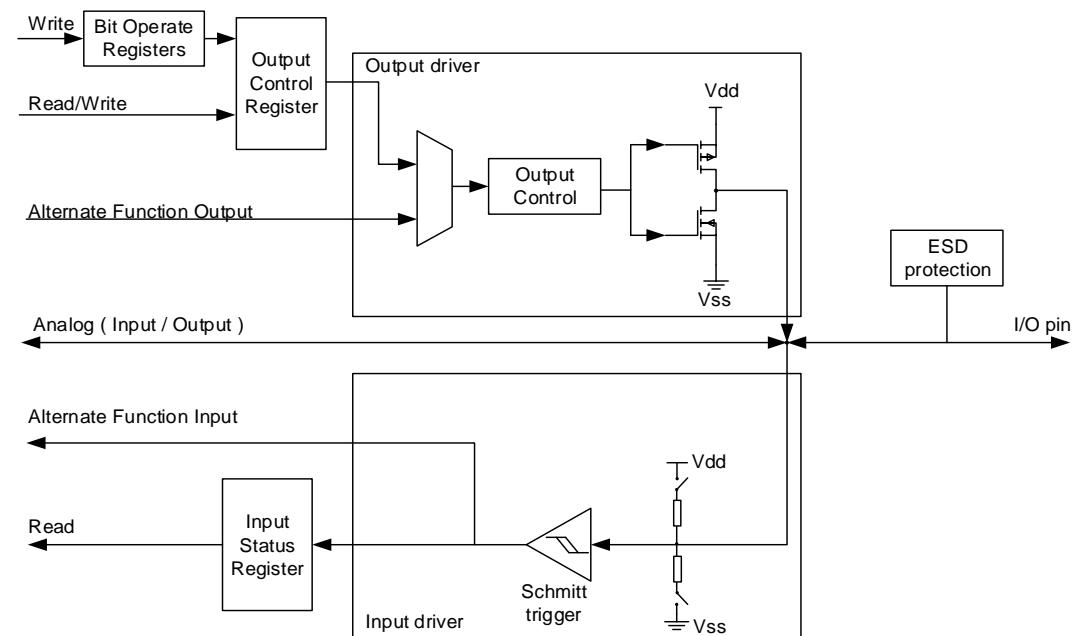
Table 8-1. GPIO configuration table

Configuration mode		CTL[1:0]	SPDy: MD[1:0]	OCTL
Input	Analog	00	x 00	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	x 00: Reserved	0 or 1
	Open-drain	01	x 01: Speed up to 10MHz x 10: Speed up to 2MHz	0 or 1
Alternate Function Output (AFIO)	Push-pull	10	0 11: Speed up to 50MHz 1 11: Speed up to 120MHz <sup>(1)</sup> (SPDy required to be set to 0b1)	don't care
	Open-drain	11		don't care

- When the port output speed is more than 50 MHz, the user should enable the I/O compensation cell. Refer to IO compensation control register (AFIO\_CPSCTL).

[\*\*Figure 8-1. Basic structure of a standard I/O port bit\*\*](#) shows the basic structure of an I/O port bit.

**Figure 8-1. Basic structure of a standard I/O port bit**



### 8.3.1. GPIO pin configuration

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

PA15: JTDI in PU mode.

PA14: JTCK / SWCLK in PD mode.

PA13: JTMS / SWDIO 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, the user can configure the speed of the ports and choose 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, the 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.

### 8.3.2. External interrupt/event lines

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

### 8.3.3. Alternate functions (AF)

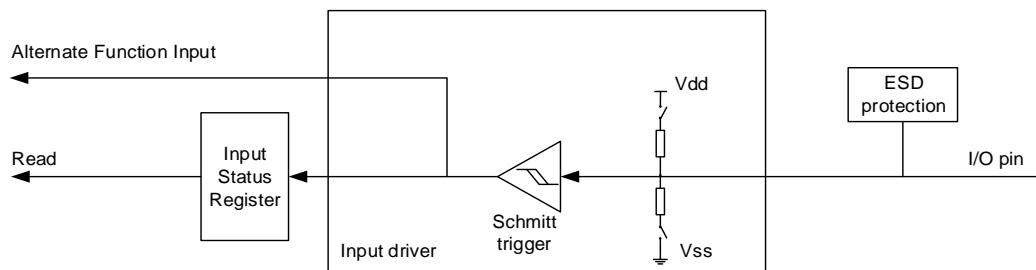
When the port is configured as AFIO (set CTLy bits to “0b10” or “0b11”, and set MDy bits to “0b01”, “0b10”, or “0b11”, which is in GPIOx\_CTL0/GPIOx\_CTL1 registers), the port is used as peripheral alternate functions. The detail alternate function assignments for each port are in the device datasheet.

### 8.3.4. 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 8-2. Input configuration](#) shows the input configuration of the GPIO pin.

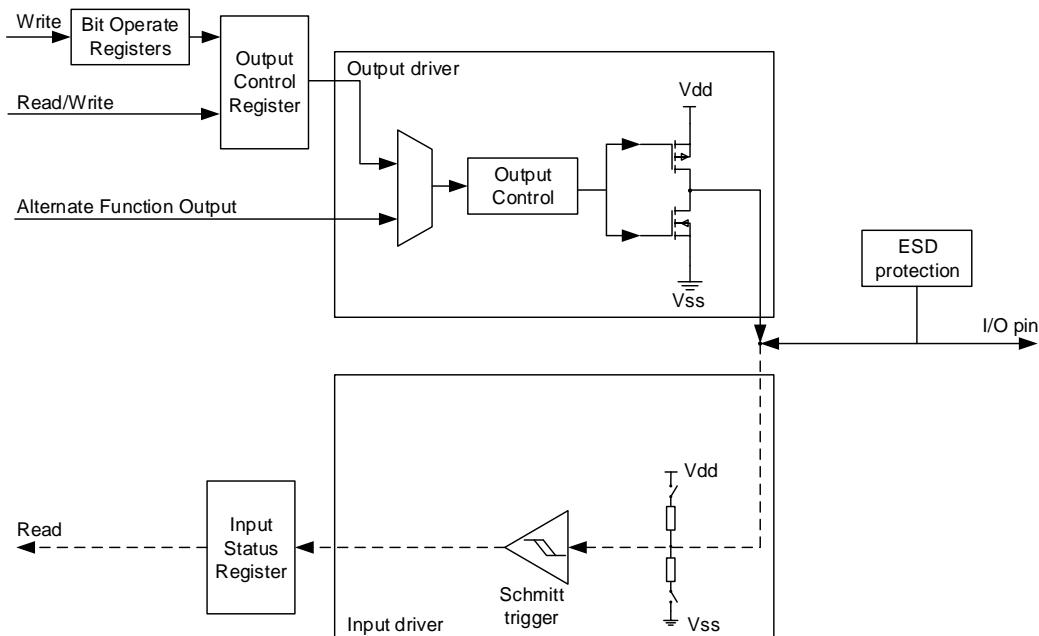
**Figure 8-2. Input configuration**


### 8.3.5. 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 outputs low level when setting “0” in the output control register. while the pad holds Hi-Z when setting “1” in the output control register.
- Push-Pull Mode, the pad outputs low level when setting “0” in the output control register; while the pad output high level when setting “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 8-3. Output configuration](#) shows the output configuration.

**Figure 8-3. Output configuration**


### 8.3.6. 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 8-4. Analog configuration\*\*](#) shows the analog configuration.

**Figure 8-4. Analog configuration**



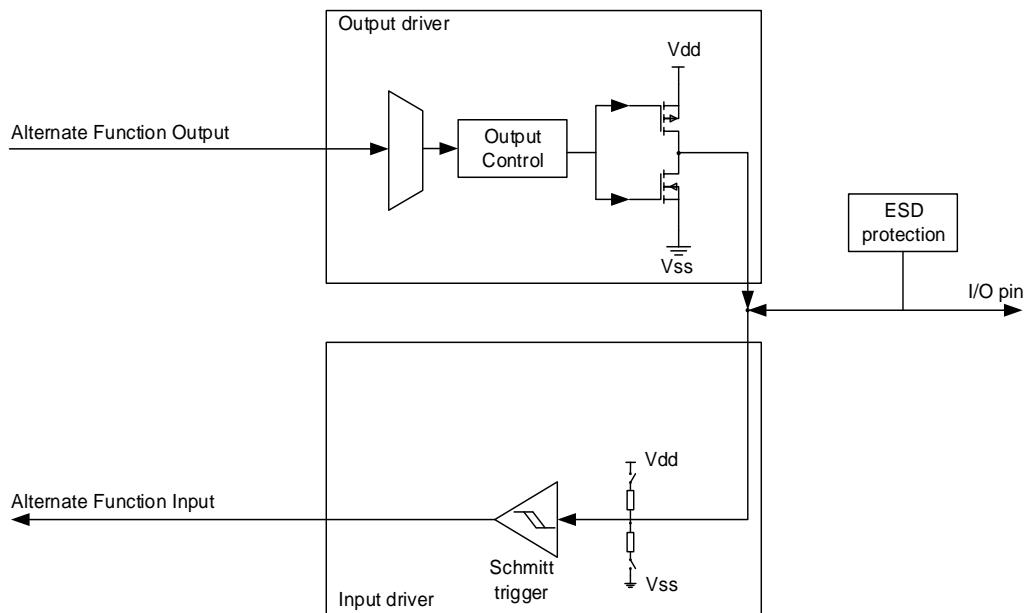
### 8.3.7. Alternate function (AF) configuration

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 8-5. Alternate function configuration\*\*](#) shows the alternate function configuration of the GPIO pin.

**Figure 8-5. Alternate function configuration**


### 8.3.8. 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 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 recommended to be used in the configuration of driving a power module.

### 8.3.9. GPIO I/O compensation cell

If the I/O port output speed need more than 50MHz, it is recommended to use the compensation cell for slew rate control to reduce the I/O noise effects on the power supply.

Compensation cell is disabled after reset, it needs to be enabled by the user. After enabling the compensation cell, the complete flag CPS\_RDY is set to indicate that the compensation cell is ready and can be used. If the supply voltage over 2.4 V~3.6V, must disable the compensation cell.

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

### 8.4.1. Introduction

In order to expand the flexibility of the GPIO or the usage of peripheral functions, each I/O pin can be configured 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 source by setting the relevant EXTI source selection register (AFIO\_EXTI\_Sx) to trigger an interrupt or event.

### 8.4.2. Main features

- EXTI source selection.
- Each pin has up to four alternative functions for configuration.

### 8.4.3. JTAG/SWD alternate function remapping

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

**Table 8-2. Debug interface signals**

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

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

**Table 8-3. Debug port mapping and Pin availability**

SWJ_CFG [2:0]	JTAG-DP and SW-DP	Pin availability				
		PA13	PA14	PA15	PB3	PB4
000	JTAG-DP Enabled and SW-DP Enabled (Reset state)	X	X	X	X	X
001	JTAG-DP Enabled and SW-DP Enabled but without NJTRST	X	X	X	X	✓
010	JTAG-DP Disabled and SW-DP Enabled	X	X	✓	✓	✓
100	JTAG-DP Disabled and SW-DP Disabled	✓	✓	✓	✓	✓
Other	Forbidden					

1. Can't released if using asynchronous trace.
2. “✓” Indicates that the corresponding pin can be used as a general-purpose I/O.
3. “X” Indicates that the corresponding pin can't be used as a general-purpose I/O.

#### 8.4.4. ADC AF remapping

Refer to AFIO Port Configuration Register 0 (AFIO\_PCF0).

**Table 8-4. ADC0/ADC1 external trigger routine conversion AF remapping<sup>(1)</sup>**

Register	ADC0	ADC1
ADC0_ETRGRER_REMA P = 0	ADC0 external signal trigger normal conversion is connected to EXTI11	-
ADC0_ETRGRER_REMA P = 1	ADC0 external signal trigger routine conversion is connected to TIMER7_TRGO	-
ADC1_ETRGRER_REMA P = 0	-	ADC1 external signal trigger routine conversion is connected to EXTI11
ADC1_ETRGRER_REMA P = 1	-	ADC1 external signal trigger routine conversion is connected to TIMER7_TRGO

1. Remap available only for High-density and Extra-density devices

#### 8.4.5. TIMER AF remapping

**Table 8-5. TIMERx alternate function remapping**

Alternate function	TIMERx_REMAP [1:0](x = 0, 1, 2)			
	TIMERx_REMAP(x = 8, 9, 10, 12, 13)		-	
	“0” /“00” (no remap)	“1” /“01” (partial remap)	“10” (partial remap)	“11” (full remap)
TIMER0_ETI	PA12		-	PE7
TIMER0_CH0	PA8		-	PE9
TIMER0_CH1	PA9		-	PE11
TIMER0_CH2	PA10		-	PE13
TIMER0_CH3	PA11		-	PE14
TIMER0_BKIN	PB12	PA6	-	PE15
TIMER0_CH0_ ON	PB13	PA7	-	PE8
TIMER0_CH1_ ON	PB14	PB0	-	PE10
TIMER0_CH2_	PB15	PB1	-	PE12

Alternate function	TIMERx_REMAP [1:0](x = 0, 1, 2)			
	TIMERx_REMAP(x = 8, 9, 10, 12, 13)		-	
	“0” /“00” (no remap)	“1” /“01” (partial remap)	“10” (partial remap)	“11” (full remap)
ON				
TIMER1_CH0/TIMER1_ETI	PA0	PA15	PA0	PA15
TIMER1_CH1	PA1	PB3	PA1	PB3
TIMER1_CH2	PA2		PB10	
TIMER1_CH3	PA3		PB11	
TIMER2_CH0	PA6	-	PB4	PC6
TIMER2_CH1	PA7	-	PB5	PC7
TIMER2_CH2	PB0	-	PB0	PC8
TIMER2_CH3	PB1	-	PB1	PC9
TIMER3_CH0	PB6	PD12	-	-
TIMER3_CH1	PB7	PD13	-	-
TIMER3_CH2	PB8	PD14	-	-
TIMER3_CH3	PB9	PD15	-	-
TIMER8_CH0	PA2	PE5	-	-
TIMER8_CH1	PA3	PE6	-	-
TIMER9_CH0	PB8	PF6	-	-
TIMER10_CH0	PB9	PF7	-	-
TIMER12_CH0	PA6	PF8	-	-
TIMER13_CH0	PA7	PF9	-	-

1. TIMER0 remap available only for 100-pin and 144-pin packages
2. TIMER1\_CH0 and TIMER1\_ETI share the same pin but cannot be used at the same time
3. TIMER2 remap available only for 64-pin, 100-pin and 144-pin packages.
4. TIMER3 remap available only for 100-pin and 144-pin packages.
5. TIMER8/9/10/12/13 refer to the AF remap and debug I/O configuration register 1(AFIO\_PCF1).

**Table 8-6. TIMER4 alternate function remapping<sup>(1)</sup>**

Alternate function	TIMER4CH3_IREMAP = 0	TIMER4CH3_IREMAP = 1
TIMER4_CH3	TIMER4_CH3 is connected to PA3	IRC40K internal clock is connected to TIMER4_CH3 input for calibration purpose

1. Remap available only for High-density and Extra-density and Connectivity lines devices.

### 8.4.6. USART AF remapping

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

**Table 8-7. USART alternate function remapping**

Register	USART0	USART1	USART2
USART0_REMAP = 0	PA9(USART0_TX) PA10(USART0_RX)	-	-
USART0_REMAP = 1	PB6(USART0_TX) PB7(USART0_RX)	-	-
USART1_REMAP = 0	-	PA0(USART1_CTS) PA1(USART1_RTS) PA2(USART1_TX) PA3(USART1_RX) PA4(USART1_CK)	-
USART1_REMAP = 1 <sup>(1)</sup>	-	PD3(USART1_CTS) PD4(USART1_RTS) PD5(USART1_TX) PD6(USART1_RX) PD7(USART1_CK)	-
USART2_REMAP[1:0] = "00" (no remap)	-	-	PB10(USART2_TX) PB11(USART2_RX) PB12(USART2_CK) PB13(USART2_CTS) PB14(USART2_RTS)
USART2_REMAP [1:0] = "01" (partial remap) <sup>(2)</sup>	-	-	PC10(USART2_TX) PC11(USART2_RX) PC12(USART2_CK) PB13(USART2_CTS) PB14(USART2_RTS)
USART2_REMAP [1:0] = "11" (full remap) <sup>(3)</sup>	-	-	PD8(USART2_TX) PD9(USART2_RX) PD10(USART2_CK) PD11(USART2_CTS) PD12(USART2_RTS)

1. Remap available only 100-pin and 144-pin packages
2. Remap available only for 64-pin, 100-pin and 144-pin packages
3. Remap available only 100-pin and 144-pin packages

### 8.4.7. I2C0 AF remapping

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

**Table 8-8. I2C0 alternate function remapping**

Register	I2C0_SCL	I2C0_SDA
I2C0_REMAP = 0	PB6	PB7
I2C0_REMAP = 1	PB8	PB9

#### 8.4.8. SPI/I2S AF remapping

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

**Table 8-9. SPI/I2S alternate function remapping**

Register	SPI0	SPI2/I2S
SPI0_REMAP = 1	PA4(SPI0_NSS) PA5(SPI0_SCK) PA6(SPI0_MISO) PA7(SPI0_MOSI) PA2(SPI0_IO2) PA3(SPI0_IO3)	-
SPI0_REMAP = 1	PA15(SPI0_NSS) PB3(SPI0_SCK) PB4(SPI0_MISO) PB5(SPI0_MOSI) PB6(SPI0_IO2) PB7(SPI0_IO3)	-
SPI2_REMAP = 0	-	PA15(SPI2_NSS/ I2S2_WS) PB3(SPI2_SCK/ I2S2_CK) PB4(SPI2_MISO) PB5(SPI2_MOSI/I2S2_SD)
SPI2_REMAP = 1	-	PA4(SPI2_NSS/ I2S2_WS) PC10(SPI2_SCK/ I2S2_CK) PC11(SPI2_MISO) PC12(SPI2_MOSI/I2S2_SD)

#### 8.4.9. CAN AF remapping

The CAN0 signals can be mapped on Port A, Port B or Port D as shown in table below. For port D, remapping is not possible in devices delivered in 64-pin packages.

**Table 8-10. CAN alternate function remapping**

Register <sup>(1)</sup>	CAN0	CAN1
CAN0_REMAP[1:0] =“00”	PA11(CAN0_RX) PA12(CAN0_TX)	-
CAN0_REMAP[1:0] =“10”	PB8(CAN0_RX) PB9(CAN0_TX)	-
CAN0_REMAP[1:0] =“11” <sup>(2)</sup>	PD0(CAN0_RX) PD1(CAN0_TX)	-

<b>Register<sup>(1)</sup></b>	<b>CAN0</b>	<b>CAN1</b>
CAN1_REMAP = "0"	-	PB12(CAN1_RX) PB13(CAN1_TX)
CAN1_REMAP = "1"	-	PB5(CAN1_RX) PB6(CAN1_TX)

1. CAN0\_RX and CAN0\_TX in connectivity line devices; CAN\_RX and CAN\_TX in other devices with a single CAN interface.
2. This remapping is available only on 100-pin packages.

#### 8.4.10. Ethernet AF remapping

**Table 8-11. ENET alternate function remapping**

<b>Register</b>	<b>ENET</b>
ENET_REMAP = "0"	PA7(RX_DV-CRS_DV) PC4(RXD0) PC5(RXD1) PB0(RXD2) PB1(RXD3)
ENET_REMAP = "1"	PD8(RX_DV-CRS_DV) PD9(RXD0) PD10(RXD1) PD11(RXD2) PD12(RXD3)

#### 8.4.11. CTC AF remapping

Refer to AFIO port configuration register 1 (AFIO\_PCF1).

**Table 8-12. CTC alternate function remapping**

<b>Register</b>	<b>CTC_SYNC</b>
CTC_REMAP [1:0] = "00"	PA8
CTC_REMAP [1:0] = "01"	PD15
CTC_REMAP [1:0] = "10" or "11"	PF0

#### 8.4.12. 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 VBAT (VDD 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 [3.3.1](#).

**Table 8-13. 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 8-14. OSC pins configuration**

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

## 8.5. Register definition

GPIOA base address: 0x4001 0800

GPIOB base address: 0x4001 0C00

GPIOC base address: 0x4001 1000

GPIOD base address: 0x4001 1400

GPIOE base address: 0x4001 1800

GPIOF base address: 0x4001 1C00

GPIOG base address: 0x4001 2000

AFIO base address: 0x4001 0000

### 8.5.1. Port control register 0 (GPIOx\_CTL0, x=A..G)

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	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	rw	rw	rw	rw	rw	rw	rw	rw

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

21:20	MD5[1:0]	Pin 5 mode bits These bits are set and cleared by software refer to MD0[1:0]description
19:18	CTL4[1:0]	Pin 4 configuration bits These bits are set and cleared by software refer to CTL0[1:0]description
17:16	MD4[1:0]	Pin 4 mode bits These bits are set and cleared by software refer to MD0[1:0]description
15:14	CTL3[1:0]	Pin 3 configuration bits These bits are set and cleared by software refer to CTL0[1:0]description
13:12	MD3[1:0]	Pin 3 mode bits These bits are set and cleared by software refer to MD0[1:0]description
11:10	CTL2[1:0]	Pin 2 configuration bits These bits are set and cleared by software refer to CTL0[1:0]description
9:8	MD2[1:0]	Pin 2 mode bits These bits are set and cleared by software refer to MD0[1:0]description
7:6	CTL1[1:0]	Pin 1 configuration bits These bits are set and cleared by software refer to CTL0[1:0]description
5:4	MD1[1:0]	Pin 1 mode bits These bits are set and cleared by software refer to MD0[1:0]description
3:2	CTL0[1:0]	Pin 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]	Pin 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
-----	----------	---

### 8.5.2. Port control register 1 (GPIOx\_CTL1, x=A..G)

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	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	rw	rw	rw	rw	rw	rw	rw	rw

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

19:18	CTL12[1:0]	Pin 12 configuration bits These bits are set and cleared by software refer to CTL0[1:0]description
17:16	MD12[1:0]	Pin 12 mode bits These bits are set and cleared by software refer to MD0[1:0]description
15:14	CTL11[1:0]	Pin 11 configuration bits These bits are set and cleared by software refer to CTL0[1:0]description
13:12	MD11[1:0]	Pin 11 mode bits These bits are set and cleared by software refer to MD0[1:0]description
11:10	CTL10[1:0]	Pin 10 configuration bits These bits are set and cleared by software refer to CTL0[1:0]description
9:8	MD10[1:0]	Pin 10 mode bits These bits are set and cleared by software refer to MD0[1:0]description
7:6	CTL9[1:0]	Pin 9 configuration bits These bits are set and cleared by software refer to CTL0[1:0]description
5:4	MD9[1:0]	Pin 9 mode bits These bits are set and cleared by software refer to MD0[1:0]description
3:2	CTL8[1:0]	Pin 8 configuration bits These bits are set and cleared by software refer to CTL0[1:0]description
1:0	MD8[1:0]	Pin 8 mode bits These bits are set and cleared by software refer to MD0[1:0]description

### 8.5.3. Port input status register (GPIOx\_ISTAT, x=A..G)

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	ISTAT 9	ISTAT 8	ISTAT 7	ISTAT 6	ISTAT 5	ISTAT 4	ISTAT 3	ISTAT 2	ISTAT 1	ISTAT 0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

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

#### 8.5.4. Port output control register (GPIOx\_OCTL, x=A..G)

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

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

#### 8.5.5. Port bit operate register (GPIOx\_BOP, x=A..G)

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
CR15	CR14	CR13	CR12	CR11	CR10	CR9	CR8	CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

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

BOP15	BOP14	BOP13	BOP12	BOP11	BOP10	BOP9	BOP8	BOP7	BOP6	BOP5	BOP4	BOP3	BOP2	BOP1	BOP0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits	Fields	Descriptions
31:16	CRy	Pin 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	Pin 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

### 8.5.6. Port bit clear register (GPIOx\_BC, x=A..G)

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

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	CRy	Pin 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

### 8.5.7. Port configuration lock register (GPIOx\_LOCK, x=A..G)

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															

rw

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 it is always readable.</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><b>Note:</b> The value of LK[15:0] must hold during the LOCK Key Writing sequence.</p>
15:0	LKy	<p>Pin 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>

### 8.5.8. Port bit speed register (GPIOx\_SPD, x=A..G)

Address offset: 0x3C

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															
SPD15	SPD14	SPD13	SPD12	SPD11	SPD10	SPD9	SPD8	SPD7	SPD6	SPD5	SPD4	SPD3	SPD2	SPD1	SPD0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

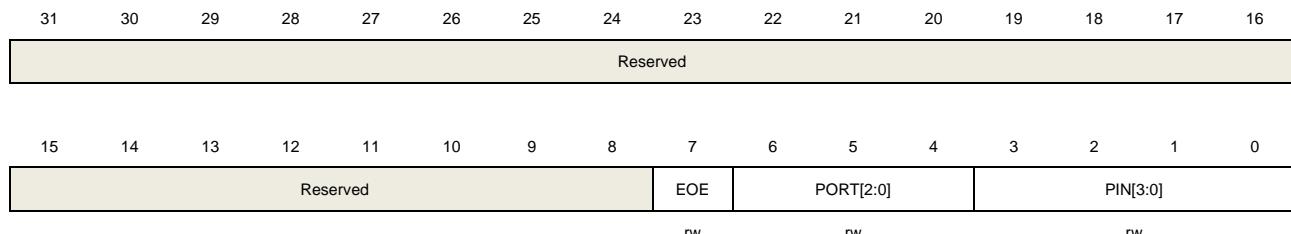
Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	SPDy	<p>Set very high output speed(120MHz) when MDx is 0b11.</p> <p>If the Pin output speed is more than 50MHz, set this bit to 1 and set MDx to 0b11. These bits are set and cleared by software.</p> <p>0: No effect</p> <p>1: Max speed more than 50MHz. (MDx required to be set to 0b11 together).</p> <p><b>Note:</b> When the pin output speed is more than 50 MHz, the user should enable the I/O compensation cell. Refer to CPS_EN bit in AFIO_CPSCTL register.</p>

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

Address offset: 0x00

Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value.
7	EOE	Event output enable Set and cleared by software. When this bit is set, the Cortex EVENTOUT output is connected to the I/O selected by the PORT[2:0] and PIN[3:0] bits.
6:4	PORT[2:0]	Event output port selection Set and cleared by software. Select the port to output the Cortex EVENTOUT signal. 000: Select PORT A 001: Select PORT B 010: Select PORT C 011: Select PORT D 100: Select PORT E
3:0	PIN[3:0]	Event output pin selection Set and cleared by software. Select the pin to output the Cortex EVENTOUT signal. 0000: Select Pin 0 0001: Select Pin 1 0010: Select Pin 2 ... 1111: Select Pin 15

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

Address offset: 0x04

Reset value: 0x0000 0000

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

Memory map and bit definitions for High-density and Extra-density devices:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Reserved	SPI2_ REMAP	Reserved	SWJ_CFG[2:0]	Reserved	ADC1_ ETRGRE R _REMAP	Reserved	ADC0_ ETRGRE R _REMAP	Reserved	H3_ IREMAP							
	rw		w							rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
PD01_ REMAP	CAN_REMAP [1:0]	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	rw	rw	rw	rw	

Bits	Fields	Descriptions
31:29	Reserved	Must be kept at reset value.
28	SPI2_REMAP	<p>SPI2/I2S2 remapping</p> <p>This bit is set and cleared by software.</p> <p>0: No remap (SPI2_NSS-I2S2_WS/PA15, SPI2_SCK-I2S2_CK/PB3, SPI2_MISO/PB4, SPI2_MOSI-I2S_SD/PB5)</p> <p>1: Full remap (SPI2_NSS-I2S2_WS/PA4, SPI2_SCK-I2S2_CK/PC10, SPI2_MISO/PC11, SPI2_MOSI-I2S_SD/PC12)</p>
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 and trace alternate function I/Os. The SWJ(Serial Wire JTAG) supports JTAG or SWD access to the Cortex debug port. The default state after reset is SWJ ON without trace.This allows JTAG or SW mode to be enabled by sending a specific sequence on the JTMS/JTCK pin.</p> <p>000: Full SWJ (JTAG-DP + SW-DP): reset state</p> <p>001: Full SWJ (JTAG-DP + SW-DP): but without NJTRST</p> <p>010: JTAG-DP Disabled and SW-DP Enabled</p> <p>100: JTAG-DP Disabled and SW-DP Disabled</p> <p>Other: no effect</p>
23:21	Reserved	Must be kept at reset value.
20	ADC1_ETRGREG_ REMAP	<p>ADC 1 external trigger routine conversion remapping</p> <p>Set and cleared by software. The bit controls the trigger input be connected to ADC1 external trigger routine conversion or not. When this bit is reset, the ADC1 external trigger routine conversion to EXTI11. When this bit is set, the ADC1 external event routine conversion is connected to TIMER7_TRGO.</p>
19	Reserved	Must be kept at reset value.
18	ADC0_ETRGREG_ REMAP	<p>ADC 0 external trigger routine conversion remapping</p> <p>Set and cleared by software.</p>

17	Reserved	Must be kept at reset value.
16	TIMER4CH3_IREMAP	<p>TIMER4 channel3 internal remapping</p> <p>Set and cleared by software. This bit controls the TIMER4_CH3 internal mapping.</p> <p>When reset, 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> <p>0: No remap</p> <p>1: Remap</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	CAN_REMAP [1:0]	<p>CAN interface remapping</p> <p>These bits are set and cleared by software.</p> <p>00: No remap (CAN_RX/PA11, CAN_TX/PA12)</p> <p>01: Not used</p> <p>10: Partial remap (CAN_RX/PB8, CAN_TX/PB9)</p> <p>11: Full remap (CAN_RX/PD0, CAN_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_REMAP[1:0]	<p>TIMER2 remapping</p> <p>These bits are set and cleared by software.</p> <p>00: No remap (TIMER2_CH0/PA6, TIMER2_CH1/PA7, TIMER2_CH2/PB0, TIMER2_CH3/PB1)</p> <p>01: Not used</p> <p>10: Partial remap (TIMER2_CH0/PB4, TIMER2_CH1/PB5, TIMER2_CH2/PB0, TIMER2_CH3/PB1)</p> <p>11: Full remap (TIMER2_CH0/PC6, TIMER2_CH1/PC7, TIMER2_CH2/PC8, TIMER2_CH3/PC9)</p>
9:8	TIMER1_REMAP [1:0]	<p>TIMER1 remapping</p> <p>These bits are set and cleared by software.</p> <p>00: No remap (TIMER1_CH0-TIMER1_ETI/PA0, TIMER1_CH1/PA1, TIMER1_CH2/PA2, TIMER1_CH3/PA3)</p> <p>01: Partial remap (TIMER1_CH0-TIMER1_ETI/PA15, TIMER1_CH1/PB3, TIMER1_CH2/PA2, TIMER1_CH3/PA3)</p> <p>10: Partial remap (TIMER1_CH0-TIMER1_ETI/PA0, TIMER1_CH1/PA1, TIMER1_CH2/PB10, TIMER1_CH3/PB11)</p>

		11: Full remap(TIMER1_CH0-TIMER1_ETI/PA15, TIMER1_CH1/PB3, TIMER1_CH2/PB10, TIMER1_CH3/PB11)
7:6	TIMER0_REMAP	TIMER0 remapping
	[1:0]	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	USART2_REMAP	USART2 remapping
	[1:0]	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	USART1_REMAP	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, SPI0_IO2 /PA2, SPI0_IO3 /PA3)

1: Remap (SPI0\_NSS/PA15, SPI0\_SCK/PB3, SPI0\_MISO/PB4, SPI0\_MOSI/PB5, SPI0\_IO2/PB6, SPI0\_IO3/PB7)

#### Memory map and bit definitions for connectivity devices:

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	PTP_ REMAP	PPS_ REMAP	TIMER1_ REMAP	SPI2_ REMAP	Reserved	SWJ_CFG[2:0]			ENET_ _PHY _SEL	CAN1_ REMAP	ENET_ REMAP	Reserved			TIMER4 CH3_ IREMAP	
	rw	rw	rw		w				rw	rw	rw					rw
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PD01_ REMAP	CANO_REMAP [1:0]	TIMER3_ REMAP	TIMER2_ REMAP[1:0]	TIMER1_ REMAP[1:0]	TIMER0_ REMA [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	rw

Bits	Fields	Descriptions
31	Reserved	Must be kept at reset value.
30	PTP_PPS_REMAP	<p>Ethernet PTP PPS remapping</p> <p>This bit is set and cleared by software. It enables the Ethernet MAC_PPS to be output on the PB5 pin</p> <p>0: PPT_PPS not output PB5 pin</p> <p>1: PPT_PPS is output on PB5 pin</p> <p>Note: This bit is available only in connectivity line devices and is reserved otherwise.</p>
29	TIMER1ITI1_REMAP	<p>TIMER1 internal trigger 1 remapping</p> <p>These bits are set and cleared by software. It control the TMER1_ITI1 internal mapping</p> <p>0: Connect TIMER1_ITI1 internally to the Ethernet PTP output for calibration purposes</p> <p>1: Connect USB OTG SOF (Start of Frame) output TIMER1_ITI1 for calibration purposes</p> <p>Note: This bit is available only in connectivity line devices and is reserved otherwise.</p>
28	SPI2_REMAP	<p>SPI2/I2S2 remapping</p> <p>This bit is set and cleared by software.</p> <p>0: No remap (SPI2_NSS-I2S2_WS/PA15, SPI2_SCK-I2S2_CK/PB3, SPI2_MISO/PB4, SPI2_MOSI-I2S_SD/PB5)</p> <p>1: Full remap (SPI2_NSS-I2S2_WS/PA4, SPI2_SCK-I2S2_CK/PC10, SPI2_MISO/PC11, SPI2_MOSI-I2S_SD/PC12)</p>
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 and trace alternate function I/Os. The SWJ(Serial Wire JTAG) supports JTAG or SWD access to the Cortex debug port. The default state after</p>

		reset is SWJ ON without trace. This allows JTAG or SW mode to be enabled by sending a specific sequence on the JTMS/JTCK pin
		000: Full SWJ(JTAG-DP +SW-DP): Reset State
		001: Full SWJ(JTAG-DP +SW-DP): but without NJTRST
		010: JTAG-DP Disabled and SW-DP Enabled
		100: JTAG-DP Disabled and SW-DP Disabled
		Other combinations: no effect
23	ENET_PHY_SEL	<p>Ethernet MII or RMII PHY selection</p> <p>This bit is set and cleared by software. It configures the Ethernet MAC internally for use with an external MII or RMII PHY.</p> <p>0: Configure Ethernet MAC for connection with an MII PHY</p> <p>1: Configure Ethernet MAC for connection with an RMII PHY</p> <p>Note: This bit is available only in connectivity line devices and is reserved otherwise.</p>
22	CAN1_REMAP	<p>CAN1 I/O remapping</p> <p>This bit is set and cleared by software. It controls the CAN1_TX and CAN1_RX pins</p> <p>0: No remap (CAN1_RX/PB12, CAN_TX/PB13)</p> <p>1: Remap (CAN1_RX/PB5, CAN_TX/PB6)</p> <p>Note: This bit is available only in connectivity line devices and is reserved otherwise.</p>
21	ENET_REMAP	<p>Ethernet MAC I/O remapping</p> <p>This bit is set and cleared by software. It controls the Ethernet MAC connections with PHY</p> <p>0: No remap (RX_DV-CRS_DV/PA7, RXD0/PC4, RXD1/PC5, RXD2/PB0, RXD3/PB1)</p> <p>1: Remap (RX_DV-CRS_DV/PD8, RXD0/PD9, RXD1/PD10, RXD2/PD11, RXD3/PD12)</p> <p>Note: This bit is available only in connectivity line devices and is reserved otherwise.</p>
20:17	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 timer TIMER4_CH3 is connected to PA3. When set the IRC40K internal clock connected to TIMER4_CH3 input for calibration purpose.</p> <p>0: No remap</p> <p>1: Remap</p>
15	PD01_REMAP	<p>Port D0/Port D1 remapped 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 alternate interface remapping</p> <p>These bits are set and cleared by software.</p> <p>00: No remap (CAN0_RX/PA11, CAN0_TX/PA12)</p>

		01: Not used
		10: Partial remap (CAN0_RX/PB8, CAN0_TX/PB9)
		11: Full remap (CAN0_RX/PD0, CAN0_TX/PD1)
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_REMAP	<p>TIMER2 remapping</p> <p>[1:0] These bits are set and cleared by software.</p> <p>00: No remap (TIMER2_CH0/PA6, TIMER2_CH1/PA7, TIMER2_CH2/PB0, TIMER2_CH3/PB1)</p> <p>01: Not used</p> <p>10: Partial remap (TIMER2_CH0/PB4, TIMER2_CH1/PB5, TIMER2_CH2/PB0, TIMER2_CH3/PB1)</p> <p>11: Full remap (TIMER2_CH0/PC6, TIMER2_CH1/PC7, TIMER2_CH2/PC8, TIMER2_CH3/PC9)</p>
9:8	TIMER1_REMAP	<p>TIMER1 remapping</p> <p>[1:0] These bits are set and cleared by software.</p> <p>00: No remap (TIMER1_CH0-TIMER1_ETI/PA0, TIMER1_CH1/PA1, TIMER1_CH2/PA2, TIMER1_CH3/PA3)</p> <p>01: Partial remap 0 (TIMER1_CH0-TIMER1_ETI/PA15, TIMER1_CH1/PB3, TIMER1_CH2/PA2, TIMER1_CH3/PA3)</p> <p>10: Partial remap 1 (TIMER1_CH0-TIMER1_ETI/PA0, TIMER1_CH1/PA1, TIMER1_CH2/PB10, TIMER1_CH3/PB11)</p> <p>11: Full remap (TIMER1_CH0-TIMER1_ETI/PA15, TIMER1_CH1/PB3, TIMER1_CH2/PB10, TIMER1_CH3/PB11)</p>
7:6	TIMER0_REMAP	<p>TIMER0 remapping</p> <p>[1:0] These bits are set and cleared by software.</p> <p>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)</p> <p>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)</p> <p>10: Not used</p> <p>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)</p>
5:4	USART2_REMAP	USART2 remapping

	[1:0]	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	USART1_REMAP	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_SC /PA5, SPI0_MISO/PA6, SPI0_MOSI/PA7, SPI0_IO2/PA2, SPI0_IO3/PA3) 1: Remap (SPI0_NSS/PA15, SPI0_SCK/PB3, SPI0_MISO/PB4, SPI0_MOSI/PB5, SPI0_IO2/PB6, SPI0_IO3/PB7)

### 8.5.11. 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			

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
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 0101: PF3 pin 0110: PG3 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 0101: PF2 pin 0110: PG2 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 0101: PF1 pin 0110: PG1 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 0101: PF0 pin 0110: PG0 pin Other configurations are reserved.

### 8.5.12. 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

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 0101: PF7 pin 0110: PG7 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 0101: PF6 pin 0110: PG6 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 0101: PF5 pin 0110: PG5 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 0101: PF4 pin 0110: PG4 pin Other configurations are reserved.
-----	----------------	---

### 8.5.13. 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 0101: PF11 pin 0110: PG11 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 0101: PF10 pin 0110: PG10 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 0101: PF9 pin 0110: PG9 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 0101: PF8 pin 0110: PG8 pin Other configurations are reserved.

#### 8.5.14. 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]			
rw				rw				rw				rw			

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

0101: PF15 pin  
 0110: PG15 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
		0101: PF14 pin
		0110: PG14 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
		0101: PF13 pin
		0110: PG13 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
		0101: PF12 pin
		0110: PG12 pin
		Other configurations are reserved.

### 8.5.15. 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	CTC_REMAP [1:0]	EXMC_N ADV	TIMER13_ REMAP	TIMER12_ REMAP	TIMER10_ REMAP	TIMER9_ REMAP	TIMER8_ REMAP	Reserved
		rw	rw	rw	rw	rw	rw	

Bits	Fields	Descriptions
31:13	Reserved	Must be kept at reset value.
12:11	CTC_REMAP [1:0]	CTC remapping  These bits are set and cleared by software, they control the mapping of the CTC_SYNC alternate function onto the GPIO ports. 00: No remap (PA8) 01: Remap 0 (PD15) 10/11: Remap 1 (PF0)
10	EXMC_NADV	EXMC_NADV connect/disconnect  This bit is set and cleared by software, it controls the use of optional EXMC_NADV signal. 0: The NADV signal is connected to the output (default) 1: The NADV signal is not connected. The I/O pin can be used by another peripheral.
9	TIMER13_REMAP	TIMER13 remapping  This bit is set and cleared by software, it controls the mapping of the TIMER13_CH0 alternate function onto the GPIO ports 0: No remap (PA7) 1: Remap (PF9)
8	TIMER12_REMAP	TIMER12 remapping  This bit is set and cleared by software, it controls the mapping of the TIMER12_CH0 alternate function onto the GPIO ports 0: No remap (PA6) 1: Remap (PF8)
7	TIMER10_REMAP	TIMER10 remapping  This bit is set and cleared by software, it controls the mapping of the TIMER10_CH0 alternate function onto the GPIO ports 0: No remap (PB9) 1: Remap (PF7)
6	TIMER9_REMAP	TIMER9 remapping  This bit is set and cleared by software, it controls the mapping of the TIMER9_CH0 alternate function onto the GPIO ports 0: No remap (PB8) 1: Remap (PF6)
5	TIMER8_REMAP	TIMER8 remapping  This bit is set and cleared by software, it controls the mapping of the TIMER8_CH0

and TIMER8\_CH1 alternate function onto the GPIO ports

0: No remap (TIMER8\_CH0 on PA2 and TIMER8\_CH1 on PA3)

1: Remap (PF6) (TIMER8\_CH0 on PE5 and TIMER8\_CH1 on PE6)

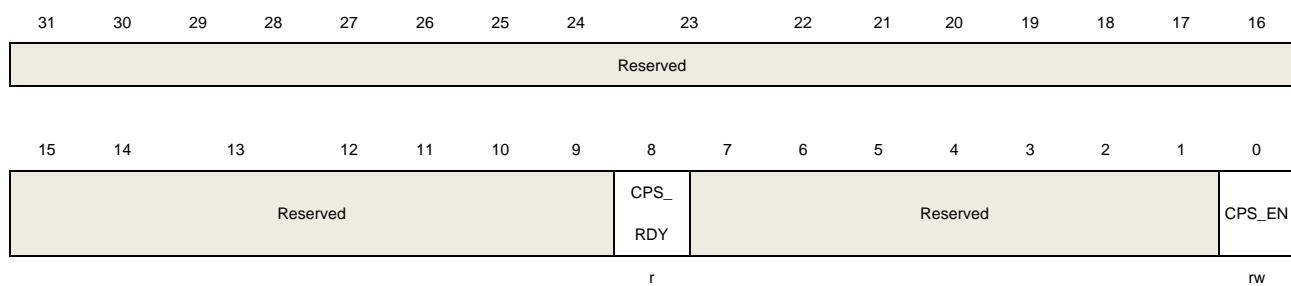
4:0            Reserved            Must be kept at reset value.

### **8.5.16. IO compensation control register (AFIO\_CPSCTL)**

Address offset: 0x20

Reset value: 0x0000 0000

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



<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:9	Reserved	Must be kept at reset value.
8	CPS_RDY	I/O compensation cell is ready or not. This bit is read-only. 0: I/O compensation cell is not ready 1: I/O compensation cell is ready
7:1	Reserved	Must be kept at reset value.
0	CPS_EN	I/O compensation cell enable. When the port output speed is more than 50 MHz, the user should enable the I/O compensation cell. 0: I/O compensation cell is disabled 1: I/O compensation cell is enable

## 9. Cyclic redundancy checks management unit (CRC)

### 9.1. 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 management unit can be used to calculate 32 bit CRC code with fixed polynomial.

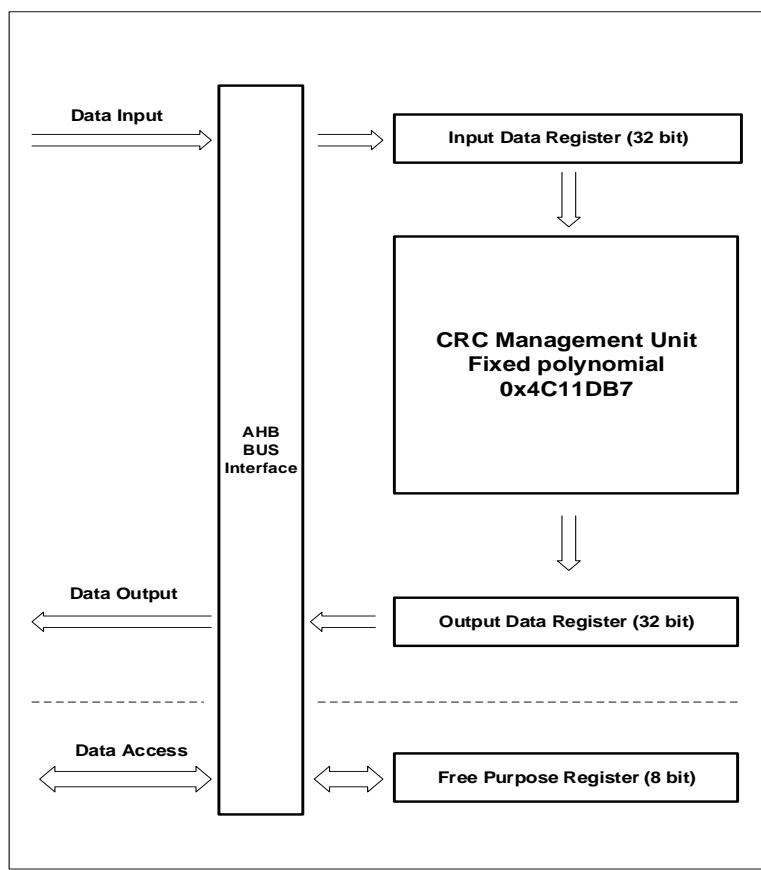
### 9.2. Characteristics

- 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 9-1. Block diagram of CRC calculation unit



### 9.3. Function overview

- CRC management 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.

During CRC calculation 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.

## 9.4. Register definition

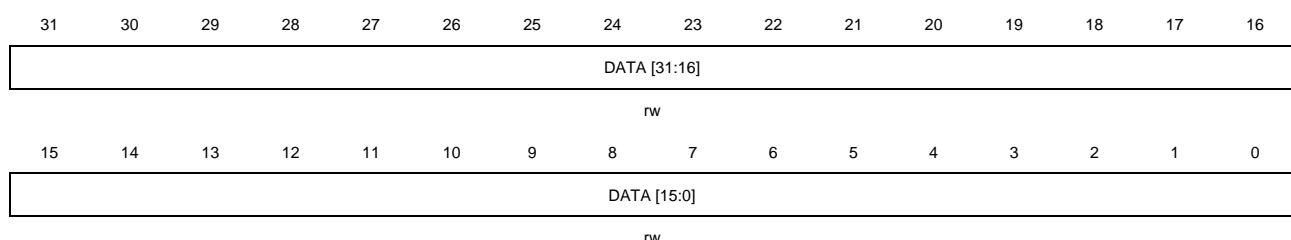
CRC base address: 0x4002 3000

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

Address offset: 0x00

Reset value: 0xFFFF FFFF

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



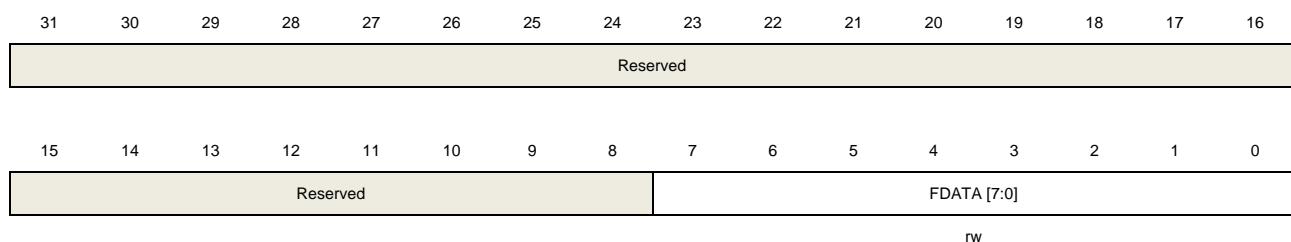
Bits	Fields	Descriptions
31:0	DATA [31:0]	CRC calculation result bits Software writes and reads. This register is used to calculate new data, and the register can be written the new data directly. Written value cannot be read because the read value is the previous CRC calculation result.

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

Address offset: 0x04

Reset value: 0x0000 0000

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



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

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

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															RST

rs

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.

## 10. Direct memory access controller (DMA)

### 10.1. 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.

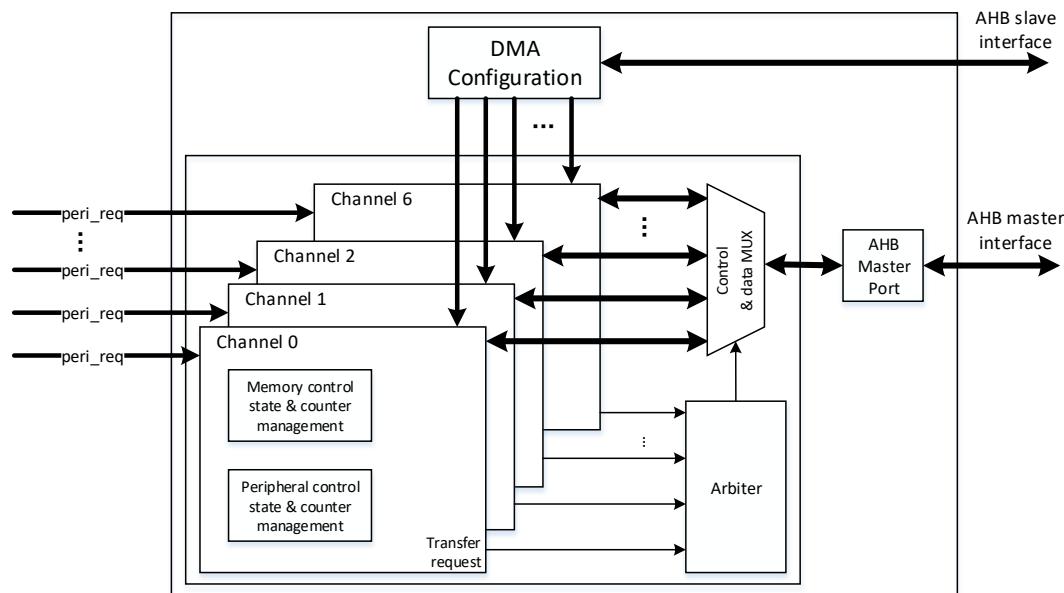
The system bus is shared by the DMA controller and the Cortex®-M4 core. When the DMA and the CPU are targeting the same destination, the DMA access may stop the CPU access to the system bus for some bus cycles. Round-robin scheduling is implemented in the bus matrix to ensure at least half of the system bus bandwidth for the CPU.

### 10.2. 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.

## 10.3. Block diagram

Figure 10-1. Block diagram of DMA



As shown in [Figure 10-1. 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

## 10.4. Function overview

### 10.4.1. DMA operation

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

Suppose DMA\_CHxCNT is 4, and both PNAGA and MNAGA are set. The DMA transfer operations for each combination of PWIDTH and MWIDTH are shown in the following [Table 10-1. DMA transfer operation](#).

**Table 10-1. 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 enable the CHEN bit in the register. During the transmission, the CNT bits indicate the remaining number of data items to be transferred.

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

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

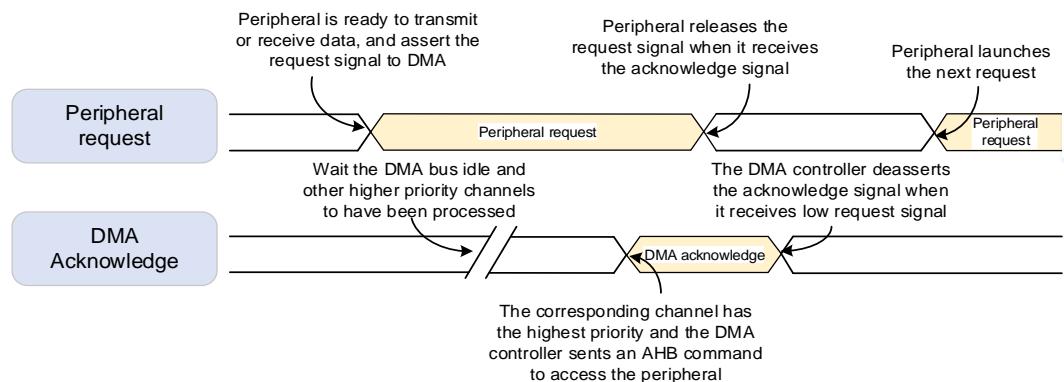
### 10.4.2. Peripheral handshake

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 10-2. Handshake mechanism** shows how the handshake mechanism works between the DMA controller and peripherals.

**Figure 10-2. Handshake mechanism**



### 10.4.3. 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.

### 10.4.4. 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.

#### 10.4.5. 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.

#### 10.4.6. 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.

#### 10.4.7. 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.

#### 10.4.8. 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.

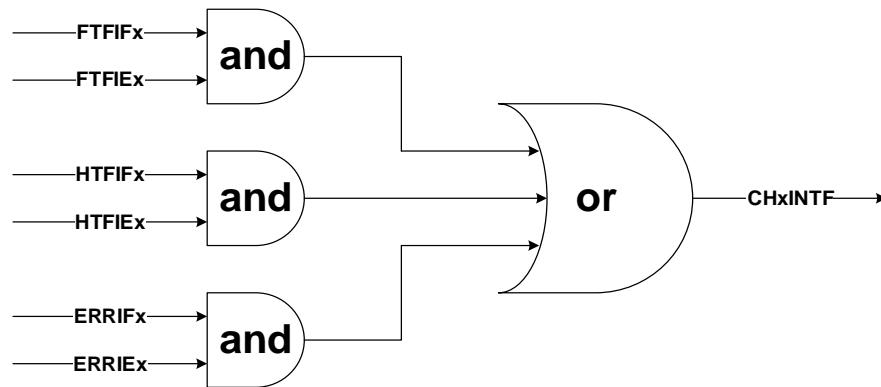
Each interrupt event has a dedicated flag bit in the DMA\_INTF register, a dedicated clear bit in the DMA\_INTC register, and a dedicated enable bit in the DMA\_CHxCTL register. The relationship is described in the following [Table 10-2. interrupt events](#).

**Table 10-2. 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 10-3. DMA interrupt logic](#), an interrupt can be produced when any type of interrupt event occurs and enabled on the channel.

**Figure 10-3. DMA interrupt logic**



**NOTE:** “x” indicates channel number (for DMA0, x=0...6. for DMA1, x=0...4).

#### 10.4.9. 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 10-4. DMA0 request mapping](#) and [Figure 10-5. 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 10-3. DMA0 requests for each channel](#) lists the support request from peripheral for each channel of DMA0, and [Table 10-4. DMA1 requests for each channel](#) lists the support request from peripheral for each channel of DMA1.

Figure 10-4. DMA0 request mapping

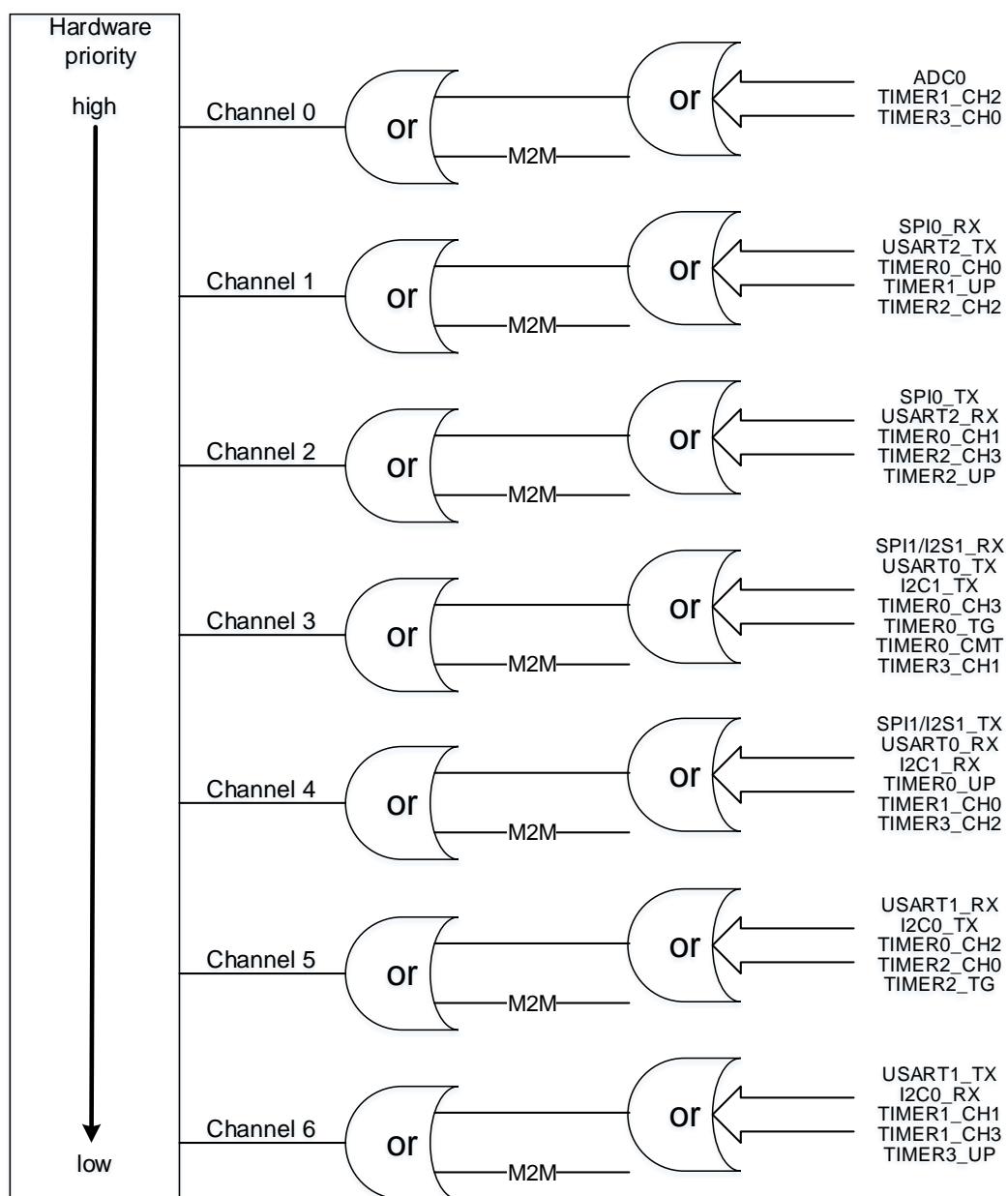
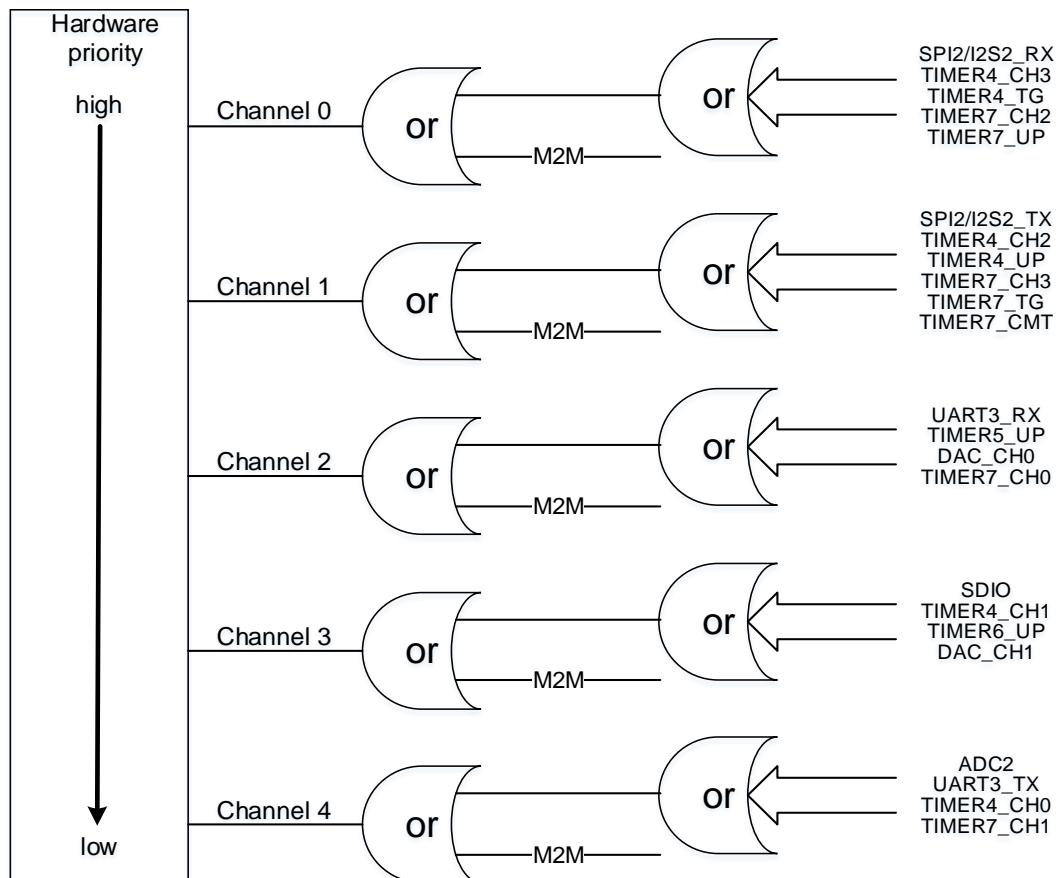


Table 10-3. DMA0 requests for each channel

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_CMT	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	•	•	•	•	•	•

Peripheral	Channel 0	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5	Channel 6
SPI/I2S	•	SPI0_RX	SPI0_TX	SPI1/I2S1_RX	SPI1/I2S1_TX	•	•
USART	•	USART2_TX	USART2_RX	USART0_TX	USART0_RX	USART1_RX	USART1_TX
I2C	•	•	•	I2C1_TX	I2C1_RX	I2C0_TX	I2C0_RX

**Figure 10-5. DMA1 request mapping**



**Table 10-4. DMA1 requests for each channel**

Peripheral	Channel 0	Channel 1	Channel 2	Channel 3	Channel 4
TIMER4	TIMER4_CH3 TIMER4_TG	TIMER4_CH2 TIMER4_UP	•	TIMER4_CH1	TIMER4_CH0
TIMER5	•	•	TIMER5_UP	•	•
TIMER6	•	•	•	TIMER6_UP	•
TIMER7	TIMER7_CH2 TIMER7_UP	TIMER7_CH3 TIMER7_TG TIMER7_CMT	TIMER7_CH0	•	TIMER7_CH1
ADC2	•	•	•	•	ADC2
DAC	•	•	DAC_CH0	DAC_CH1	•
SPI/I2S	SPI2/I2S2_RX	SPI2/I2S2_TX	•	•	•
USART	•	•	UART3_RX	•	UART3_TX

Peripheral	Channel 0	Channel 1	Channel 2	Channel 3	Channel 4
SDIO	•	•	•	SDIO	•

## 10.5. 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.

### 10.5.1. 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

### 10.5.2. 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	HTFIFC2	FTFIFC2	GIFC2	ERRIFC1	HTFIFC1	FTFIFC1	GIFC1	ERRIFC0	HTFIFC0	FTFIFC0	GIFC0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits	Fields	Descriptions
31:28	Reserved	Must be kept at reset value.
27/23/19/	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

### 10.5.3. 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

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:15	Reserved	Must be kept at reset value.
14	M2M	<p>Memory to Memory Mode</p> <p>Software set and cleared</p> <p>0: Disable Memory to Memory Mode</p> <p>1: Enable Memory to Memory mode</p> <p>This bit can not be written when CHEN is '1'.</p>
13:12	PRIO[1:0]	<p>Priority level</p> <p>Software set and cleared</p> <p>00: Low</p> <p>01: Medium</p> <p>10: High</p> <p>11: Ultra high</p> <p>These bits can not be written when CHEN is '1'.</p>
11:10	MWIDTH[1:0]	<p>Transfer data size of memory</p> <p>Software set and cleared</p> <p>00: 8-bit</p> <p>01: 16-bit</p> <p>10: 32-bit</p> <p>11: Reserved</p> <p>These bits can not be written when CHEN is '1'.</p>
9:8	PWIDTH[1:0]	<p>Transfer data size of peripheral</p> <p>Software set and cleared</p> <p>00: 8-bit</p> <p>01: 16-bit</p> <p>10: 32-bit</p> <p>11: Reserved</p> <p>These bits can not be written when CHEN is '1'.</p>
7	MNAGA	<p>Next address generation algorithm of memory</p> <p>Software set and cleared</p> <p>0: Fixed address mode</p> <p>1: Increasing address mode</p> <p>This bit can not be written when CHEN is '1'.</p>
6	PNAGA	<p>Next address generation algorithm of peripheral</p> <p>Software set and cleared</p> <p>0: Fixed address mode</p> <p>1: Increasing address mode</p> <p>This bit can not be written when CHEN is '1'.</p>

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

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

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

Address offset: 0x0C + 0x14 × x

Reset value: 0x0000 0000

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>

### 10.5.5. 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

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>

### 10.5.6. 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

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															

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>

## 11. Debug (DBG)

### 11.1. Introduction

The GD32F30x series provide a large variety of debug, trace and test features. They are implemented with a standard configuration of the Arm® CoreSight™ module together with a daisy chained standard TAP controller. Debug and trace functions are integrated into the Arm® Cortex®-M4. The debug system supports serial wire debug (SWD) and trace functions in addition to standard JTAG debug. The debug and trace functions refer to the following documents:

- Cortex-M4 Technical Reference Manual
- Arm® Debug Interface v5 Architecture Specification

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.

### 11.2. JTAG/SW function description

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

#### 11.2.1. Switch JTAG or SW interface

By default, the JTAG interface is active. The sequence for switching from JTAG to SWD is:

- Send 50 or more TCK cycles with TMS = 1.
- Send the 16-bit sequence on TMS = 111001110011110 (0xE79E LSB first).
- Send 50 or more TCK cycles with TMS = 1.

The sequence for switching from SWD to JTAG is:

- Send 50 or more TCK cycles with TMS = 1.
- Send the 16-bit sequence on TMS = 1110011100111100 (0xE73C LSB first).
- Send 50 or more TCK cycles with TMS = 1.

#### 11.2.2. 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 serial wire debug (SWD) provide 2-pin SW interface, known as SW data input/output (SWDIO) and SW clock (SWCLK). The two SW pin are multiplexed with two of five JTAG pin, which is SWDIO multiplexed with JTMS, SWCLK multiplexed with JTCK. The JTDO is also used as Trace async data output (TRACESWO) when async trace enabled.

The pin assignment are:

PA15	:	JTDI
PA14	:	JTCK/SWCLK
PA13	:	JTMS/SWDIO
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 switch to SW debug mode, the PA15/PB4/PB3 are released to other GPIO functions. If JTAG and SW not used, all 5-pin can be released to other GPIO functions.

Please refer to [JTAG/SWD alternate function remapping.](#)

### 11.2.3. JTAG daisy chained structure

The Cortex-M4 JTAG TAP is connected to a Boundary-Scan (BSD) JTAG TAP. The BSD JTAG IR is 5-bit width, while the Cortec-M4 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 Cortext-M4 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.

### 11.2.4. Debug reset

The JTAG-DP and SW-DP register are in the power on reset domain. The System reset initializes the majority of the Cortex-M4, excluding NVIC and debug logic, (FPB, DWT, and ITM). The NJTRST reset can reset JTAG TAP controller only. So, it can perform debug feature under system reset. Such as, halt-after-reset, which is the debugger sets halt under system reset, and the core halts immediately after the system reset is released.

### 11.2.5. JEDEC-106 ID code

The Cortex-M4 integrates JEDEC-106 ID code, which is located in ROM table and mapped on the address of 0xE00FF000\_0xE00FFFF.

## 11.3. Debug hold function description

### 11.3.1. Debug support for power saving mode

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

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

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

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

When the core halted and the corresponding bit in DBG control register 1 (DBG\_CTL0) 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.

## 11.4. DBG registers

DEBUG base address: 0xE0042000U

### 11.4.1. 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

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

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.	TIMER10_HOLD	TIMER9_HOLD	TIMER8_HOLD	TIMER13_HOLD	TIMER12_HOLD	TIMER11_HOLD	Reserved			CAN1_HO_LD	TIMER6_HOLD	TIMER5_HOLD	TIMER4_HOLD	TIMER7_HOLD	I2C1_HOL_D
	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
I2C0_HOL_D	CAN0_HO_LD	TIMER3_HOLD	TIMER2_HOLD	TIMER1_HOLD	TIMER0_HOLD	WWDGT_HOLD	FWDGT_HOLD	Reserved		TRACE_IOEN	Reserved		STB_HOLD	DSLP_HOLD	SLP_HOLD
	rw	rw	rw	rw	rw	rw	rw			rw			rw	rw	rw

Bits	Fields	Descriptions
31	Reserved	Must be kept at reset value.
30	TIMER10_HOLD	TIMER 10 hold bit This bit is set and reset by software 0: no effect 1: hold the TIMER 10 counter for debug when core halted
29	TIMER9_HOLD	TIMER 9 hold bit

		This bit is set and reset by software 0: no effect 1: hold the TIMER 9 counter for debug when core halted
28	TIMER8_HOLD	TIMER 8 hold bit This bit is set and reset by software 0: no effect 1: hold the TIMER 8 counter for debug when core halted
27	TIMER13_HOLD	TIMER 13 hold bit This bit is set and reset by software 0: no effect 1: hold the TIMER 13 counter for debug when core halted
26	TIMER12_HOLD	TIMER 12 hold bit This bit is set and reset by software 0: no effect 1: hold the TIMER 12 counter for debug when core halted
25	TIMER11_HOLD	TIMER 11 hold bit This bit is set and reset by software 0: no effect 1: hold the TIMER 11 counter for debug when core halted
24:22	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	TIMER6 hold bit This bit is set and reset by software 0: no effect 1: hold the TIMER6 counter for debug when core halted
19	TIMER5_HOLD	TIMER5 hold bit This bit is set and reset by software 0: no effect 1: hold the TIMER5 counter for debug when core halted
18	TIMER4_HOLD	TIMER4 hold bit This bit is set and reset by software 0: no effect 1: hold the TIMER4 counter for debug when core halted
17	TIMER7_HOLD	TIMER7 hold bit This bit is set and reset by software 0: no effect

---

		1: hold the TIMER7 counter for debug when core halted
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	<p>TIMER 0 hold bit</p> <p>This bit is set and reset by software</p> <p>0: no effect</p> <p>1: hold the TIMER 0 counter for debug when core halted</p>
9	WWDGT_HOLD	<p>WWDGT hold bit</p> <p>This bit is set and reset by software</p> <p>0: no effect</p> <p>1: hold the WWDGT counter clock for debug when core halted</p>
8	FWDGT_HOLD	<p>FWDGT hold bit</p> <p>This bit is set and reset by software</p> <p>0: no effect</p> <p>1: hold the FWDGT counter clock for debug when core halted</p>
7:6	Reserved	Must be kept at reset value

5	TRACE_IOEN	Trace pin allocation enable This bit is set and reset by software 0: Trace pin allocation disable 1: Trace pin allocation enable
4:3	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.

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

### 12.1. Overview

A 12-bit successive approximation analog-to-digital converter module(ADC) is integrated on the MCU chip, which can sample analog signals from 16 external channels and 2 internal channels. The 18 ADC sampling channels all support a variety of operation modes. After sampling and conversion, the conversion results can be stored in the corresponding data registers according to the least significant bit alignment or the most significant bit alignment. An on-chip hardware oversample scheme improves performances and reduces the computational burden of MCU.

### 12.2. Characteristics

- High performance.
  - ADC sampling resolution: 12-bit, 10-bit, 8-bit or 6-bit.
  - Foreground calibration function.
  - Programmable sampling time.
  - Data storage mode: the most significant bit and the least significant bit.
  - 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.
- Operation modes.
  - Converts a single channel or scans a sequence of channels.
  - Single operation mode converts selected inputs once per trigger.
  - Continuous operation mode converts selected inputs continuously.
  - Discontinuous operation mode.
  - SYNC mode (the device with two or more ADCs).
- Conversion result threshold monitor function: analog watchdog.
- Interrupt generation.
  - At the end of routine conversions.
  - Analog watchdog event.
- Oversampler.
  - 16-bit data register.
  - Oversampling rate adjustable from 2 to 256x.
  - Programmable data shift up to 8-bit.

- Module supply requirements: 2.6V to 3.6V, and typical power supply voltage is 3.3V.
- Channel input voltage range:  $V_{REFN} \leq V_{IN} \leq V_{REFP}$ .

## 12.3. Pins and internal signals

[\*\*Figure 12-1. ADC module block diagram\*\*](#) shows the ADC block diagram. [\*\*Table 12-1. ADC internal input signals\*\*](#) and [\*\*Table 12-2. ADC input pins definition\*\*](#) gives the ADC pin description.

**Table 12-1. ADC internal input signals**

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

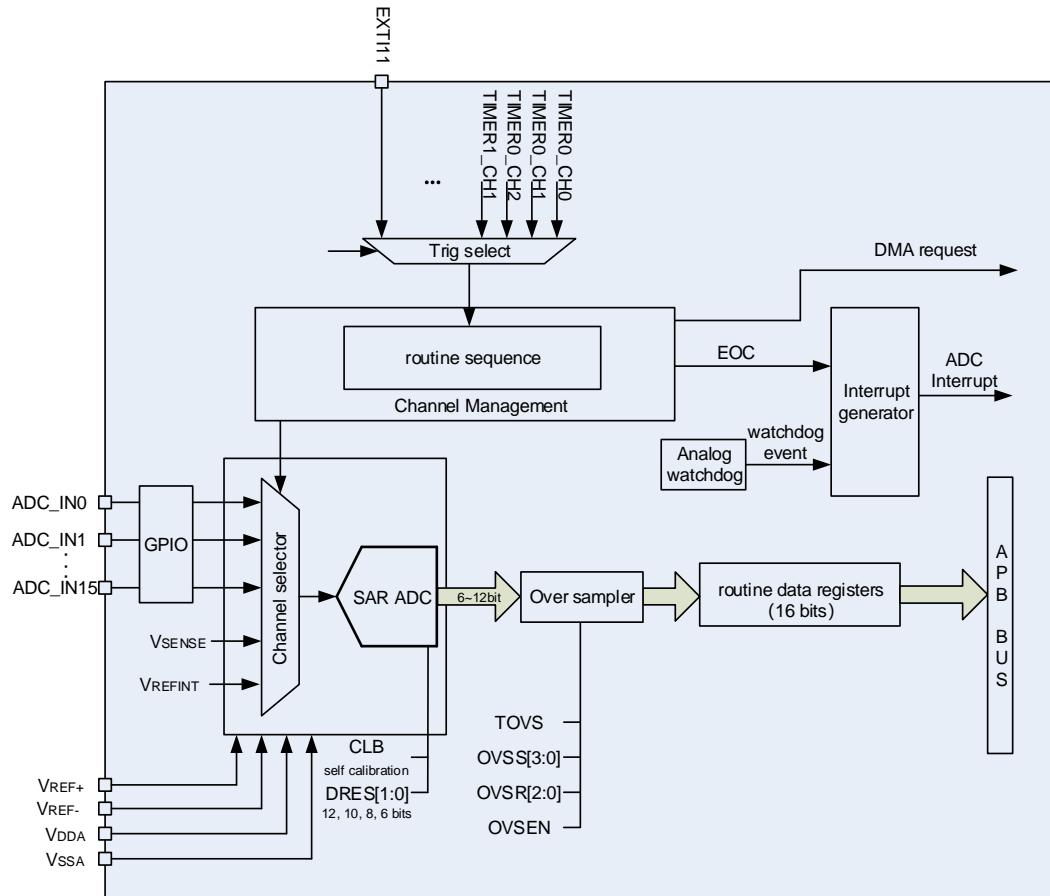
**Table 12-2. ADC input pins definition**

Name	Description
$V_{DDA}$	Analog power supply equal to $V_{DD}$ and $2.6 \text{ V} \leq V_{DDA} \leq 3.6 \text{ V}$
$V_{SSA}$	Ground for analog power supply equal to $V_{SS}$
$V_{REFP}$	The positive reference voltage for the ADC, $2.6 \text{ V} \leq V_{REFP} \leq V_{DDA}$
$V_{REFN}$	The negative reference voltage for the ADC, $V_{REFN} = V_{SSA}$
$ADCx\_IN[15:0]$	Up to 16 external analog channels

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

## 12.4. Functional overview

Figure 12-1. ADC module block diagram



### 12.4.1. Foreground calibration function

During the foreground calibration 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 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  $V_{REFP}$ , 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 CK\_ADC to wait for ADC stability.
3. Set RSTCLB (optional).

4. Set CLB=1.
5. Wait until CLB=0.

### 12.4.2. ADC clock

The CK\_ADC clock is synchronous with the AHB and APB2 clock and provided by the clock controller. ADC clock can be divided and configured by RCU controller.

### 12.4.3. ADC enable

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 off mode. After ADC is enabled, you need delay  $t_{su}$  time for sampling, the value of  $t_{su}$  please refer to the chip datasheet.

### 12.4.4. Routine sequence

The channel management circuit can organize the sampling conversion channels into a sequence: routine sequence. The routine sequence supports up to 16 channels, and each channel is called routine channel.

The RL[3:0] bits in the ADC\_RSQ0 register specify the total conversion sequence length. The ADC\_RSQ0~ADC\_RSQ2 registers specify the selected channels of the routine sequence .

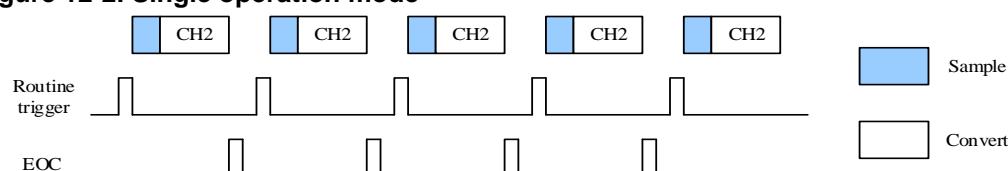
**Note:** Although the ADC supports 18 multiplexed channels, the maximum length of the sequence is only 16.

### 12.4.5. Operation modes

#### Single operation mode

In the single operation mode, the ADC performs conversion on the channel specified in the RSQ0[4:0] bits of ADC\_RSQ2 at a routine trigger. 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 12-2. Single operation mode**



After conversion of a single routine 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.

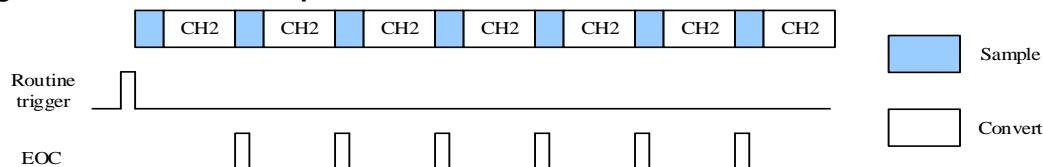
Software procedure for single operation mode of a routine channel:

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 routine sequence.
6. Wait the EOC flag to be set.
7. Read the converted data in the ADC\_RDATA register.
8. Clear the EOC flag by writing 0 to it.

### **Continuous operation mode**

The continuous operation 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 12-3. Continuous operation mode**



Software procedure for continuous operation on a routine 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 routine sequence.
6. Wait the EOC flag to be set.
7. Read the converted data 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:

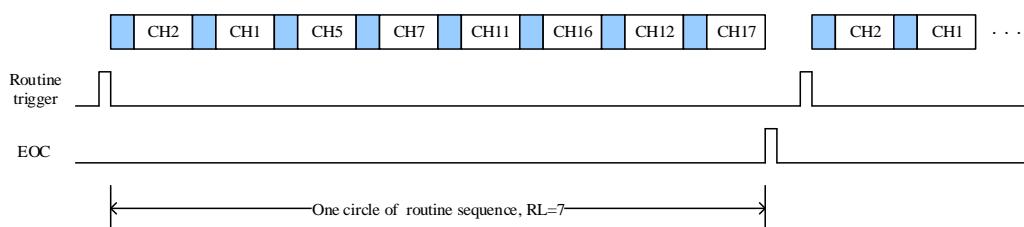
1. Set the CTN and DMA bit in the ADC\_CTL1 register.
2. Configure RSQ0 with the analog channel number.
3. Configure ADC\_SAMPTx register.
4. Configure ETERC and ETSRC bits in the ADC\_CTL1 register if in need.
5. Prepare the DMA module to transfer data from the ADC\_RDATA.
6. Set the SWRCST bit, or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

### Scan operation mode

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

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

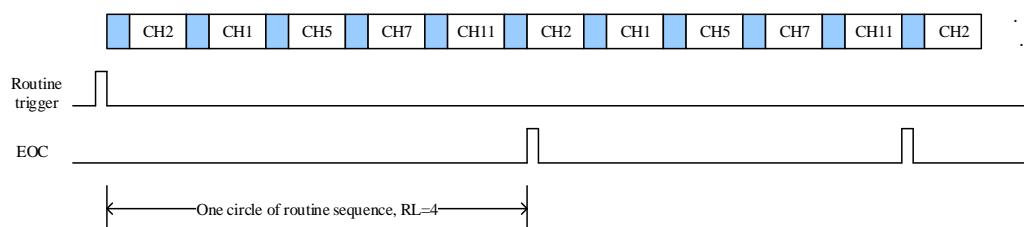
**Figure 12-4. Scan operation mode, continuous disable**



Software procedure for scan operation mode on a routine sequence:

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 routine sequence.
6. Wait the EOC flag to be set.
7. Clear the EOC flag by writing 0 to it.

**Figure 12-5. Scan operation mode, continuous enable**

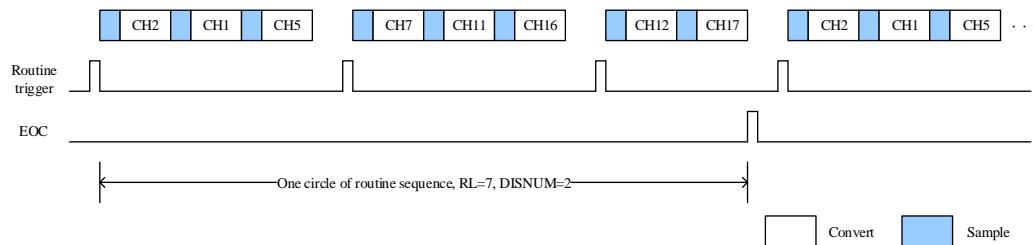


### Discontinuous operation mode

The discontinuous operation 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 does not exceed 8) which is a part of the conversions selected in the ADC\_RSQ0~ADC\_RSQ2 registers. The value of n is configured 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 configured in the ADC\_RSQ0~ADC\_RSQ2 registers until all the channels of routine sequence are done. The EOC will be set after every circle of the routine sequence. An interrupt will be generated if the EOcie bit is set.

**Figure 12-6. Discontinuous operation mode**



Software procedure for discontinuous operation mode on a routine sequence:

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 routine sequence.
7. Repeat step6 if in need.
8. Wait the EOC flag to be set.
9. Clear the EOC flag by writing 0 to it.

#### 12.4.6. Conversion result threshold monitor function

The analog watchdog is enabled when the RWDEN bit in the ADC\_CTL0 register is set for routine sequence. This function is used to monitor whether the conversion result exceeds the set thresholds, and 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 values are independent of the alignment, which is specified by the DAL bit in the ADC\_CTL1 register. One or more channels, which are select by the RWDEN, WDSC and WDCHSEL[4:0] bits in ADC\_CTL0 register, can be monitored by the analog watchdog.

#### 12.4.7. Data storage mode

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

**Figure 12-7. 12-bit Data storage mode**

Routine channel data

0	0	0	0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
DAL=0															

Routine channel data

D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	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 12-8. 6-bit Data storage mode\*\*](#).

**Figure 12-8. 6-bit Data storage mode**

Routine channel data

0	0	0	0	0	0	0	0	0	0	D5	D4	D3	D2	D1	D0
DAL=0															

Routine channel data

0	0	0	0	0	0	0	0	D5	D4	D3	D2	D1	D0	0	0
DAL=1															

## 12.4.8. Sample time configuration

The number of CK\_ADC 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 sampling and conversion time is “sampling time + 12.5” CK\_ADC cycles.

Example:

CK\_ADC = 30MHz and sample time is 1.5 cycles, the total conversion time is “1.5+12.5” CK\_ADC cycles, that means 0.467us.

## 12.4.9. External trigger configuration

The conversion of routine sequence can be triggered by rising edge of external trigger inputs. The external trigger source of routine sequence is controlled by the ETSRC[2:0] bits in the ADC\_CTL1 register.

**Table 12-3. External trigger source for ADC0 and ADC1**

ETSRC[2:0]	Trigger Source	Trigger Type
000	TIMER0_CH0	Hardware trigger
001	TIMER0_CH1	
010	TIMER0_CH2	
011	TIMER1_CH1	

ETSRC[2:0]	Trigger Source	Trigger Type
100	TIMER2_TRGO	
101	TIMER3_CH3	
110	EXTI11/ TIMER7_TRGO	
111	SWRCST	Software trigger

**Table 12-4. External trigger source for ADC2**

ETSRC[2:0]	Trigger Source	Trigger Type
000	TIMER2_CH0	Hardware trigger
001	TIMER1_CH2	
010	TIMER0_CH2	
011	TIMER7_CH0	
100	TIMER7_TRGO	
101	TIMER4_CH0	
110	TIMER4_CH2	
111	SWRCST	Software trigger

#### 12.4.10. DMA request

The DMA request, which is enabled by the DMA bit of ADC\_CTL1 register, is used to transfer data of routine sequence for conversion of more than one channel. The ADC generates a DMA request at the end of conversion of a routine 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.

#### 12.4.11. ADC internal channels

When the TSVREN bit of ADC\_CTL1 register is set, the temperature sensor channel (ADC0\_CH16) and V<sub>REFINT</sub> 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  $t_{s\_temp}$   $\mu$ s (please refer to the datasheet). 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 the chip production process variation, the internal temperature sensor is more appropriate to 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<sub>REFINT</sub>) provides a stable (bandgap) voltage output for the ADC and Comparators. V<sub>REFINT</sub> 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 ( $t_{s\_temp}$   $\mu$ 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 the triggers.
4. Read the internal temperature sensor output voltage( $V_{\text{temperature}}$ ), and get the temperature with the following equation:

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

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

Avg\_Slope: average slope for curve between temperature vs. internal temperature sensor output voltage, the typical value please refer to the datasheet.

#### 12.4.12. Programmable resolution (DRES)

The resolution is configured by programming the DRES[1:0] bits in the ADC\_OVSAMPCTL register. For applications that do not require high data accuracy, lower resolution allows faster conversion time. 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 12-5. tCONV timings depending on resolution](#).

**Table 12-5. tCONV timings depending on resolution**

DRES[1:0] bits	t <sub>CONV</sub> (ADC clock cycles)	t <sub>CONV(ns)</sub> at f <sub>ADC</sub> =30MHz	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> =30MHz
12	12.5	417 ns	1.5	14	467 ns
10	10.5	350 ns	1.5	12	400 ns
8	8.5	283 ns	1.5	10	333 ns
6	6.5	217 ns	1.5	8	267 ns

#### 12.4.13. On-chip hardware oversampling

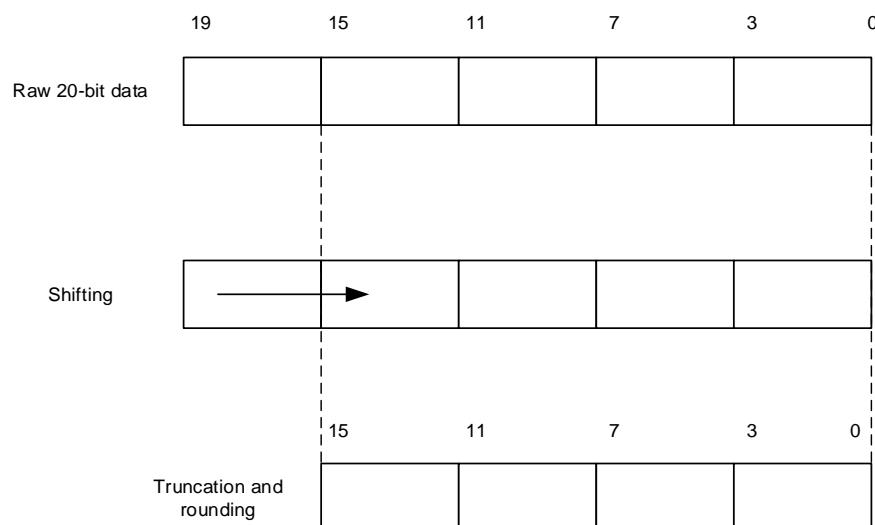
The on-chip hardware oversampling circuit performs data preprocessing to offload the CPU. It can handle multiple conversions and average them into a single data with increased data width, up to 16-bit. The on-chip hardware oversampling circuit is enabled by OVSEN bit in the ADC\_OVSAMPCTL register. It 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:

$$\text{Result} = \frac{1}{M} * \sum_{n=0}^{N-1} D_{out}(n) \quad (12-1)$$

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.

Summation units can produce up to 20 bits ( $256 \times 12\text{-bit}$ ), 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.

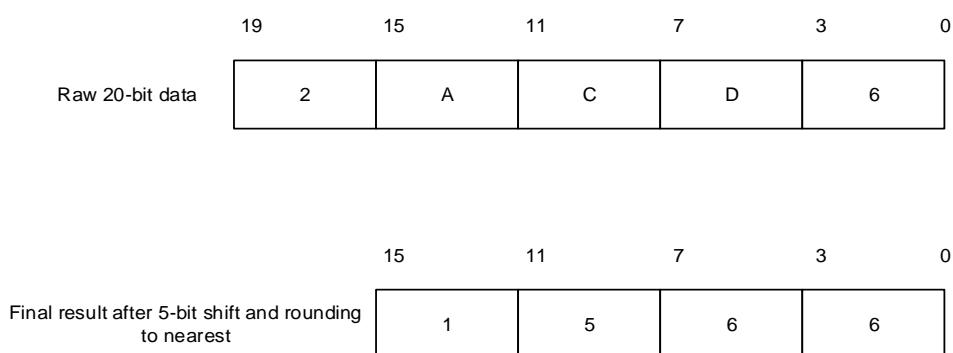
**Figure 12-9. 20-bit to 16-bit result truncation**



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

[\*\*Figure 12-10. 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 12-10. Numerical example with 5-bits shift and rounding**



The [\*\*Table 12-6. 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 12-6. Maximum output results vs N and M Grayed values indicates truncation**

Oversa	Max	No-shift	1-bit	2-bit	3-bit	4-bit	5-bit	6-bit	7-bit	8-bit
--------	-----	----------	-------	-------	-------	-------	-------	-------	-------	-------

mpling ratio	Raw data	OVSS= 0000	shift OVSS= 0001	shift OVSS= 0010	shift OVSS= 0011	shift OVSS= 0100	shift OVSS= 0101	shift OVSS= 0110	shift OVSS= 0111	shift OVSS= 1000
2x	0x1FFE	0x1FFE	0x0FFF	0x07FF	0x03FF	0x01FF	0x00FF	0x007F	0x003F	0x001F
4x	0x3FFC	0x3FFC	0x1FFE	0x0FFF	0x07FF	0x03FF	0x01FF	0x00FF	0x007F	0x003F
8x	0x7FF8	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x07FF	0x03FF	0x01FF	0x00FF	0x007F
16x	0xFFFF0	0xFFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x07FF	0x03FF	0x01FF	0x00FF
32x	0x1FFE0	0xFFE0	0xFFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x07FF	0x03FF	0x01FF
64x	0x3FFC0	0xFFC0	0xFFE0	0xFFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x07FF	0x03FF
128x	0x7FF80	0xFF80	0xFFC0	0xFFE0	0xFFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x07FF
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 sampling time remains equal throughout the oversampling sequence. New data is supplied every N conversions, and the equivalent delay is equal to:

$$N \cdot t_{ADC} = N \cdot (t_{SMPL} + t_{CONV}) \quad (3-1)$$

## 12.5. ADC sync mode

In devices with more than one ADC, the ADC sync mode can be used. In ADC sync mode, the conversion starts alternately or simultaneously triggered by ADC0 to ADC1, according to the sync mode configurated by the SYNCM[3:0] bits in ADC1\_CTL0 register.

In sync mode, when configure the conversion which is triggered by an external event, the ADC1 must be configured as triggered by the software. However, the external trigger must be enabled for ADC0 and ADC1.

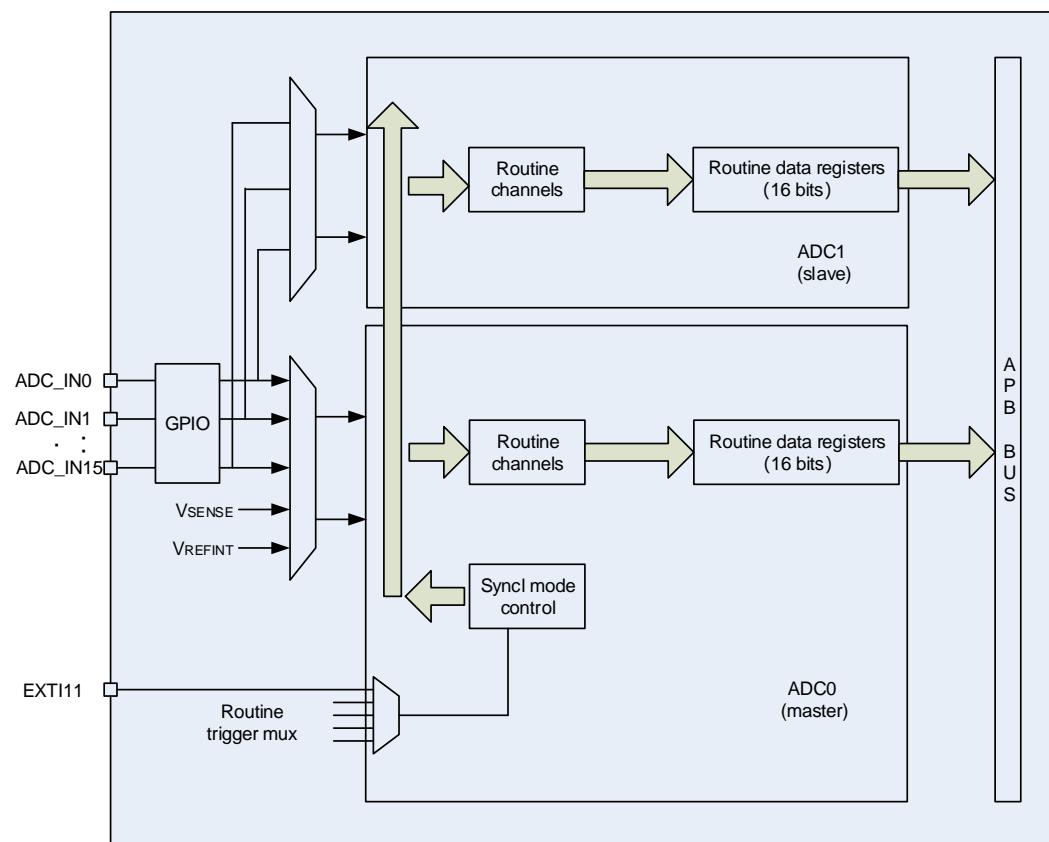
The following modes can be configured in [Table 12-7. ADC sync mode table](#).

**Table 12-7. ADC sync mode table**

SYNCM[3: 0]	mode
0000	Free mode
0110	Routine parallel mode
0111	Routine follow-up fast mode
1000	Routine follow-up slow mode

In ADC sync mode, the DMA bit must be set even if it is not used; the converted data of ADC1 routine channel can be read from the ADC0 data register.

Figure 12-11. ADC sync block diagram



### 12.5.1. Free mode

In this mode, each ADC works independently and does not interfere with each other.

### 12.5.2. Routine parallel mode

This mode converts the routine sequence simultaneously. The source of external trigger comes from the ADC0 routine sequence (configured by the ETSRC[2:0] bits in the ADC\_CTL1 register), and ADC1 routine sequence is configured as software trigger mode.

At the end of conversion event on ADC0 or ADC1, an EOC interrupt is generated (if enabled on one of the two ADC interrupt) when the ADC0/ADC1 routine channels are all converted. The behavior of routine parallel mode shows in the [Figure 12-12. Routine parallel mode on 10 channels](#).

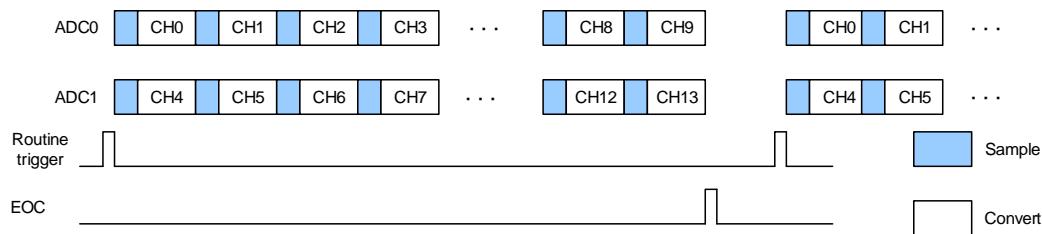
A 32-bit DMA is used, which transfers ADC\_RDATA 32-bit register (the ADC\_RDATA 32-bit register containing the ADC1 converted data in the [31: 16] bits field and the ADC0 converted data in the [15: 0] bits field) to SRAM.

**Note:**

1. If two ADCs use the same sampling channel, it should be ensured that the channel is not used at the same time.

2. Two channels sampled by two ADCs at the same time should be configured with the same sampling time.

**Figure 12-12. Routine parallel mode on 10 channels**



### 12.5.3. Routine follow-up fast mode

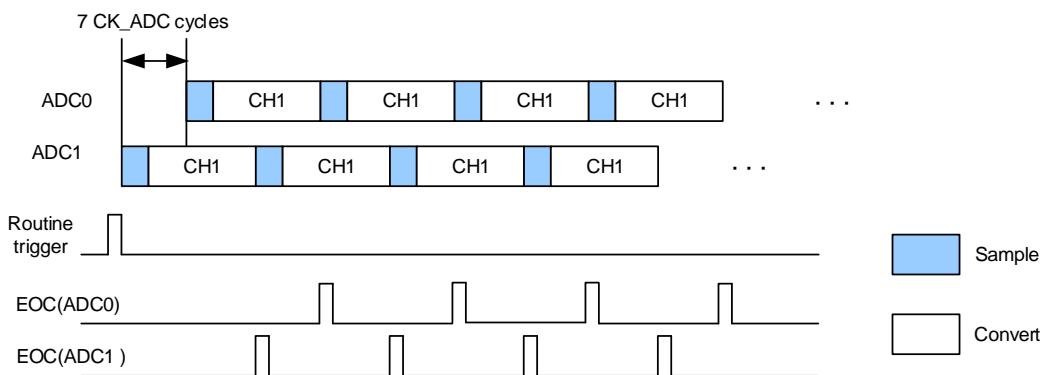
The routine follow-up fast mode is applicable to sample the same channel of two ADCs. The source of external trigger comes from the ADC0 routine channel (selected by the ETSRC[2:0] bits in the ADC\_CTL1 register). When the trigger occurs, ADC1 runs immediately and ADC0 runs after 7 ADC clock cycles.

If the continuous mode is enabled for both ADC0 and ADC1, the selected routine channels of two ADCs are continuously converted. The behavior of follow-up fast mode shows in the [Figure 12-13. Routine follow-up fast mode \(the CTN bit of ADCs are set\).](#)

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 register containing the ADC1 converted data in the [31: 16] bits field and the ADC0 converted data in the [15: 0] bits field.

**Note:** The sampling time of the routine channel of the two ADCs should be less than 7 ADC clock cycles.

[Figure 12-13. Routine follow-up fast mode \(the CTN bit of ADCs are set\)](#)



### 12.5.4. Routine follow-up slow mode

The routine follow-up slow mode is applicable to sample the same channel of two ADCs. The source of external trigger comes from the ADC0 routine channel (selected by the ETSRC[2:0] bits in the ADC\_CTL1 register). When the trigger occurs, ADC1 runs immediately, ADC0 runs

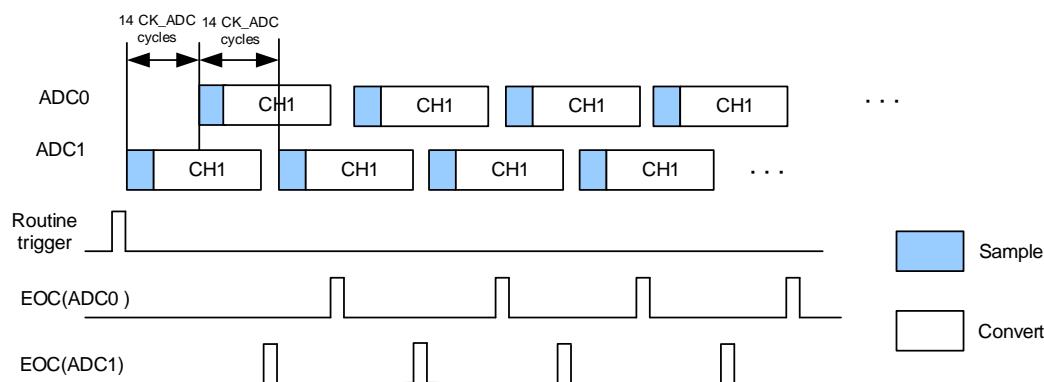
after 14 ADC clock cycles, after the second 14 ADC clock cycles the ADC1 runs again.

Continuous mode can't be used in this mode, because it continuously converts the routine channel. The behavior of follow-up slow mode shows in the [Figure 12-14. Routine follow-up slow mode](#).

After an EOC interrupt is generated by ADC0 (if EOCIE bit is set), we can use a 32-bit DMA, which transfers to SRAM the ADC\_RDATA register containing the ADC1 converted data in the [31: 16] bits field and the ADC0 converted data in the [15: 0] bits field.

**Note:** The maximum sampling time allowed is <14 CK\_ADC cycles to avoid the overlap between ADC0 and ADC1 sampling phases in the event that they convert the same channel.

**Figure 12-14. Routine follow-up slow mode**



## 12.6. ADC interrupts

The interrupt can be produced on one of the events:

- End of conversion for routine sequence.
- The analog watchdog event.

## 12.7. ADC registers

ADC0 base address: 0x4001 2400

ADC1 base address: 0x4001 2800

ADC2 base address: 0x4001 3C00

### 12.7.1. 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	Reserved	EOC	rc_w0	rc_w0	rc_w0

Bits	Fields	Descriptions
31:5	Reserved	Must be kept at reset value.
4	STRC	Start flag of routine sequence conversion 0: Conversion is not started 1: Conversion is started Set by hardware when routine sequence conversion starts. Cleared by software writing 0 to it.
3:2	Reserved	Must be kept at reset value.
1	EOC	End flag of routine sequence conversion 0: No end of routine sequence conversion 1: End of routine sequence conversion Set by hardware at the end of a routine sequence conversion. Cleared by software writing 0 to it or by reading the ADC_RDATA register.
0	WDE	Analog watchdog event flag 0: Analog watchdog event is not happened 1: Analog watchdog event is happening 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.

### 12.7.2. 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	Reserved				SYNCM[3:0]		
									rw					rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DISNUM[2:0]		Reserved	DISRC	Reserved	WDSC	SM	Reserved	WDEIE	EOCIE	WDCHSEL[4:0]					

Bits	Fields	Descriptions
31:24	Reserved	Must be kept at reset value.
23	RWDEN	Routine channel analog watchdog enable 0: Analog watchdog disable 1: Analog watchdog enable
22:20	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. 0001~0101: Reserved 0110: Routine parallel mode 0111: Routine follow-up fast mode 1000: Routine follow-up slow mode 1001~1111: Reserved <b>Note:</b> 1) These bits are only used in ADC0. 2) Users 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 in routine sequence.
12	Reserved	Must be kept at reset value.
11	DISRC	Discontinuous mode on routine sequence 0: Discontinuous operation mode disable 1: Discontinuous operation mode enable
10	Reserved	Must be kept at reset value.
9	WDSC	When in scan mode, analog watchdog is effective on a single channel 0: All channels have analog watchdog function

		1: A single channel has analog watchdog function
8	SM	<p>Scan mode</p> <p>0: Scan operation mode disable</p> <p>1: Scan operation mode enable</p>
7	Reserved	Must be kept at reset value.
6	WDEIE	<p>Interrupt enable for WDE</p> <p>0: Interrupt disable</p> <p>1: Interrupt enable</p>
5	EOCIE	<p>Interrupt enable for EOC</p> <p>0: Interrupt disable</p> <p>1: Interrupt enable</p>
4:0	WDCHSEL[4:0]	<p>Analog watchdog channel select</p> <p>00000: ADC channel0</p> <p>00001: ADC channel1</p> <p>00010: ADC channel2</p> <p>00011: ADC channel 3</p> <p>00100: ADC channel 4</p> <p>00101: ADC channel 5</p> <p>00110: ADC channel 6</p> <p>00111: ADC channel 7</p> <p>01000: ADC channel 8</p> <p>01001: ADC channel 9</p> <p>01010: ADC channel 10</p> <p>01011: ADC channel 11</p> <p>01100: ADC channel 12</p> <p>01101: ADC channel 13</p> <p>01110: ADC channel 14</p> <p>01111: ADC channel15</p> <p>10000: ADC channel16</p> <p>10001: ADC channel17</p> <p>Other values are reserved.</p>
		<p><b>Note:</b> 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>. ADC2 analog inputs Channel16, and Channel17 are internally connected to V<sub>SSA</sub>.</p>

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

Bits	Fields	Descriptions
31:24	Reserved	Must be kept at reset value.
23	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	Software start conversion of routine sequence Set 1 on this bit starts a conversion of a routine sequence if ETSRC is 111. It is set by software and cleared by software or by hardware immediately after the conversion starts.
21	Reserved	Must be kept at reset value.
20	ETERC	External trigger enable for routine sequence 0: External trigger for routine sequence disable 1: External trigger for routine sequence enable
19:17	ETSRC[2:0]	External trigger select for routine sequence 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/ Timer 7 TRGO 111: SWRCST For ADC2: 000: Timer 2 CH0 001: Timer 1 CH2 010: Timer 0 CH2 011: Timer 7 CH0 100: Timer 7 TRGO 101: Timer 4 CH0 110: Timer 4 CH2 111: SWRCST

16:12	Reserved	Must be kept at reset value
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 operation mode disable 1: Continuous operation 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

#### 12.7.4. 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	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
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: channel sampling time is 1.5 cycles 001: channel sampling time is 7.5 cycles 010: channel sampling time is 13.5 cycles 011: channel sampling time is 28.5 cycles 100: channel sampling time is 41.5 cycles 101: channel sampling time is 55.5 cycles 110: channel sampling time is 71.5 cycles 111: channel sampling time is 239.5 cycles

### 12.7.5. 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]					
		rw		rw		rw									
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]					
rw		rw		rw		rw		rw		rw		rw		rw	

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
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: channel sampling time is 1.5 cycles 001: channel sampling time is 7.5 cycles 010: channel sampling time is 13.5 cycles 011: channel sampling time is 28.5 cycles 100: channel sampling time is 41.5 cycles 101: channel sampling time is 55.5 cycles 110: channel sampling time is 71.5 cycles 111: channel sampling time is 239.5 cycles

### 12.7.6. 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]	High threshold for analog watchdog These bits define the high threshold for the analog watchdog.

### 12.7.7. 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]	Low threshold for analog watchdog These bits define the low threshold for the analog watchdog.

### 12.7.8. Routine 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												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]	Routine sequence length. The total number of conversion in routine sequence 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

### 12.7.9. Routine 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]					
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]					

Bits	Fields	Descriptions
31:30	Reserved	Must be kept at reset value.
29:25	RSQ11[4:0]	refer to RSQ0[4:0] description
24:20	RSQ10[4:0]	refer to RSQ0[4:0] description
19:15	RSQ9[4:0]	refer to RSQ0[4:0] description
14:10	RSQ8[4:0]	refer to RSQ0[4:0] description
9:5	RSQ7[4:0]	refer to RSQ0[4:0] description
4:0	RSQ6[4:0]	refer to RSQ0[4:0] description

### 12.7.10. Routine 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]					
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]					

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 routine sequence.

### 12.7.11. Routine 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]															
r															
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 routine channel data  In sync mode, these bits contain the routine data of ADC1. These bits are only used in ADC0.
15:0	RDATA[15:0]	Routine channel data  These bits contain routine channel conversion value, which is read only.

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

Address offset: 0x80

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

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	<p>Triggered Oversampling</p> <p>This bit is set and cleared by software.</p> <p>0: All oversampled conversions for a channel are done consecutively after a trigger</p> <p>1: Each conversion needs a trigger for a oversampled channel and the number of triggers is determined by the oversampling ratio(OVSR[2:0]).</p> <p><b>Note:</b> The software allows this bit to be written only when ADCON = 0 (this ensures that no conversion is in progress).</p>
8:5	OVSS[3:0]	<p>Oversampling shift</p> <p>This bit is set and cleared by software.</p> <p>0000: No shift</p> <p>0001: Shift 1-bit</p> <p>0010: Shift 2-bits</p> <p>0011: Shift 3-bits</p> <p>0100: Shift 4-bits</p> <p>0101: Shift 5-bits</p> <p>0110: Shift 6-bits</p> <p>0111: Shift 7-bits</p> <p>1000: Shift 8-bits</p> <p>Other codes reserved</p> <p><b>Note:</b> The software allows this bit to be written only when ADCON = 0 (this ensures that no conversion is in progress).</p>
4:2	OVSR[2:0]	<p>Oversampling ratio</p> <p>This bit filed defines the number of oversampling ratio.</p> <p>000: 2x</p> <p>001: 4x</p> <p>010: 8x</p> <p>011: 16x</p> <p>100: 32x</p> <p>101: 64x</p> <p>110: 128x</p> <p>111: 256x</p> <p><b>Note:</b> The software allows this bit to be written only when ADCON = 0 (this ensures that no conversion is in progress).</p>
1	Reserved	Must be kept at reset value.
0	OVSEN	<p>Oversampler Enable</p> <p>This bit is set and cleared by software.</p> <p>0: Oversampler disabled</p> <p>1: Oversampler enabled</p> <p><b>Note:</b> The software allows this bit to be written only when ADCON = 0 (this</p>

---

ensures that no conversion is in progress).

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

### 13.1. Introduction

The Digital-to-analog converter converts 12-bit digital data to a voltage on the external pins. The digital data can be configured in 8-bit or 12-bit mode, left-aligned or right-aligned mode. DMA can be used to update the digital data on external triggers. The output voltage can be optionally buffered for higher drive capability.

The two DACs can work independently or concurrently.

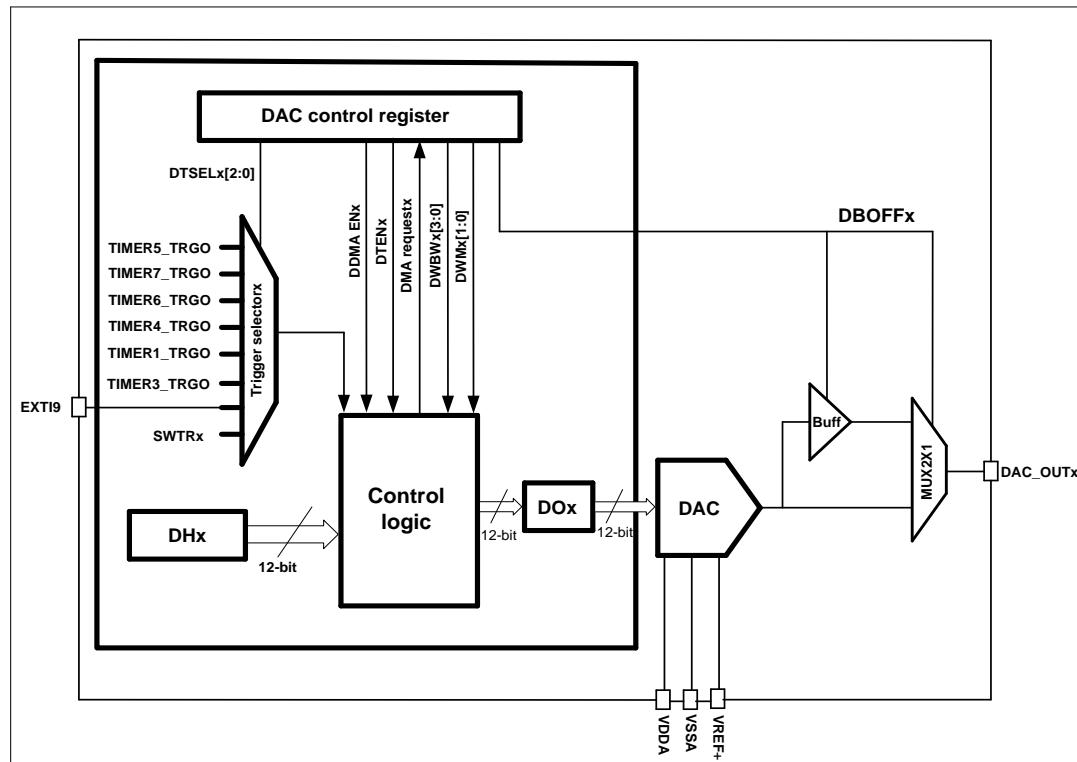
### 13.2. Characteristics

DAC's main features are as follows:

- 8-bit or 12-bit resolution. Right or left data alignment.
- DMA support.
- Conversion update synchronously.
- Conversion triggered by external triggers.
- Configurable internal buffer.
- extern voltage reference, VREF+.
- Noise wave (LFSR noise mode and Triangle noise mode).
- Two DACs in concurrent mode.

[Figure 13-1. DAC block diagram](#) shows the block diagram of DAC and [Table 13-1. DAC I/O description](#) gives the pin description.

Figure 13-1. DAC block diagram



**Note:** The TIMER7\_TRGO trigger is replaced by TIMER2\_TRGO in connectivity line devices.

Table 13-1. DAC I/O description

Name	Description	Signal type
V <sub>DDA</sub>	Analog power supply	Power
V <sub>SSA</sub>	Ground for analog power supply	Power
V <sub>REF+</sub>	reference voltage	Analog Input
DAC_OUTx	DACx analog output	Analog output

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

### 13.3. Function description

#### 13.3.1. DAC enable

The DACs can be powered on by setting the DENx bit in the DAC\_CTL register. A  $t_{WAKEUP}$  time is needed to startup the analog DAC submodule.

#### 13.3.2. DAC output buffer

For reducing output impedance and driving external loads, an output buffer is integrated inside each DAC module.

The output buffer, which is turned on by default, can be turned off by setting the DBOFFx bits in the DAC\_CTL register.

### 13.3.3. DAC data configuration

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

### 13.3.4. DAC trigger

The DAC external trigger is enabled by setting the DTENx bits in the DAC\_CTL register. The DAC external triggers are selected by the DTSELx bits in the DAC\_CTL register.

**Table 13-2. External triggers of DAC**

DTSELx[2:0]	Trigger Source	Trigger Type
3b'000	TIMER5_TRGO	Hardware trigger
3b'001	TIMER 2_TRGO in connectivity line devices; TIMER7_TRGO in other type devices	
3b'010	TIMER6_TRGO	
3b'011	TIMER4_TRGO	
3b'100	TIMER1_TRGO	
3b'101	TIMER3_TRGO	
3b'110	EXTI9	
3b'111	SWTRIG	Software trigger

The TIMERx\_TRGO signals are generated from the timers, while the software trigger can be generated by setting the SWTRx bits in the DAC\_SWT register.

### 13.3.5. DAC workflow

If the external trigger is enabled by setting the DTENx bit in DAC\_CTL register, the DAC holding data is transferred to the DAC output data (DACx\_DO) register when the selected trigger event happened. When the external trigger is disabled, the transfer is performed automatically.

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

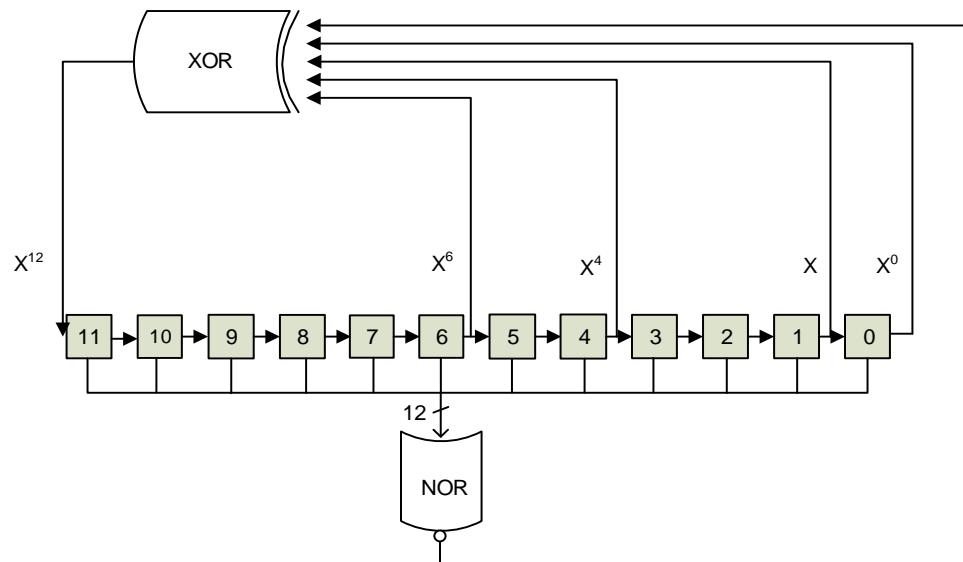
### 13.3.6. DAC noise wave

There are two methods of adding noise wave to the DAC output data: LFSR noise wave mode

and Triangle wave mode. 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.

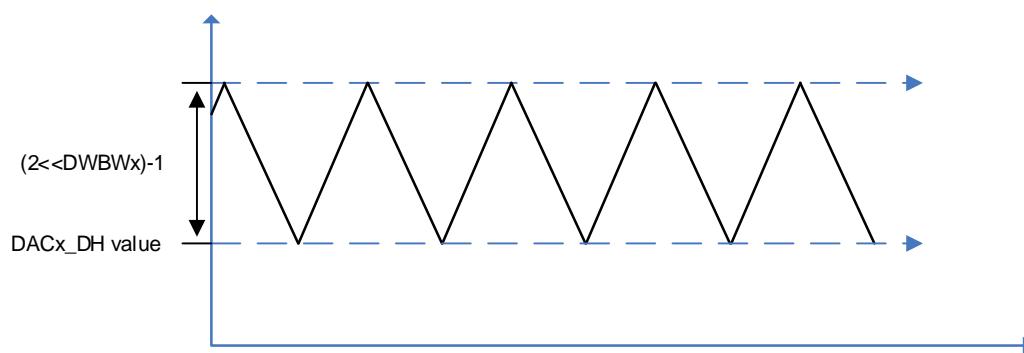
LFSR noise wave mode: there is a Linear Feedback Shift Register (LFSR) in the DAC control logic, it controls the LFSR noise signal which is added to the DACx\_DH value. When the configured DAC noise wave bit width is less than 12, the noise signal equals to the LSB DWBWx bits of the LFSR register, while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

**Figure 13-2. 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  $(2^{<<\text{DWBWx}})-1$ .

**Figure 13-3. DAC triangle noise wave**



### 13.3.7. DAC output calculate

The output voltages on the DAC pin are determined by the following equation:

$$V_{\text{DACx\_out}} = V_{\text{REF+}} * \text{DAC\_D0}/4096 \quad (13-1)$$

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

### 13.3.8. DMA function

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

### 13.3.9. DAC concurrent conversion

In order to maximize the utilization of the bus bandwidth, we can make the two DACs work at the same time using concurrent mode. In this mode, the data transfer (DAC $x$ \_DH to DAC $x$ \_DO) of two DACs is performing at the same time.

There are three concurrent registers that can be used to load the DAC $x$ \_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, DTEN $x$  bit of two DACs must be set both. DTSEL0 and DTSEL1 bits should be configured with the same value.

When DMA is enabled, only one of the DDMAEN $x$  bits should be set.

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

## 13.4. DAC registers

DAC base address: 0x4000 7400

### 13.4.1. 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	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	DDMAEN0		DWBW0[3:0]		DWM0[1:0]		DTSEL0[2:0]		DTEN0	DBOFF0	DENO				
			rw		rw		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: TIMER5 TRGO 001: TIMER2 TRGO (connectivity line devices); TIMER7 TRGO (other type devices) 010: TIMER6 TRGO 011: TIMER4 TRGO 100: TIMER1 TRGO 101: TIMER3 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 turns 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]	DAC0 noise wave mode  These bits specify the mode selection of the noise wave signal of DAC0 when external trigger of DAC0 is enabled (DTEN0=1). 00: wave disabled 01: LFSR noise mode 1x: Triangle noise mode
5:3	DTSEL0[2:0]	DAC0 trigger selection  These bits select the external trigger of DAC0 when DTEN0=1. 000: TIMER5 TRGO 001: TIMER2 TRGO (connectivity line devices); TIMER7 TRGO (other type devices) 010: TIMER6 TRGO 011: TIMER4 TRGO 100: TIMER1 TRGO 101: TIMER3 TRGO 110: EXTI line 9 111: Software trigger
2	DTEN0	DAC0 trigger enable 0: DAC0 trigger disabled 1: DAC0 trigger enabled
1	DBOFF0	DAC0 output buffer turn off 0: DAC0 output buffer turns on to reduce the output impedance and improve the driving capability 1: DAC0 output buffer turns off
0	DEN0	DAC0 enable 0: DAC0 disabled 1: DAC0 enabled

### 13.4.2. 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
w w															

Bits	Fields	Descriptions
31:2	Reserved	Must be kept at reset value.
1	SWTR1	DAC1 software trigger, cleared by hardware 0: Software trigger disabled 1: Software trigger enabled
0	SWTR0	DAC0 software trigger, cleared by hardware 0: Software trigger disabled 1: Software trigger enabled

### 13.4.3. 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															

Bits	Fields	Descriptions
31:12	Reserved	Must be kept at reset value.
11:0	DAC0_DH[11:0]	DAC0 12-bit right-aligned data These bits specify the data that is to be converted by DAC0.

### 13.4.4. 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
Reserved				DAC0_LH[11:0]											
rw															

DAC0_DH[11:0]	Reserved
---------------	----------

rw

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:4	DAC0_DH[11:0]	DAC0 12-bit left-aligned data These bits specify the data that is to be converted by DAC0.
3:0	Reserved	Must be kept at reset value

### 13.4.5. 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

Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value.
7:0	DAC0_DH[7:0]	DAC0 8-bit right-aligned data These bits specify the MSB 8 bits of the data that is to be converted by DAC0.

### 13.4.6. 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.

### 13.4.7. 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

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

### 13.4.8. 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

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

### 13.4.9. 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															

Bits	Fields	Descriptions
31:28	Reserved	Must be kept at reset value.
27:16	DAC1_DH[11:0]	DAC1 12-bit right-aligned data These bits specify the data that is to be converted by DAC1.
15:12	Reserved	Must be kept at reset value.
11:0	DAC0_DH[11:0]	DAC0 12-bit right-aligned data These bits specify the data that is to be converted by DAC0.

### 13.4.10. 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															

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

---

31:20	DAC1_DH[11:0]	DAC1 12-bit left-aligned data These bits specify the data that is to be converted by DAC1.
19:16	Reserved	Must be kept at reset value.
15:4	DAC0_DH[11:0]	DAC0 12-bit left-aligned data These bits specify the data that is to be converted by DAC0.
3:0	Reserved	Must be kept at reset value.

### 13.4.11. 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.

### 13.4.12. 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.

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

Address offset: 0x30

Reset value: 0x0000 0000

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

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_DO [11:0]											

r

<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.

## 14. 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 when the internal counter reaches a given value. The watchdog timer counter can be stopped while the processor is in the debug mode.

### 14.1. Free watchdog timer (FWDGT)

#### 14.1.1. 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.

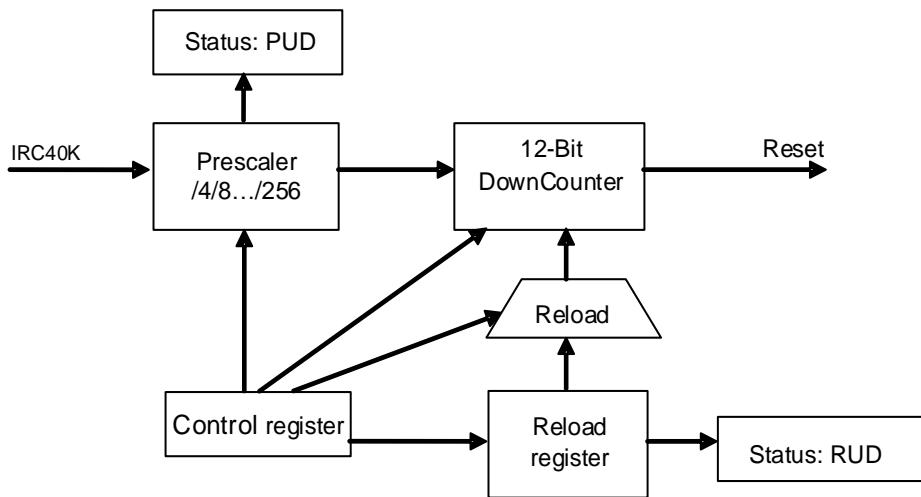
The free watchdog timer causes a reset when the internal down counter reaches 0. The register write protection function in free watchdog can be enabled to prevent it from changing the configuration unexpectedly.

#### 14.1.2. 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 or not when power on.
- FWDGT debug mode, the FWDGT can stop or continue to work in debug mode.

#### 14.1.3. Function overview

The free watchdog consists of an 8-stage prescaler and a 12-bit down-counter. [Figure 14-1. Free watchdog block diagram](#) shows the functional block of the free watchdog module.

**Figure 14-1. Free watchdog block diagram**


The free watchdog is enabled by writing the value (0xFFFF) to the control register (FWDGT\_CTL), then the counter starts counting down. When the counter reaches the value (0x000), there will be a reset.

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 when power on if the hardware free watchdog bit in the device option bits is set. To avoid a reset, the software should reload the counter before the counter reaches (0x000).

The FWDGT\_PSC register and the FWDGT\_RLD register are written 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 ongoing, 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 Cortex®-M4 core halted (Debug mode). The FWDGT stops in Debug mode if the FWDGT\_HOLD bit is set.

**Table 14-1. 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	0.025	409.525
1 / 8	001	0.025	819.025
1 / 16	010	0.025	1638.025
1 / 32	011	0.025	3276.025
1 / 64	100	0.025	6552.025
1 / 128	101	0.025	13104.025

<b>Prescaler divider</b>	<b>PSC[2:0] bits</b>	<b>Min timeout (ms) RLD[11:0]=0x000</b>	<b>Max timeout (ms) RLD[11:0]=0xFFFF</b>
1 / 256	110 or 111	0.025	26208.025

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

**Note:** When after the execution of watchdog reload operation, if the MCU needs enter the deepsleep / standby mode immediately, more than 3 IRC40K clock intervals must be inserted in the middle of reload and deepsleep / standby mode commands by software setting.

#### 14.1.4. 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												PSC[2:0]			
rw															

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 (Before entering low-power mode, it is necessary to wait until PUD is reset).

### **Reload register (FWDGT\_RLD)**

Address offset: 0x08

Reset value: 0x0000 0FFF

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

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				RLD [11:0]											
rw															

Bits	Fields	Descriptions
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 (Before entering low-power mode, it is necessary to wait until RUD is reset).</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															
													r	r	

Bits	Fields	Descriptions
31:2	Reserved	Must be kept at reset value.
1	RUD	<p>Free watchdog timer counter reload value update.</p> <p>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.</p>
0	PUD	<p>Free watchdog timer prescaler value update.</p> <p>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.</p>

## 14.2. Window watchdog timer (WWDGT)

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

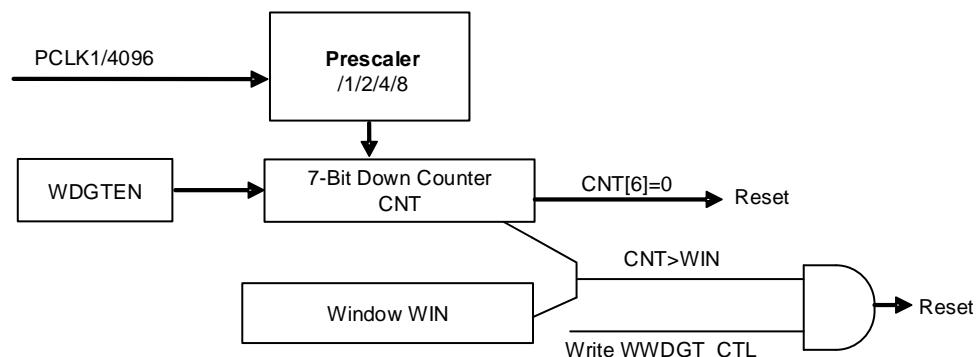
### 14.2.2. Characteristics

- 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): the watchdog is started and the interrupt is enabled, the interrupt occurs when the counter reaches 0x40.
- WWDGT debug mode, the WWDGT can stop or continue to work in debug mode.

### 14.2.3. Function overview

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

**Figure 14-2. Window watchdog timer block diagram**



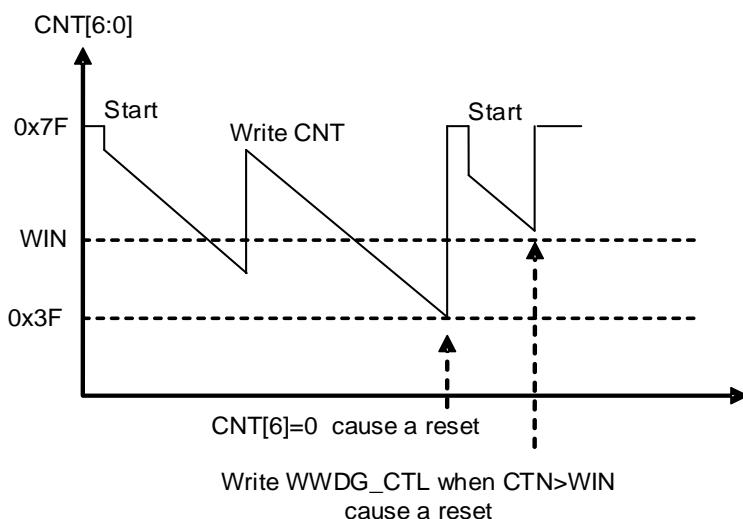
The window watchdog timer is always disabled after power on reset. The software starts the watchdog by setting the WDGTE bit in the WWDGT\_CTL register. When window watchdog timer is enabled, the counter counts down all the time, the configured value of the counter should be greater than 0x3F (it implies that the CNT[6] bit should be set). The CNT[5:0] determine the maximum time interval between two reloading. The count down 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 down counter. The 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 will be generated when the counter reaches 0x40. 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 14-3. Window watchdog timing diagram**



Calculate the WWDGT timeout by using the formula below.

$$t_{\text{WWDGT}} = t_{\text{PCLK1}} \times 4096 \times 2^{\text{PSC}} \times (\text{CNT}[5:0] + 1) \quad (\text{ms}) \quad (16-1)$$

where:

$t_{\text{WWDGT}}$ : WWDGT timeout

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

The table below shows the minimum and maximum values of the  $t_{\text{WWDGT}}$ .

Table 14-2. Min/max timeout value at 60 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	68.2 $\mu$ s	4.3ms
1 / 2	01	136.4 $\mu$ s	8.6 ms
1 / 4	10	272.8 $\mu$ s	17.2 ms
1 / 8	11	545.6 $\mu$ s	34.4 ms

If the WWDGT\_HOLD bit in DBG module is cleared, the WWDGT continues to work even the Cortex®-M4 core halted (Debug mode). While the WWDGT\_HOLD bit is set, the WWDGT stops in Debug mode.

#### 14.2.4. Register definition

WWDGT 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).

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								WDGTEN	CNT[6:0]						
rs								rw							

Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value.
7	WDGTEN	Start the window watchdog timer. Cleared by a hardware reset. Writing 0 has no effect. 0: Window watchdog timer disabled 1: Window watchdog timer enabled
6:0	CNT[6:0]	The value of the watchdog timer counter. A reset 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).

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								EWIE	PSC[1:0]						
rs								rw							

Bits	Fields	Descriptions
31:10	Reserved	Must be kept at reset value.
9	EWIE	Early wakeup interrupt enable. If the bit is set, an interrupt occurs when the counter

reaches 0x40. It can be cleared by a hardware reset or software reset by setting the WWDGTRST bit of the RCU module. 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

Bits	Fields	Descriptions
31:1	Reserved	Must be kept at reset value.
0	EWIF	Early wakeup interrupt flag. When the counter reaches 0x40, 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.

## 15. Real-time Clock(RTC)

### 15.1. 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.

### 15.2. 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

### 15.3. Function overview

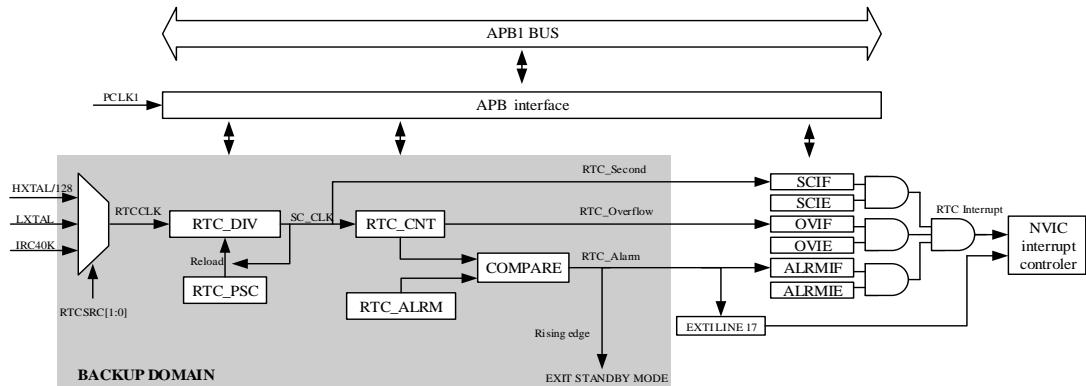
The RTC circuits consist of two major units: APB interface located in PCLK1 clock domain and RTC core located in RTC clock domain.

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 15-1. Block diagram of RTC**



### 15.3.1. 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).

### 15.3.2. 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.

### 15.3.3. 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'.

#### 15.3.4. RTC flag assertion

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

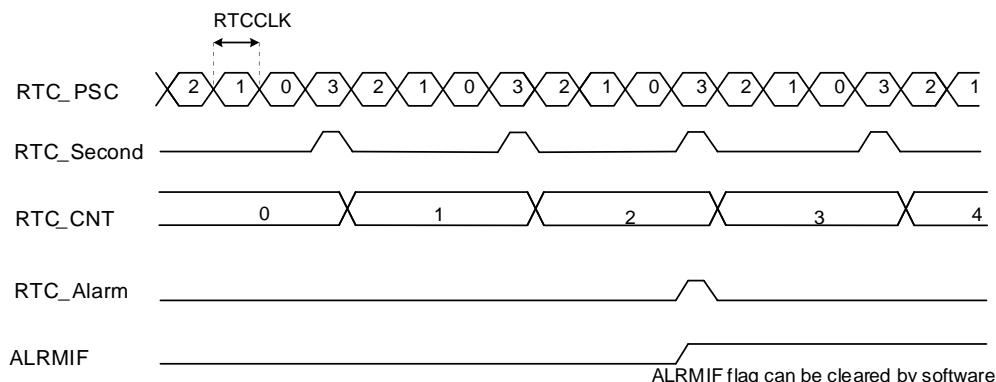
Before the counter equal to the RTC Alarm value which stored in the Alarm register increases by one, the RTC Alarm interrupt flag (ALRMIF) is asserted on the last RTCCLK cycle.

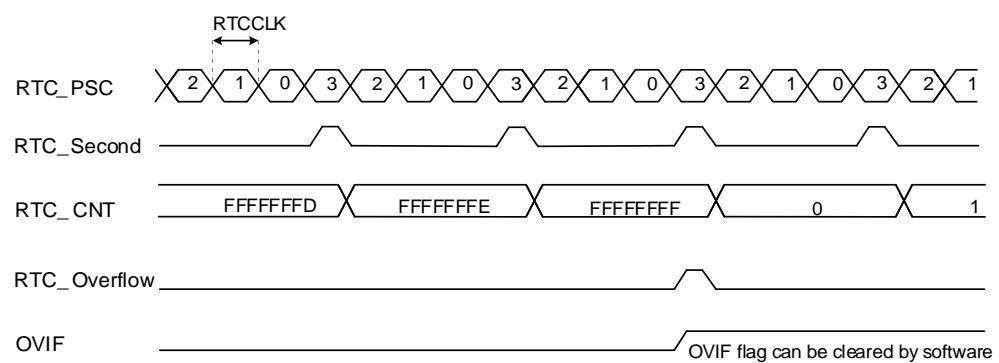
Before the counter equals to 0x0, the RTC Overflow interrupt flag (OVIF) is asserted on the last RTCCLK cycle.

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 15-2. RTC second and alarm waveform example (RTC\_PSC = 3, RTC\_ALRM = 2)**



**Figure 15-3. RTC second and overflow waveform example (RTC\_PSC= 3)**


## 15.4. RTC Register

RTC base address: 0x4000 2800

### 15.4.1. 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	

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

### 15.4.2. 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		

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

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.

### 15.4.3. 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															

Bits	Fields	Descriptions
31:4	Reserved	Must be kept at reset value.

3:0            PSC[19:16]            RTC prescaler value high

#### 15.4.4. 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

Bits	Fields	Descriptions
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).

#### 15.4.5. 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												DIV[19:16]			

r

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

#### 15.4.6. 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															

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

#### **15.4.7. 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															

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

#### **15.4.8. 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)

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:16	Reserved	Must be kept at reset value.

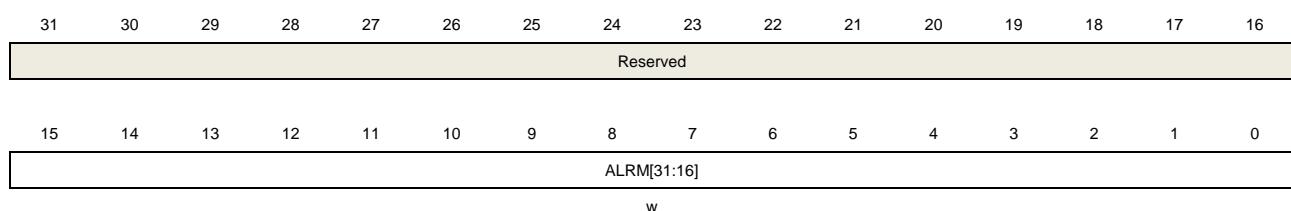
15:0      CNT[15:0]      RTC counter value low

### 15.4.9. 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)



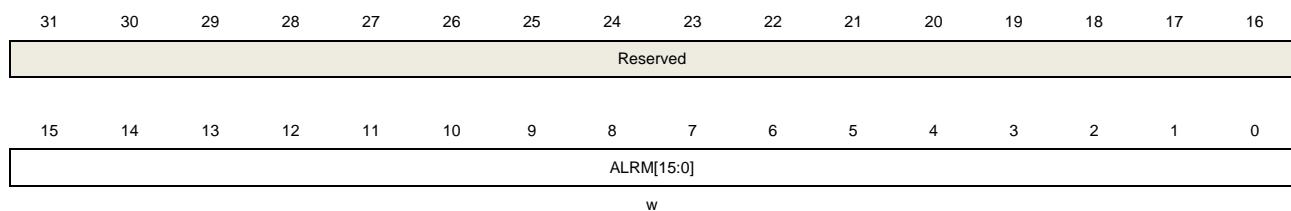
Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	ALRM[31:16]	RTC alarm value high

### 15.4.10. 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)



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

## 16. Timer(TIMERx)

Table 16-1. Timers (TIMERx) are divided into five sorts

TIMER	TIMER0/7	TIMER1~4	TIMER8/11	TIMER9/10/12/13	TIMER5/6
<b>TYPE</b>	Advanced	General-L0	General-L1	General-L2	Basic
<b>Prescaler</b>	16-bit	16-bit	16-bit	16-bit	16-bit
<b>Counter</b>	16-bit	16-bit	16-bit	16-bit	16-bit
<b>Count mode</b>	UP,DOWN, Center-aligned	UP,DOWN, Center-aligned	UP ONLY	UP ONLY	UP ONLY
<b>Repetition</b>	•	✗	✗	✗	✗
<b>CH Capture/ Compare</b>	4	4	2	1	0
<b>Complementary &amp; Dead-time</b>	•	✗	✗	✗	✗
<b>Break</b>	•	✗	✗	✗	✗
<b>Single Pulse</b>	•	•	•	✗	•
<b>Quadrature Decoder</b>	•	•	✗	✗	✗
<b>Master-slave management</b>	•	•	•	✗	✗
<b>Inter connection</b>	• <sup>(1)</sup>	• <sup>(2)</sup>	• <sup>(3)</sup>	✗	TRGO TO DAC
<b>DMA</b>	•	•	✗	✗	• <sup>(4)</sup>
<b>Debug Mode</b>	•	•	•	•	•

(1) TIMER0    **ITI0:** TIMER4\_TRGO    **ITI1:** TIMER1\_TRGO    **ITI2:** TIMER2\_TRGO    **ITI3:** TIMER3\_TRGO  
                   TIMER7    **ITI0:** TIMER0\_TRGO    **ITI1:** TIMER1\_TRGO    **ITI2:** TIMER3\_TRGO    **ITI3:** TIMER4\_TRGO

(2) TIMER1    **ITI0:** TIMER0\_TRGO    **ITI1:** refer to note (5)    **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:** TIMER7\_TRGO  
                   TIMER4    **ITI0:** TIMER1\_TRGO    **ITI1:** TIMER2\_TRGO    **ITI2:** TIMER3\_TRGO    **ITI3:** TIMER7\_TRGO

(3) TIMER8    **ITI0:** TIMER1\_TRGO    **ITI1:** TIMER2\_TRGO    **ITI2:** TIMER9\_TRGO    **ITI3:** TIMER10\_TRGO  
                   TIMER11    **ITI0:** TIMER3\_TRGO    **ITI1:** TIMER4\_TRGO    **ITI2:** TIMER12\_TRGO    **ITI3:** TIMER13\_TRGO

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

In connectivity line devices, the source of TIMER1 ITI1 is decided by TIMER1ITI1\_REMAP in [AFIO port configuration register 0 \(AFIO\\_PCF0\)](#):

In non-connectivity line devices, the source of TIMER1 ITI1 is internally connected to TIMER7\_TRGO;

## 16.1. Advanced timer (TIMERx, x=0, 7)

### 16.1.1. Overview

The advanced timer module (Timer0&Timer7) 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.

Timer and timer are completely independent with each other, but they may be synchronized to provide a larger timer with their counters incrementing in unison.

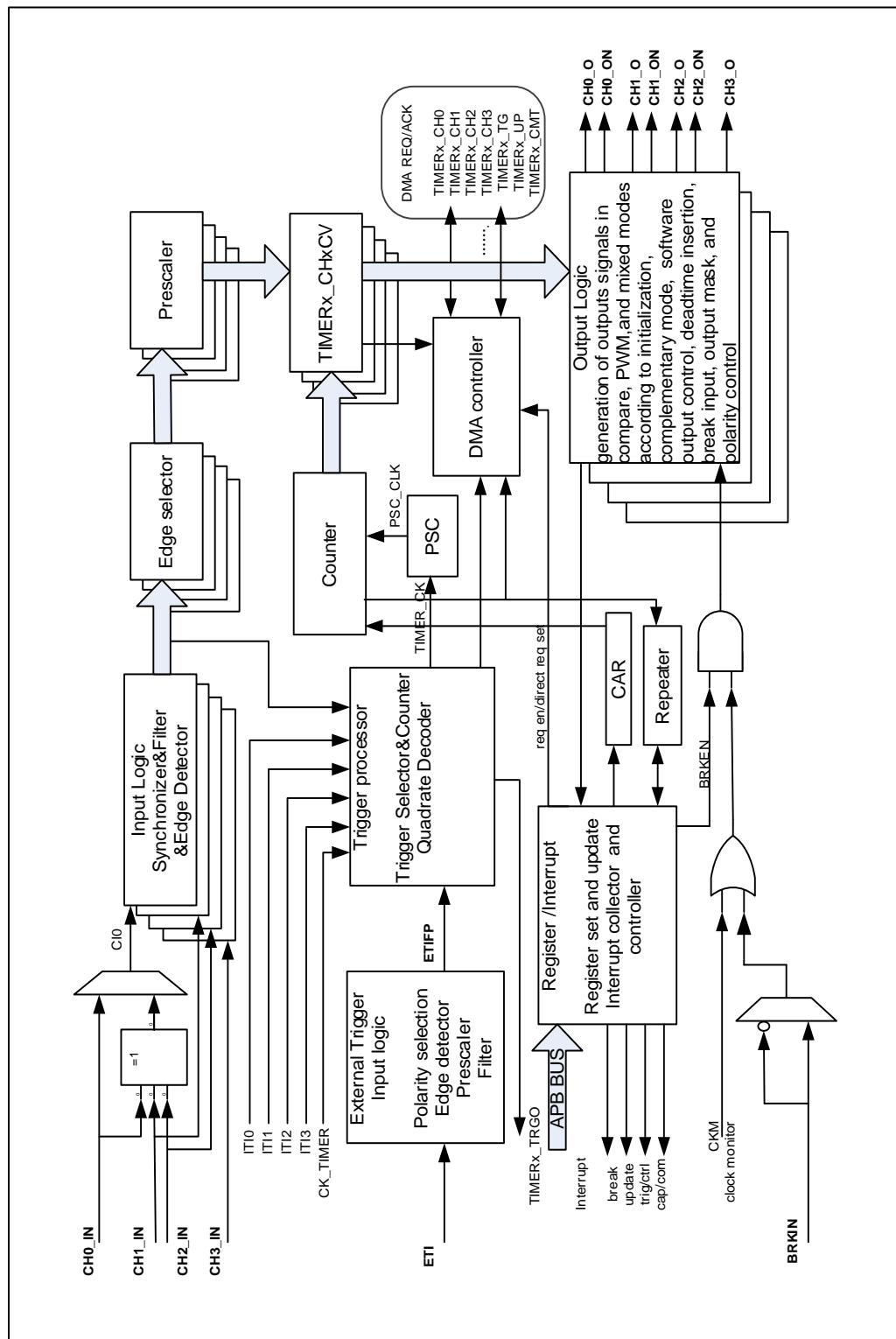
### 16.1.2. Characteristics

- Total channel num: 4.
- Counter width: 16 bits.
- Source of counter clock is selectable: internal clock, internal trigger, external input, external trigger.
- Multiple counter modes: count up, count down, count up/down.
- Quadrature Decoder: used to track motion and determine both rotation direction and position.
- Hall sensor: for 3-phase motor control.
- Programmable prescaler: 16 bits. The factor can be changed on the go.
- Each channel is user-configurable: input capture mode, output compare mode, programmable PWM mode, single pulse mode
- Programmable dead time insertion.
- Auto reload function.
- Programmable counter repetition function.
- Break input.
- Interrupt output or DMA request on: update, trigger event, compare/capture event, commutation event and break input.
- Daisy chaining of timer modules allows a single timer to initiate multiple timers.
- Timer synchronization allows selected timers to start counting on the same clock cycle.
- Timer master-slave management.

### 16.1.3. Block diagram

[Figure 16-1. Advanced timer block diagram](#) provides details of the internal configuration of the advanced timer.

**Figure 16-1. Advanced timer block diagram**



### 16.1.4. Function overview

#### Clock source configuration

The advanced timer has the capability of being clocked by either the CK\_TIMER or an alternate clock source controlled by SMC (TIMERx\_SMCFG bit [2:0]).

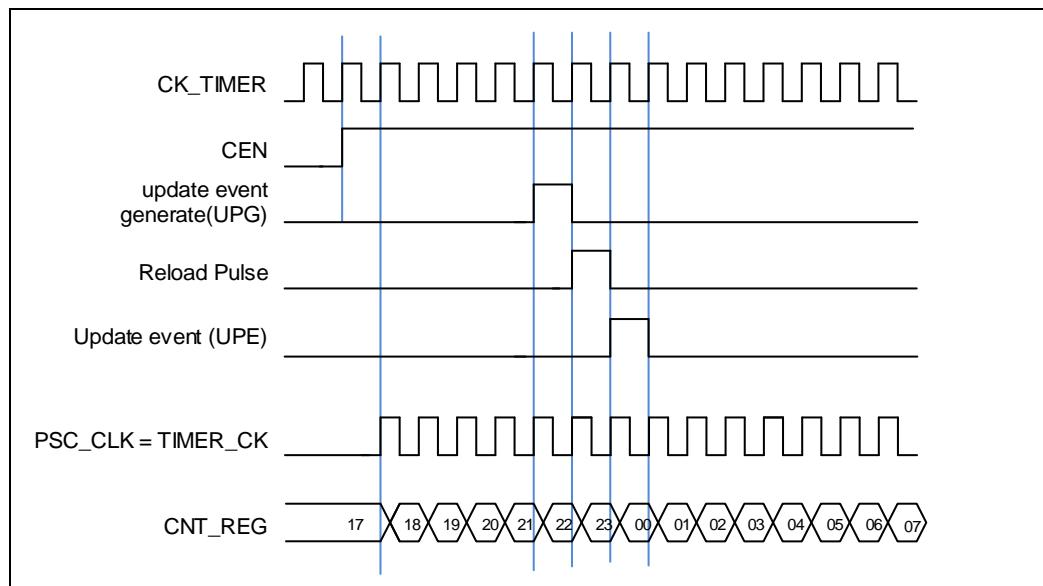
- SMC [2:0] == 3'b000. Internal clock CK\_TIMER is selected as timer clock source which is from module RCU.

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

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

If the SMC [2:0] in the TIMERx\_SMCFG register are setting to an available value including 0x1, 0x2, 0x3 and 0x7, the prescaler is clocked by other clock sources selected by the TRGS [2:0] in the TIMERx\_SMCFG register, details as follows. When the SMC [2:0] bits are set to 0x4, 0x5 or 0x6, the internal clock CK\_TIMER is the counter prescaler driving clock source.

**Figure 16-2. Timing chart of 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.

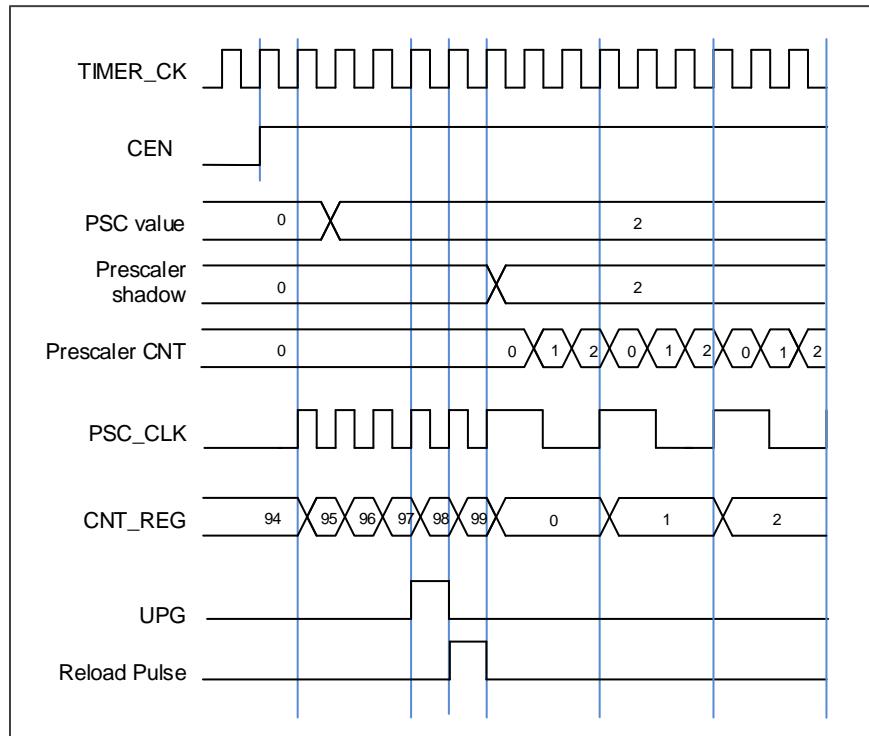
- SMC1== 1'b1 (external clock mode 1). External input ETI is selected as timer clock source (ETI)

The TIMER\_CK, which drives counter's prescaler to count, can be triggered by the event of rising or falling edge on the external pin ETI. This mode can be selected by setting the SMC1 bit in the TIMERx\_SMCFG register to 1. The other way to select the ETI signal as the clock source is to set the SMC [2:0] to 0x7 and the TRGS [2:0] to 0x7 respectively. Note that the ETI signal is derived from the ETI pin sampled by a digital filter. When the ETI signal is selected as clock source, the trigger controller including the edge detection circuitry will generate a clock pulse on each ETI signal rising edge to clock the counter prescaler.

### Clock prescaler

The counter clock (PSC\_CK) is obtained by the TIMER\_CK through the prescaler, and the prescale factor can be configured from 1 to 65536 through the prescaler register (TIMERx\_PSC). The new written prescaler value will not take effect until the next update event.

**Figure 16-3. Timing chart of PSC value change from 0 to 2**



### Counter up counting

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 will start counting up from 0 again and an overflow event will be generated. In addition, the update events will be generated after (TIMERx\_CREP+1)

times of overflow events. The counting direction bit DIR in the TIMERx\_CTL0 register should be set to 0 for the up counting mode.

Whenever, if the update event software trigger is enabled by setting the UPG bit in the TIMERx\_SWEVG register, the counter value will be initialized to 0 and generates an update event.

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

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

**Figure 16-4. Timing chart of up counting mode, PSC=0/2** show some examples of the counter behavior for different clock prescaler factor when TIMERx\_CAR=0x99.

**Figure 16-4. Timing chart of up counting mode, PSC=0/2**

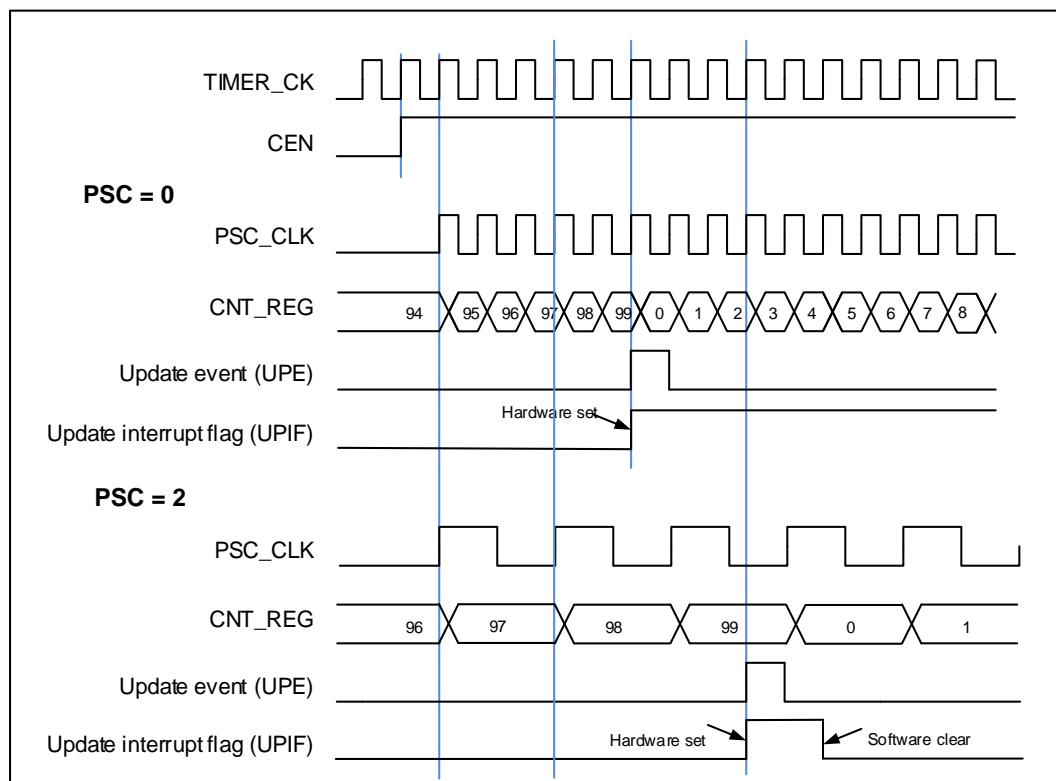
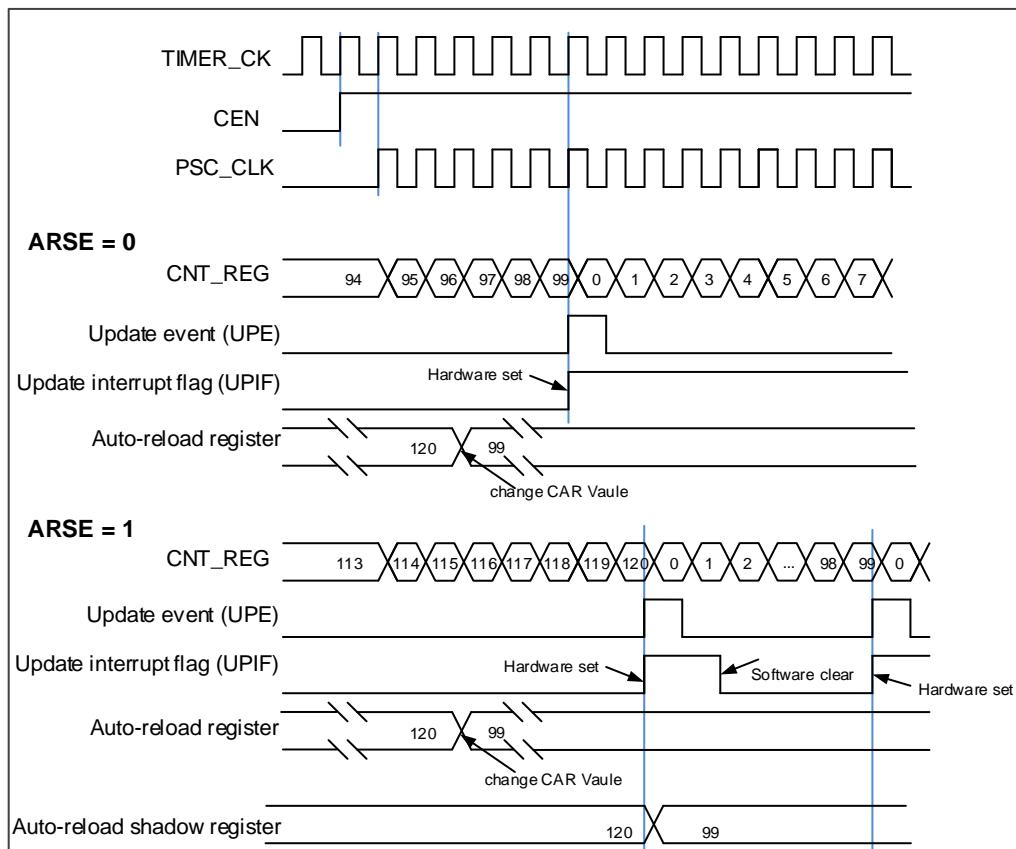


Figure 16-5. Timing chart of up counting mode, change TIMERx\_CAR on the go



### Counter down counting

In this mode, the counter counts down continuously from the counter-reload value, which is defined in the **TIMERx\_CAR** register, to 0 in a count-down direction. Once the counter reaches to 0, the counter will start counting down from the counter-reload value again and an underflow event will be generated. In addition, the update event will be generated after  $(\text{TIMERx_CREP}+1)$  times of underflow. The counting direction bit **DIR** in the **TIMERx\_CTL0** register should be set to 1 for the down-counting mode.

When the update event is set by the **UPG** bit in the **TIMERx\_SWEVG** register, the counter value will be initialized to the counter-reload value and generates an update event.

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

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

[\*\*Figure 16-6. Timing chart of down counting mode, PSC=0/2\*\*](#) show some examples of the

counter behavior in different clock frequencies when TIMERx\_CAR=0x99.

**Figure 16-6. Timing chart of down counting mode, PSC=0/2**

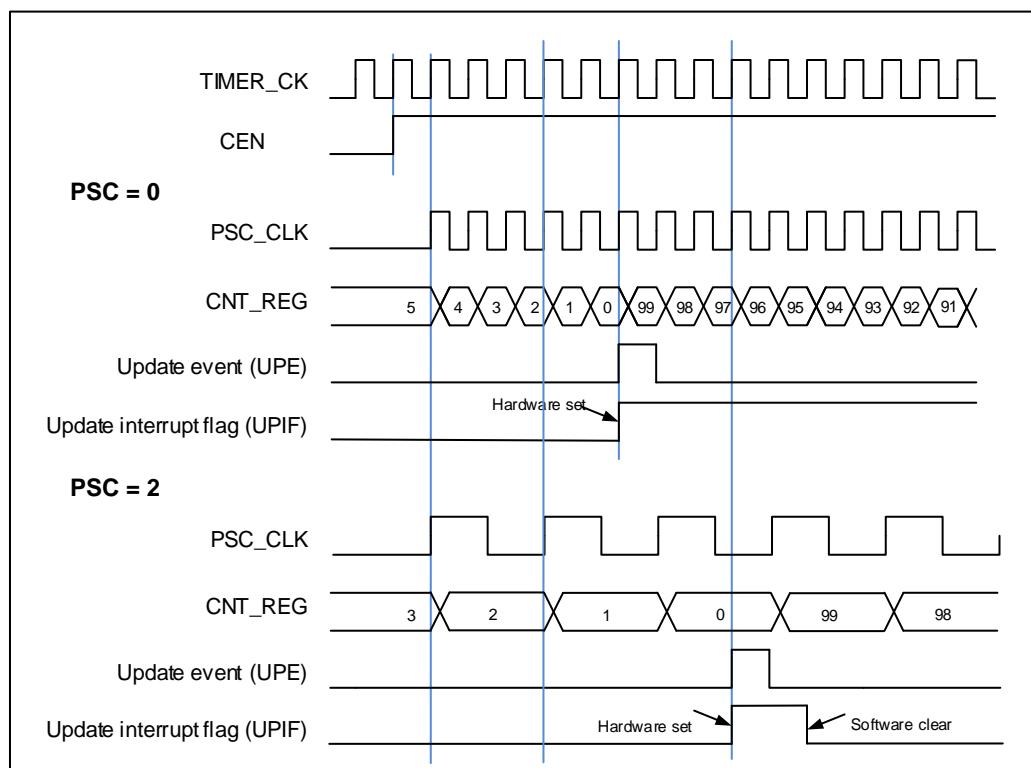
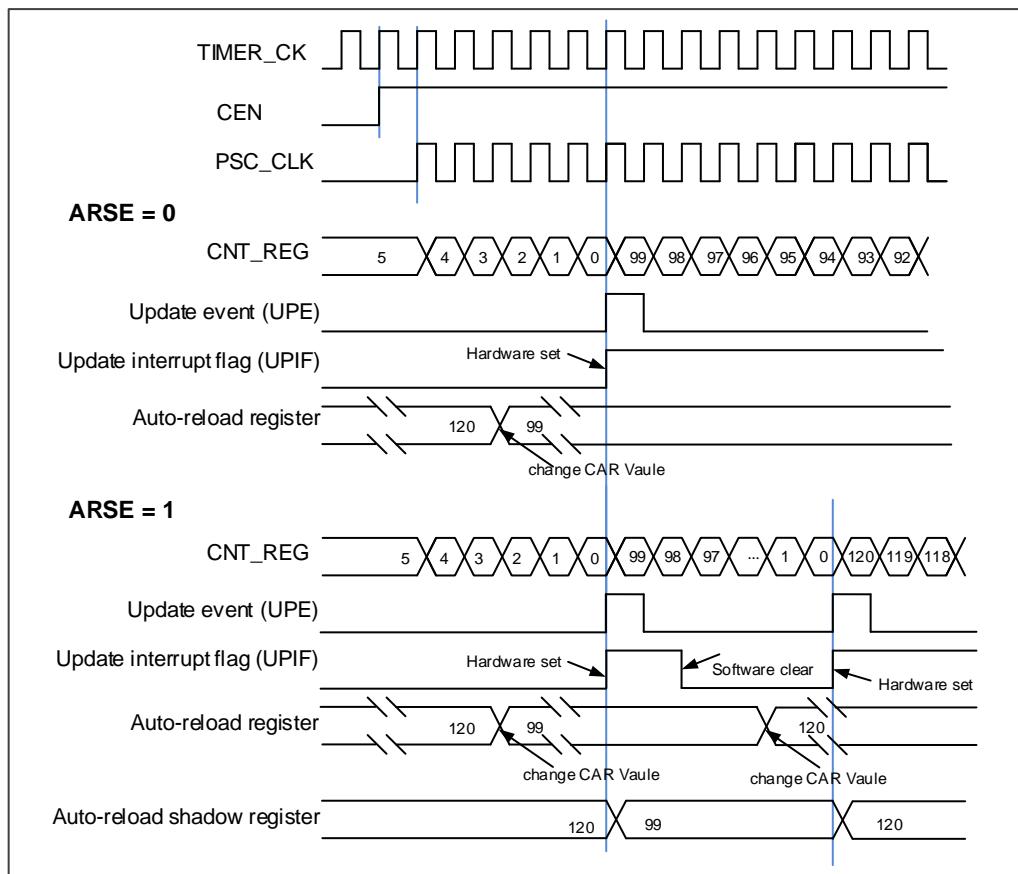


Figure 16-7. Timing chart of down counting mode, change TIMERx\_CAR on the go



### Counter center-aligned counting

In this mode, the counter counts up from 0 to the counter-reload value and then counts down to 0 alternatively. The Timer module generates an overflow event when the counter counts to the counter-reload value subtract 1 in the up-counting direction and generates an underflow event when the counter counts to 1 in the down-counting direction. The counting direction bit DIR in the TIMERx\_CTL0 register is read-only and indicates the counting direction when in the center-aligned mode.

Setting the UPG bit in the TIMERx\_SWEVG register will initialize the counter value to 0 and generates an update event irrespective of whether the counter is counting up or down in the center-align counting mode.

The UPIF bit in the TIMERx\_INTF register can be set to 1 either when an underflow event or an overflow event occurs. While the CHxIF bit is associated with the value of CAM in TIMERx\_CTL0. The details refer to [Figure 16-8. Center-aligned counter timechart](#).

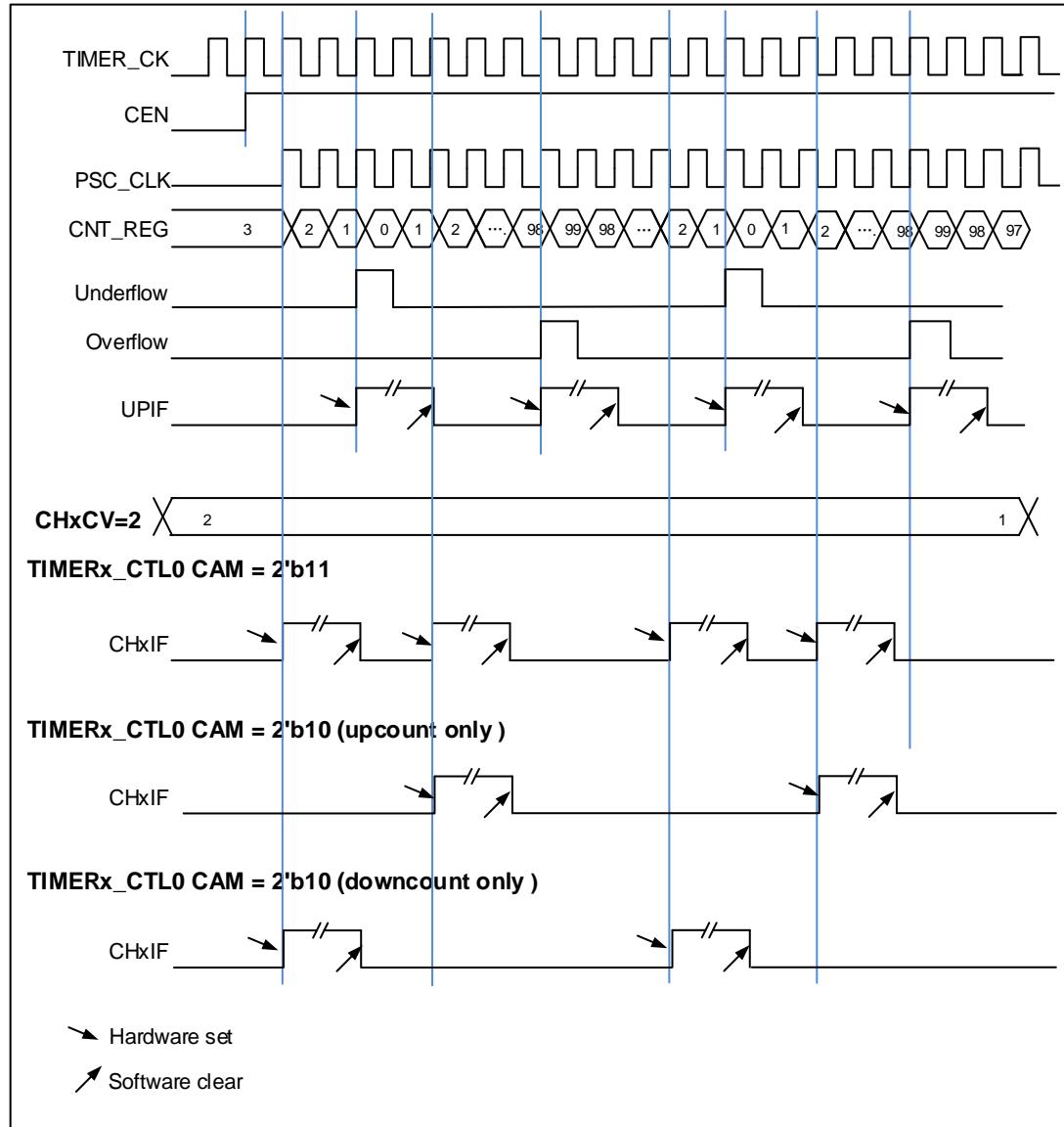
If set the UPDIS bit in the TIMERx\_CTL0 register, the update event is disabled.

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

[Figure 16-8. Center-aligned counter timechart](#) show some examples of the counter

behavior when TIMERx\_CAR=0x99. TIMERx\_PSC=0x0

**Figure 16-8. Center-aligned counter timechart**



#### Update event (from overflow/underflow) rate configuration

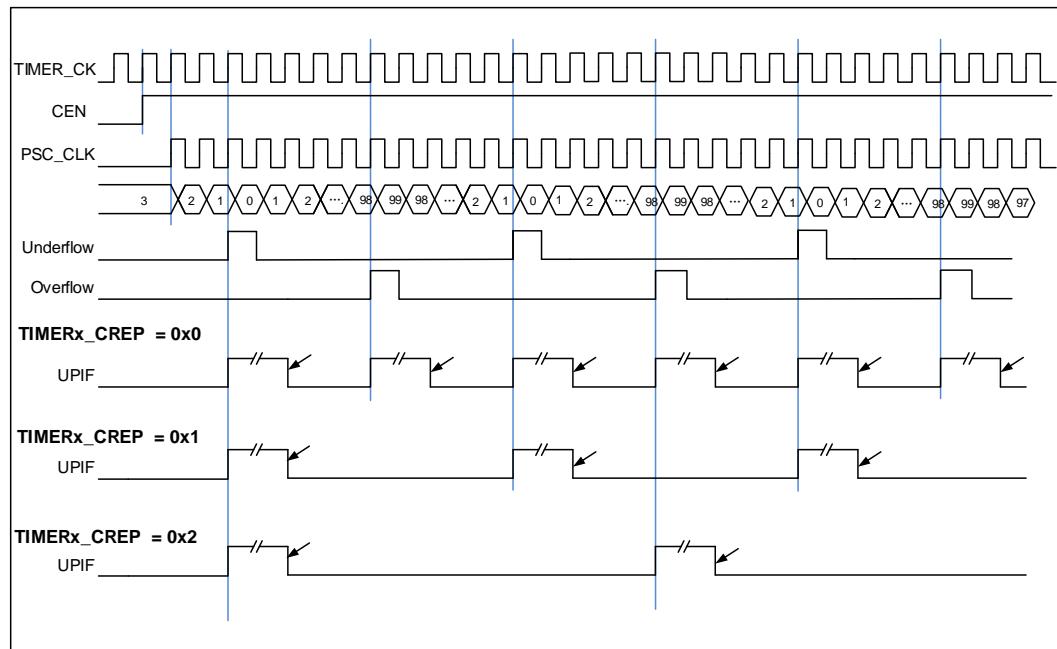
The rate of update events generation (from overflow and underflow events) can be configured by the TIMERx\_CREP register. Counter repetition is used to generate update event or updates the timer registers only after a given number ( $N+1$ ) of cycles of the counter, where  $N$  is CREP in TIMERx\_CREP register. The repetition counter is decremented at each counter overflow (does not exist in down counting mode) and underflow (does not exist in up counting mode).

Setting the UPG bit in the TIMERx\_SWEVG register will reload the content of CREP in TIMERx\_CREP register and generate an update event.

The new written CREP value will not take effect until the next update event. When the value of CREP is odd, and the counter is counting in center-aligned mode, the update event is

generated (on overflow or underflow) depending on when the written CREP value takes effect. If an update event is generated by software after writing an odd number to CREP, the update events will be generated on the underflow. If the next update event occurs on overflow after writing an odd number to CREP, then the subsequent update events will be generated on the overflow.

**Figure 16-9. Repetition timechart for center-aligned counter**



**Figure 16-10. Repetition timechart for up-counter**

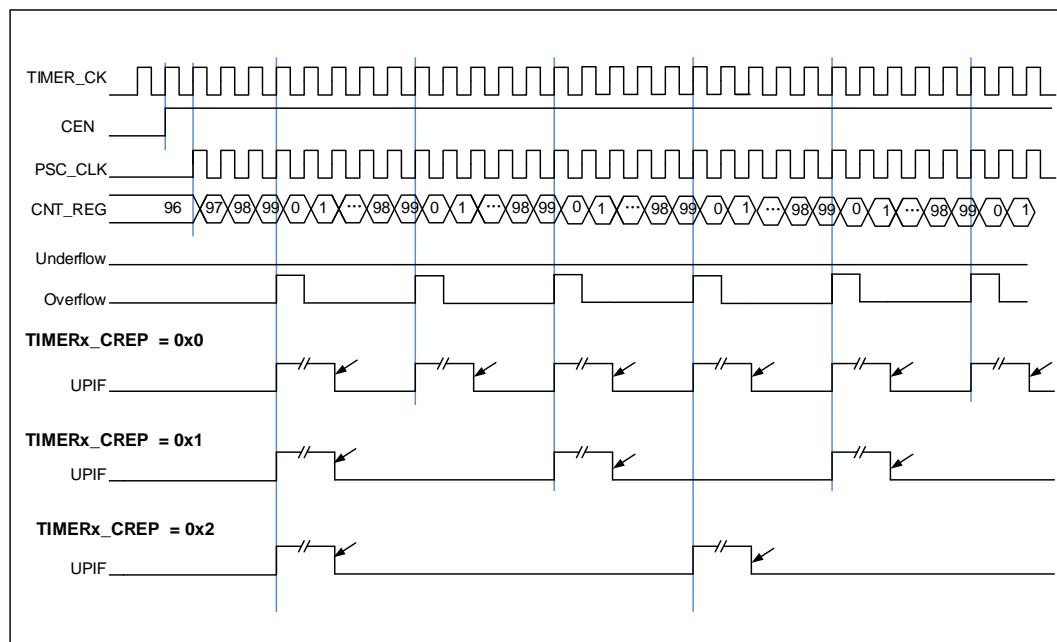
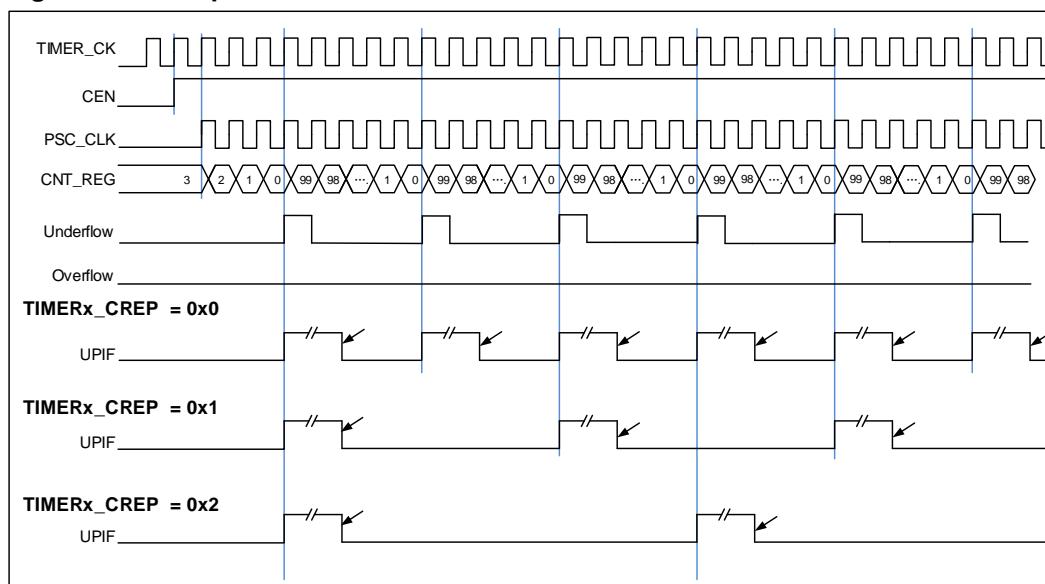


Figure 16-11. Repetition timechart for down-counter



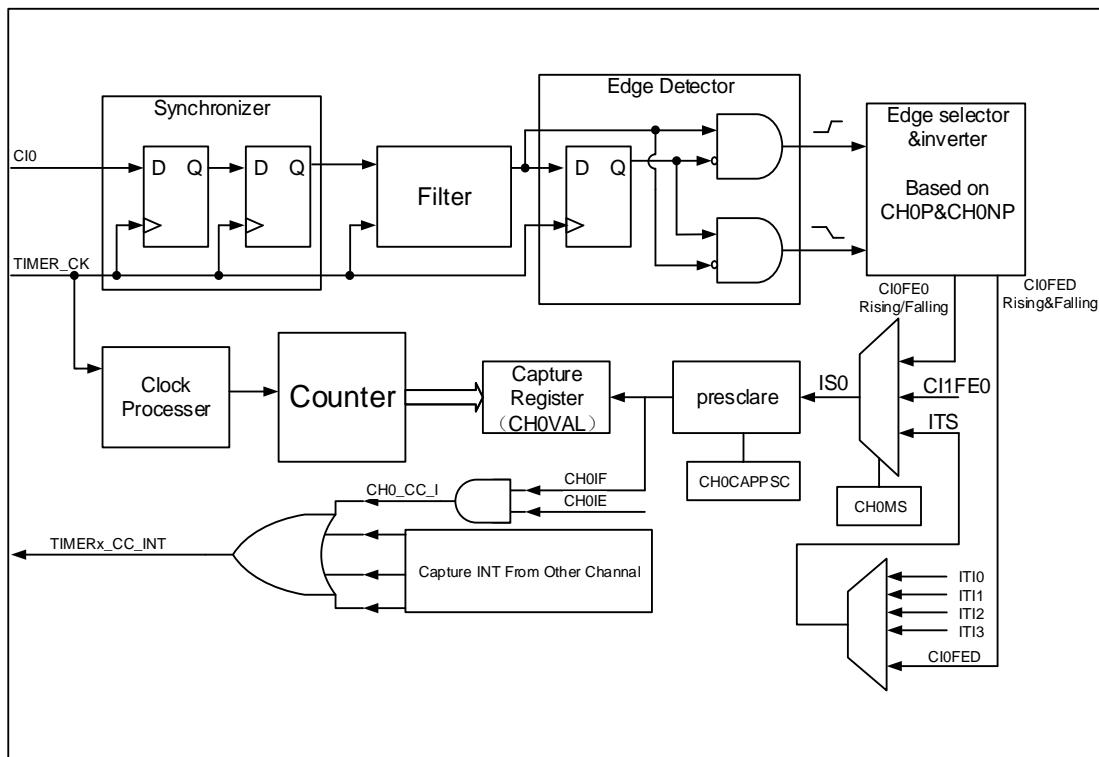
### Input capture and output compare channels

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

#### ■ Channel input capture function

Channel input capture function allows the channel to perform measurements such as pulse timing, frequency, period, duty cycle and so on. The input stage consists of a digital filter, a channel polarity selection, edge detection and a channel prescaler. When a selected edge occurs on the channel input, the current value of the counter is captured into the TIMERx\_CHxCV register, at the same time the CHxIF bit is set and the channel interrupt is generated if enabled by CHxIE = 1.

Figure 16-12. Channel input capture principle



One of channels' input signals (Clx) can be chosen from the TIMERx\_CHx signal or the Exclusive-OR function of the TIMERx\_CH0, TIMERx\_CH1 and TIMERx\_CH2 signals. First, the channel input signal (Clx) is synchronized to TIMER\_CK domain, and then sampled by a digital filter to generate a filtered input signal. Then through the edge detector, the rising and falling edge are detected. You can select one of them by CHxP. One more selector is for the other channel and trig, controlled by CHxMS. The IC\_prescaler make several the input event generate one effective capture event. On the capture event, TIMERx\_CHxCV will restore the value of counter.

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

**Step1:** Filter configuration. (CHxCAPFLT in TIMERx\_CHCTL0)

Based on the input signal and requested signal quality, configure compatible CHxCAPFLT.

**Step2:** Edge selection. (CHxP/CHxNP in TIMERx\_CHCTL2)

Rising or falling edge, choose one by CHxP/CHxNP.

**Step3:** Capture source selection. (CHxMS in TIMERx\_CHCTL0)

As soon as you select one input capture source by CHxMS, you have set the channel to input mode (CHxMS! =0x0) and TIMERx\_CHxCV cannot be written any more.

**Step4:** Interrupt enable. (CHxIE and CHxDEN in TIMERx\_DMAINTEN)

Enable the related interrupt enable; you can got the interrupt and DMA request.

**Step5:** Capture enables. (CHxEN in TIMERx\_CHCTL2)

**Result:** when you wanted input signal is got, TIMERx\_CHxCV will be set by counter's value. And CHxIF is asserted. If the CHxIF is high, the CHxOF will be asserted also. The interrupt and DMA request will be asserted based on the configuration of CHxIE and CHxDEN in TIMERx\_DMAINTEN

**Direct generation:** if you want to generate a DMA request or Interrupt, you can set CHxG by software directly.

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

## ■ Channel output compare function

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 is set and the channel (n) interrupt is generated if CHxIE = 1. And the DMA request will be assert, if CxCDE=1.

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

**Step1:** Clock Configuration. Such as clock source, clock prescaler and so on.

**Step2:** Compare mode configuration.

- \* Set the shadow enable mode by CHxCOMSEN
- \* Set the output mode (Set/Clear/Toggle) by CHxCOMCTL.
- \* Select the active high polarity by CHxP/CHxNP
- \* Enable the output by CHxEN

**Step3:** Interrupt/DMA-request enables configuration by CHxIE/CxCDE

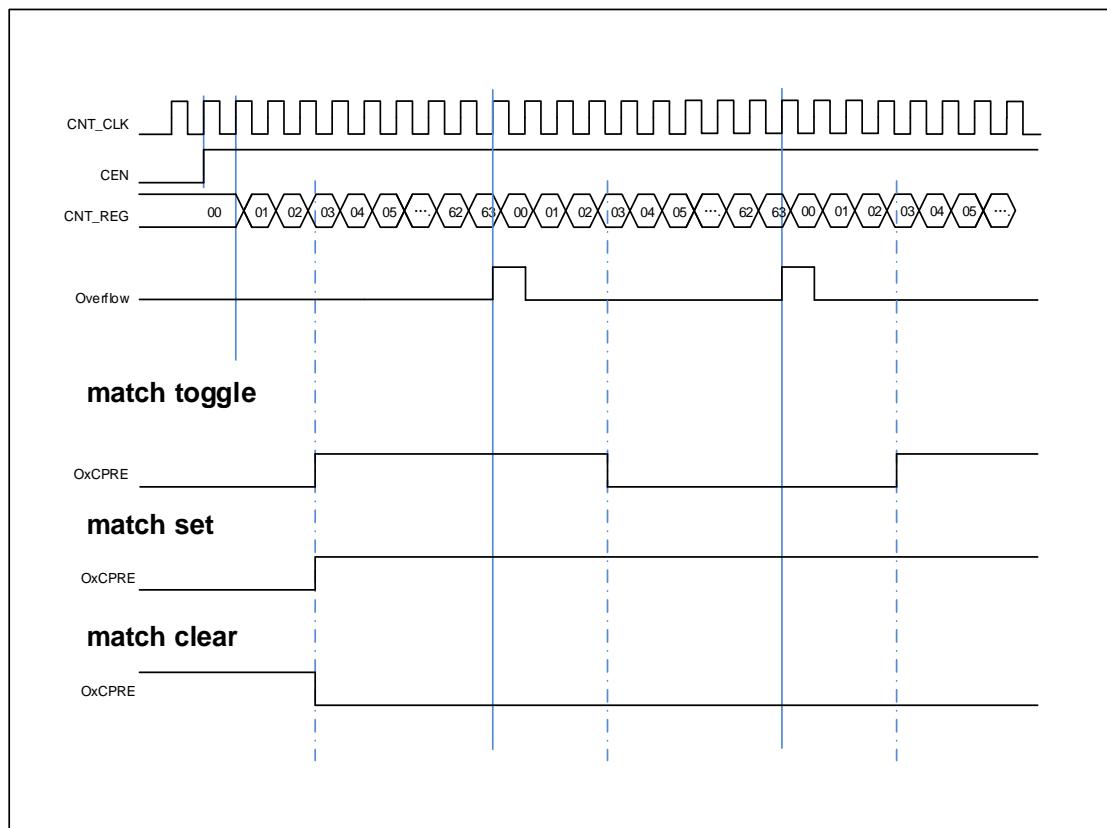
**Step4:** Compare output timing configuration by TIMERx\_CAR and TIMERx\_CHxCV

About the TIMERx\_CHxCV; you can change it on the go to meet the waveform you expected.

**Step5:** Start the counter by CEN.

The timechart below show the three compare modes toggle/set/clear. CAR=0x63, CHxVAL=0x3

Figure 16-13. Output-compare under three modes



### Output PWM function

In the output PWM function (by setting the CHxCOMCTL bits to 3'b110 (PWM mode0) or to 3'b 111(PWM mode1), the channel can generate PWM waveform according to the TIMERx\_CAR registers and TIMERx\_CHxCV registers.

Based on the counter mode, we can also divide PWM into EAPWM (Edge aligned PWM) and CAPWM (Centre aligned PWM).

The EAPWM period is determined by TIMERx\_CAR and duty cycle is determined by TIMERx\_CHxCV. [Figure 16-14. EAPWM timechart](#) shows the EAPWM output and interrupts waveform.

The CAPWM period is determined by 2\*TIMERx\_CAR, and duty cycle is by 2\*TIMERx\_CHxCV. [Figure 16-15. CAPWM timechart](#) shows the CAPWM output and interrupts waveform.

If TIMERx\_CHxCV is greater than TIMERx\_CAR, the output will be always active under PWM mode0 (CHxCOMCTL==3'b110).

And if TIMERx\_CHxCV is equal to zero, the output will be always inactive under PWM mode0 (CHxCOMCTL==3'b110).

Figure 16-14. EAPWM timechart

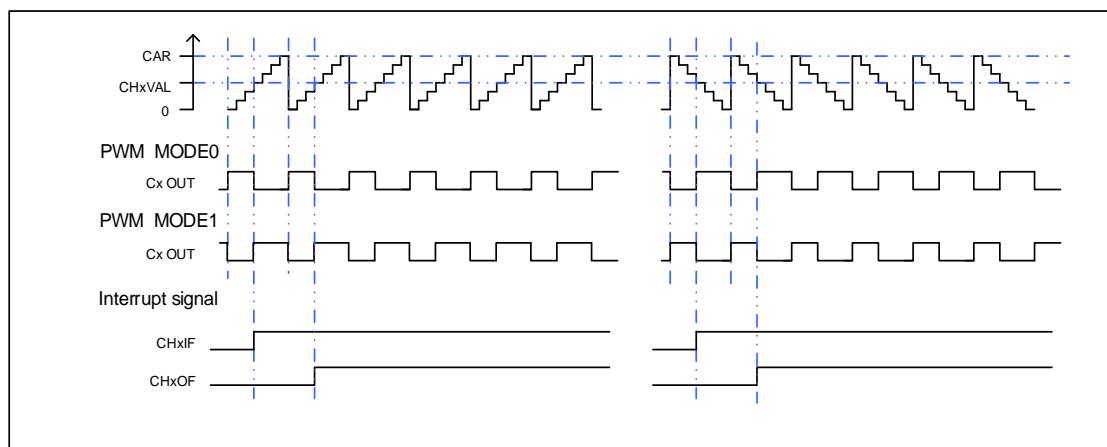
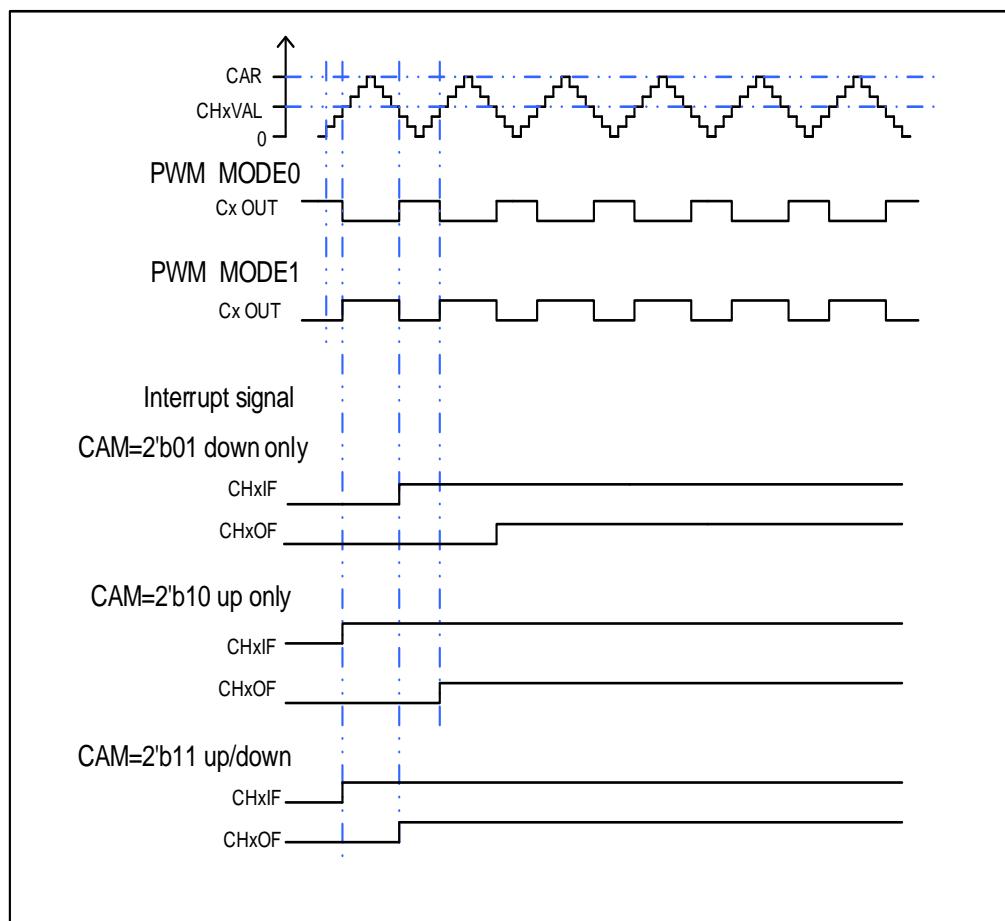


Figure 16-15. CAPWM timechart



### 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 ETIFE signal is derived from the external ETI pin and when it is set to a high level by setting the CHxCOMCEN bit to 1 in the TIMERx\_CHCTL0 register. The OxCPRE signal will not return to its active level until the next update event occurs.

### Channel output complementary PWM

Function of complementary is for a pair of CHx\_O and CHx\_ON. Those two output signals cannot be active at the same time. The TIMERx has 4 channels, but only the first three channels have this function. The complementary signals CHx\_O and CHx\_ON are controlled by a group of parameters: the CHxEN and CHxNEN bits in the TIMERx\_CHCTL2 register and the POEN, ROS, IOS, ISOx and ISOxN bits in the TIMERx\_CCHP and TIMERx\_CTL1 registers. The outputs polarity is determined by CHxP and CHxNP bits in the TIMERx\_CHCTL2 register.

**Table 16-2. 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.	
			1	0	CHx_O = CHxP CHx_ON = CHxNP CHx_O/CHx_ON output disable.	
				1	If clock is enable: CHx_O = ISOx CHx_ON = ISOxN	
		1	0	0	CHx_O = CHxP CHx_ON = CHxNP CHx_O/CHx_ON output disable.	
				1	CHx_O = CHxP CHx_ON = CHxNP CHx_O/CHx_ON output enable.	
			1	0	CHx_O = CHxP CHx_ON = CHxNP CHx_O/CHx_ON output enable.	
				1	If clock is enable: CHx_O = ISOx CHx_ON = ISOxN	
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⊕CHxNP CHx_ON output enable
			1	0	CHx_O=OxCPRE⊕CHxP CHx_O output enable	CHx_ON = LOW CHx_ON output disable.
				1	CHx_O=OxCPRE⊕CHxP CHx_O output enable	CHx_ON=(!OxCPRE)⊕CHxNP CHx_ON output enable
		1	0	0	CHx_O = CHxP CHx_O output disable.	CHx_ON = CHxNP CHx_ON output disable.
				1	CHx_O = CHxP CHx_O output enable	CHx_ON=OxCPRE⊕CHxNP CHx_ON output enable
			1	0	CHx_O=OxCPRE⊕CHxP CHx_O output enable	CHx_ON = CHxNP CHx_ON output enable.
				1	CHx_O=OxCPRE⊕CHxP CHx_O output enable	CHx_ON=(!OxCPRE)⊕CHxNP CHx_ON output enable.

### Insertion dead time for complementary PWM

The dead time insertion is enabled when both CHxEN and CHxNEN are 1'b1, and set POEN is also necessary. The field named DTCFG defines the dead time delay that can be used for all channels except for channel 3. The detail about the delay time, refer to the register TIMERx\_CCHP.

The dead time delay insertion ensures that no two complementary signals drive the active state at the same time.

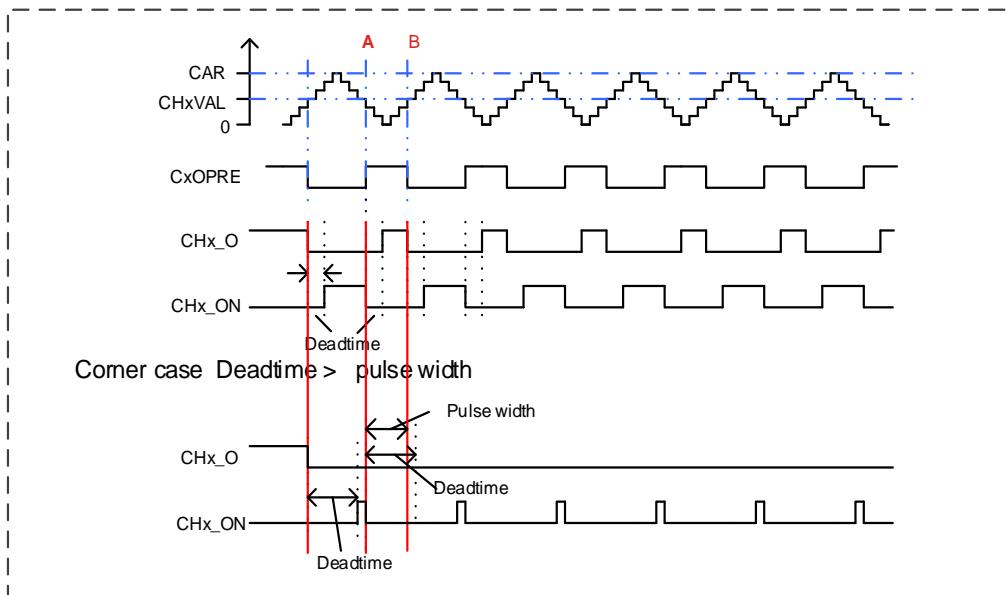
When the channel (x) match (TIMERx counter = CHxVAL) occurs, OxCPRE will be toggled because under PWM0 mode. At point A in the [Figure 16-16. Complementary output with dead-time insertion](#), CHx\_O signal remains at the low value until the end of the deadtime delay, while CHx\_ON will be cleared at once. Similarly, At point B when counter match (counter = CHxVAL) occurs again, OxCPRE is cleared, CHx\_O signal will be cleared at once, while CHx\_ON signal remains at the low value until the end of the dead time delay.

Sometimes, we can see corner cases about the dead time insertion. For example:

The dead time delay is greater than or equal to the CHx\_O duty cycle, then the CHx\_O signal is always the inactive value. (As show in the [Figure 16-16. Complementary output with dead-time insertion](#).)

- The dead time delay is greater than or equal to the CHx\_ON duty cycle, then the CHx\_ON signal is always the inactive value.

**Figure 16-16. Complementary output with dead-time insertion.**



### Break mode

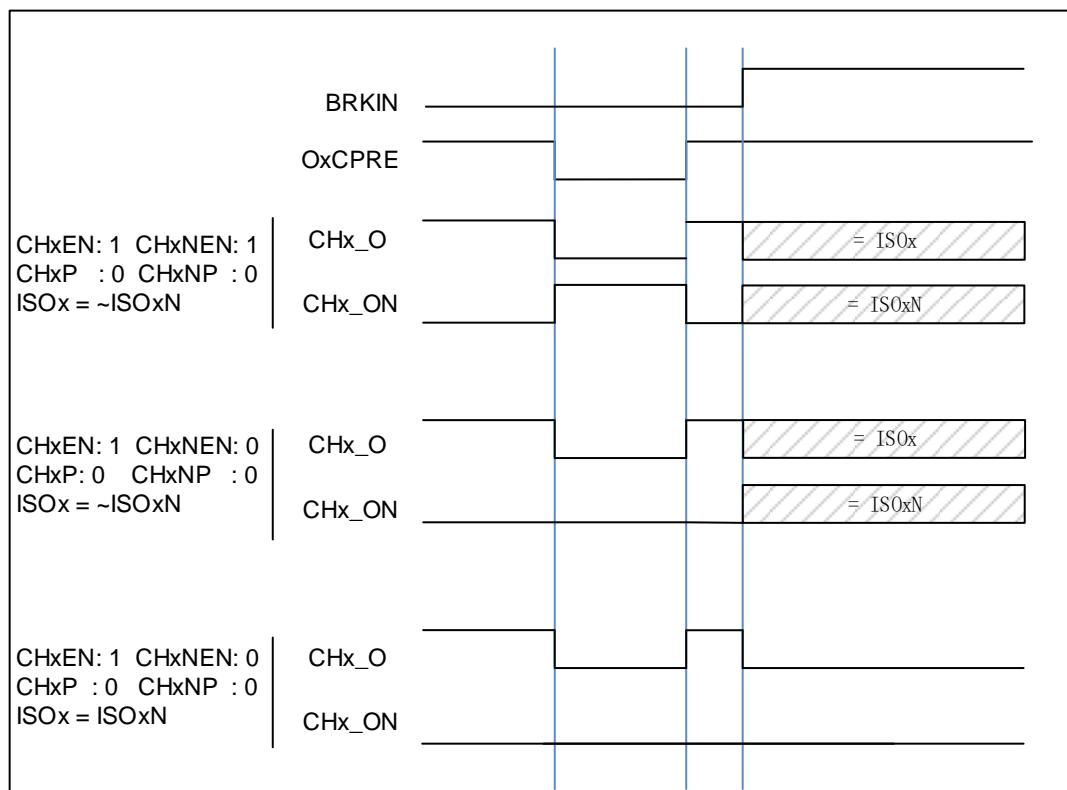
In this mode, the output CHx\_O and CHx\_ON are controlled by the POEN, IOS and ROS bits in the TIMERx\_CCHP register, ISOx and ISOxN bits in the TIMERx\_CTL1 register and cannot be set both to active level when break occurs. The break sources are input break pin and

HXTAL stuck event by Clock Monitor (CKM) in RCU. The break function enabled by setting the BRKEN bit in the TIMERx\_CCHP register. The break input polarity is setting by the BRKP bit in TIMERx\_CCHP.

When a break occurs, the POEN bit is cleared asynchronously, the output CHx\_O and CHx\_ON are driven with the level programmed in the ISOx bit and ISOxN in the TIMERx\_CTL1 register as soon as POEN is 0. If IOS is 0 then the timer releases the enable output else the enable output remains high. The complementary outputs are first put in reset state, and then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the ISOx and ISOxN bits after a dead-time.

When a break occurs, the BRKIF bit in the TIMERx\_INTF register is set. If BRKIE is 1, an interrupt generated.

**Figure 16-17. Output behavior in response to a break(The break high active)**



### Quadrature decoder

The quadrature decoder function uses two quadrature inputs CI0FE0 and CI1FE1 derived from the TIMERx\_CH0 and TIMERx\_CH1 pins respectively to interact to control the counter value. The DIR bit is modified during each input source transition. The counter can be changed by the edges of CI0FE0 only, CI1FE1 only or both CI0FE0 and CI1FE1, the selection mode by setting the SMC[2:0] to 0x01, 0x02 or 0x03. The mechanism for changing the counter direction is shown in [Table 16-3. Counting direction in different quadrature](#)

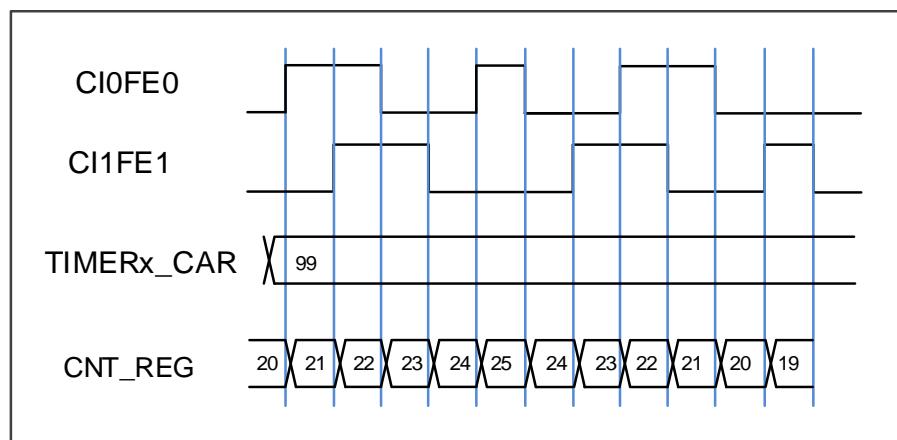
**decoder mode.** The quadrature decoder can be regarded as an external clock with a directional selection. This means that the counter counts continuously in the interval between 0 and the counter-period value. Therefore, TIMERx\_CAR register must be configured before the counter starts to count.

**Table 16-3. Counting direction in different quadrature decoder mode**

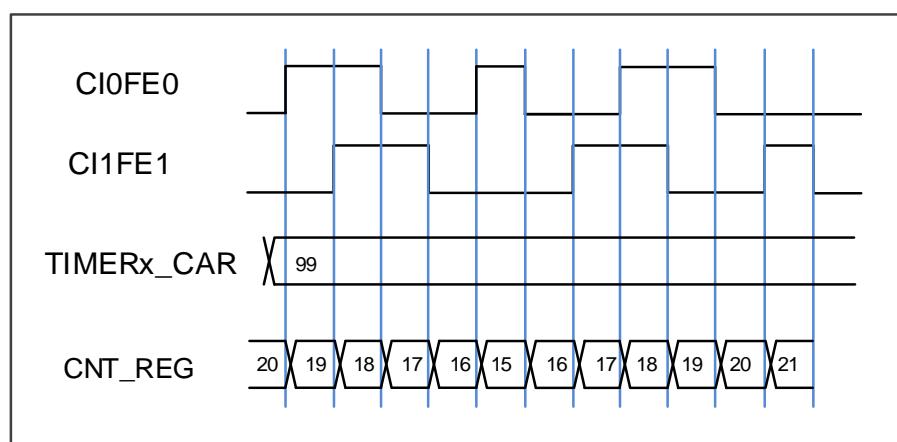
Counting mode	Level	CI0FE0		CI1FE1	
		Rising	Falling	Rising	Falling
Quadrature decoder mode 0 SMC[2:0]=3'b001	CI1FE1=1	Down	Up	-	-
	CI1FE1=0	Up	Down	-	-
Quadrature decoder mode 1 SMC [2:0]=3'b010	CI0FE0=1	-	-	Up	Down
	CI0FE0=0	-	-	Down	Up
Quadrature decoder mode 2 SMC [2:0]=3'b011	CI1FE1=1	Down	Up	X	X
	CI1FE1=0	Up	Down	X	X
	CI0FE0=1	X	X	Up	Down
	CI0FE0=0	X	X	Down	Up

**Note:** "-" means "no counting", "X" means impossible. "0" means "low level", "1" means "high level".

**Figure 16-18. Counter behavior with CI0FE0 polarity non-inverted in mode 2**



**Figure 16-19. Counter behavior with CI0FE0 polarity inverted in mode 2**



## Hall sensor function

Hall sensor is generally used to control BLDC Motor; the timers can support this function.

[\*\*Figure 16-20. Hall sensor is used to BLDC motor\*\*](#) show how to connect. And we can see we need two timers. First TIMER\_in (Advanced/General L0 TIMER) should accept three HALL sensor signals.

Each of the three input of HALL 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/General L0 TIMER.

And TIMER\_out need have functions of complementary and Dead-time, so only advanced timer can be chosen. Else, based on the timers' internal connection relationship, pair's timers can be selected. For example:

TIMER\_in (TIMER0) -> TIMER\_out (TIMER7 ITI0)

TIMER\_in (TIMER1) -> TIMER\_out (TIMER0 ITI1)

And so on.

After getting appropriate timers combination, and wire connection, we need to configure timers. Some key settings include:

- Enable XOR by setting TI0S, then, each of input signal change will make the CI0 toggle. CH0VAL will record the value of counter at that moment.
- Enable ITIx connected to commutation function directly by setting CCUC and CCSE.
- Configuration PWM parameter based on your request.

**Figure 16-20. Hall sensor is used to BLDC motor**

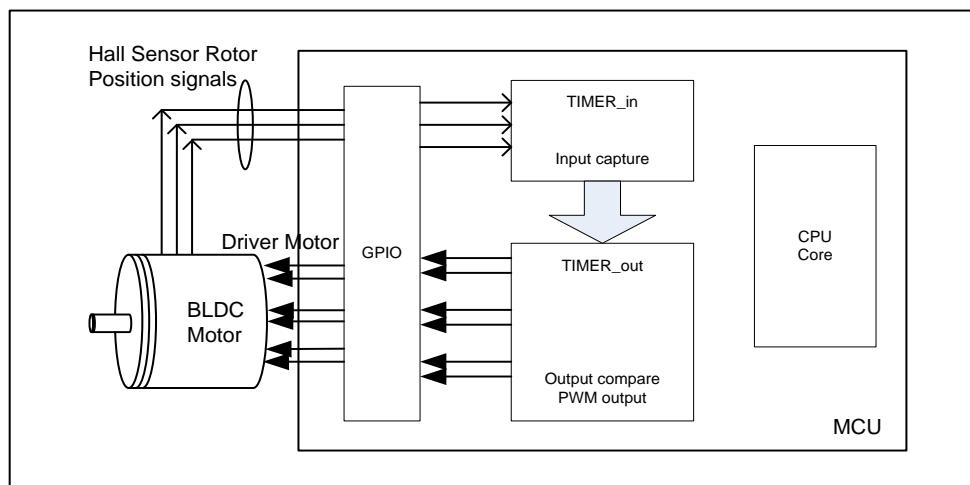
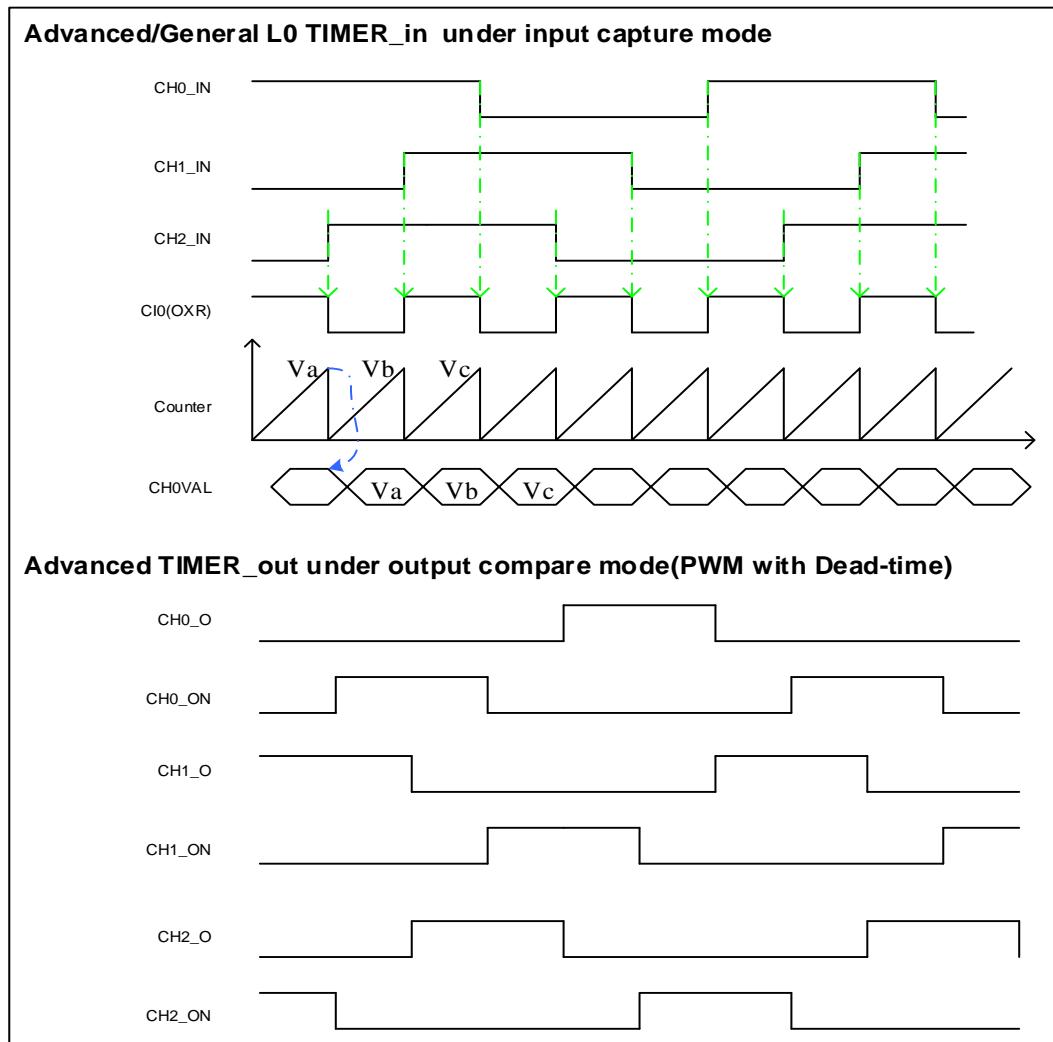


Figure 16-21. Hall sensor timing between two timers



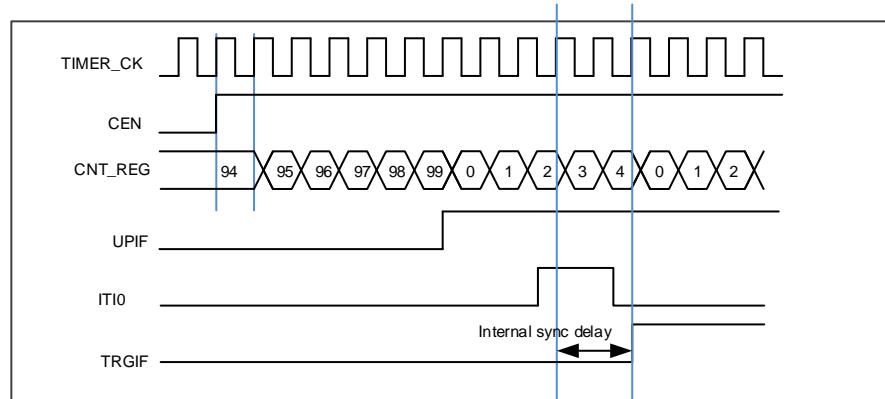
### Master-slave management

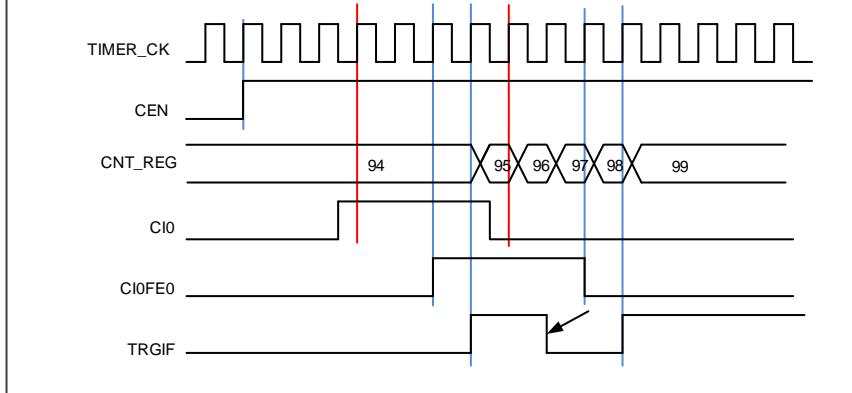
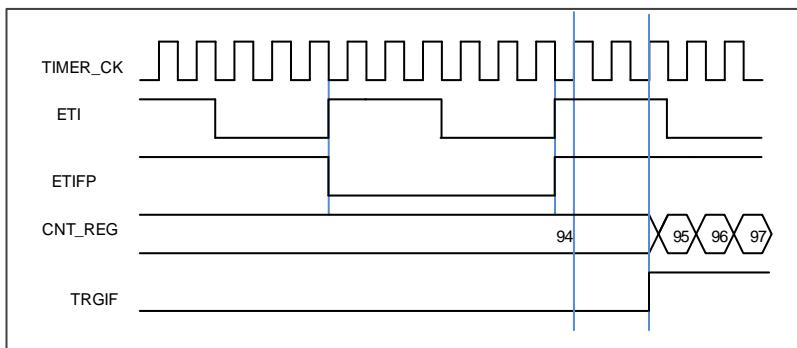
The TIMERx can be synchronized with a trigger in several modes including the restart mode, the pause mode and the event mode which is selected by the SMC[2:0] in the TIMERx\_SMCFG register. The trigger input of these modes can be selected by the TRGS [2:0] in the TIMERx\_SMCFG register.

Table 16-4. Examples of slave mode

	Mode Selection	Source Selection	Polarity Selection	Filter and Prescaler
LIST	SMC[2:0] 3'b100 (restart mode) 3'b101 (pause mode) 3'b110 (event mode)	TRGS[2:0] 000: ITI0 001: ITI1 010: ITI2 011: ITI3 100: CI0F_ED 101: CI0FE0 110: CI1FE1 111: 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.
Exam1	Restart mode  The counter can be clear and restart when a rising trigger input.	TRGS[2:0]=3'b000  ITI0 is the selection.	-  For ITI0, no polarity selector can be used.	-  For the ITI0, no filter and prescaler can be used.
Exam2	Pause mode  The counter can be paused when the trigger input is low.	TRGS[2:0]=3'b101  CI0FE0 is the selection.	TI0S=0. (Non-xor) [CH0NP==0, CH0P==0] no inverted. Capture will be sensitive to the rising edge only.	Filter is bypass in this example.

Figure 16-22. Restart mode



	Mode Selection	Source Selection	Polarity Selection	Filter and Prescaler
	<b>Figure 16-23. Pause mode</b>			
		 <p>The diagram illustrates the timing sequence for Pause mode. The <b>TIMER_CK</b> signal is a square wave. The <b>CEN</b> signal is asserted at the start of the sequence. The <b>CNT_REG</b> counter starts at 94 and increments to 99. The <b>CI0</b> signal is asserted during the count from 95 to 98. The <b>CI0FE0</b> signal is asserted during the count from 95 to 98. The <b>TRGIF</b> signal is asserted at the end of the count sequence.</p>		
Exam3	Event mode The counter will start to count when a rising trigger input.	TRGS[2:0]=3'b11 1 ETIF is the selection.	ETP = 0 no polarity change. ETPSC = 1, divided by 2.	ETFC = 0 , no filter
	<b>Figure 16-24. Event mode</b>			
		 <p>The diagram illustrates the timing sequence for Event mode. The <b>TIMER_CK</b> signal is a square wave. The <b>ETI</b> signal triggers the counter. The <b>ETIFFP</b> signal is asserted during the count from 95 to 97. The <b>CNT_REG</b> counter starts at 94 and increments to 97. The <b>TRGIF</b> signal is asserted at the end of the count sequence.</p>		

### Single pulse mode

Single pulse mode is opposite to the repetitive mode, which can be enabled by setting SPM in **TIMERx\_CTL0**. When you set SPM, the counter will be clear and stop when the next update event. In order to get pulse waveform, you can set the **TIMERx** to PWM mode or compare by **CHxCOMCTL**.

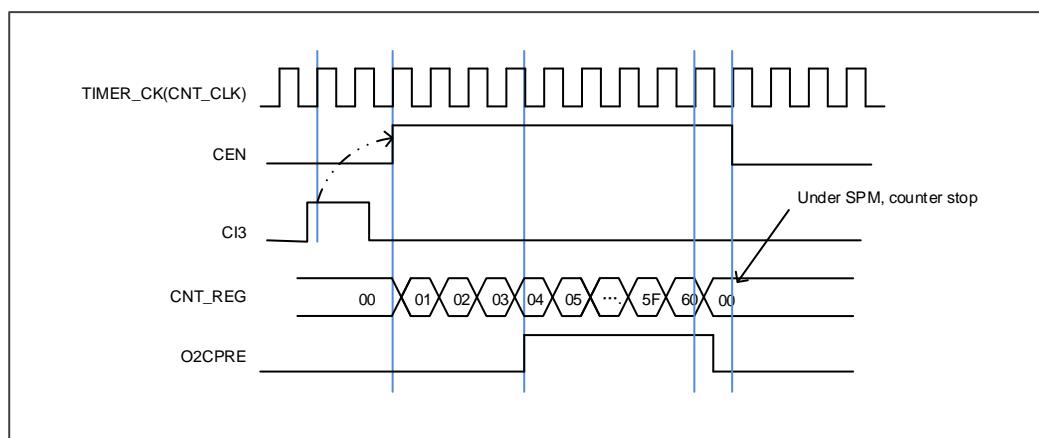
Once the timer is set to operate in the single pulse mode, it is not necessary to set the timer enable bit **CEN** in the **TIMERx\_CTL0** register to 1 to enable the counter. The trigger to generate a pulse can be sourced from the trigger signals edge or by setting the **CEN** bit to 1 using software. Setting the **CEN** bit to 1 or a trigger from the trigger signals edge can generate a pulse and then keep the **CEN** bit at a high state until the update event occurs or the **CEN** bit is written to 0 by software. If the **CEN** bit is cleared to 0 using software, the counter will be

stopped and its value held.

In the single pulse mode, the trigger active edge which sets the CEN bit to 1 will enable the counter. However, there exist several clock delays to perform the comparison result between the counter value and the TIMER<sub>x</sub>\_CHxCV value. In order to reduce the delay to a minimum value, the user can set the CHxCOMFEN bit in each TIMER<sub>x</sub>\_CHCTL0/1 register. After a trigger rising occurs in the single pulse mode, the OxCPRE signal will immediately be forced to the state which the OxCPRE signal will change to, as the compare match event occurs without taking the comparison result into account. The CHxCOMFEN bit is available only when the output channel is configured to operate in the PWM0 or PWM1 output mode and the trigger source is derived from the trigger signal.

[\*\*Figure 16-25. Single pulse mode TIMER<sub>x</sub>\\_CHxCV = 4 TIMER<sub>x</sub>\\_CAR=99\*\*](#) shows an example.

**Figure 16-25. Single pulse mode TIMER<sub>x</sub>\_CHxCV = 4 TIMER<sub>x</sub>\_CAR=99**

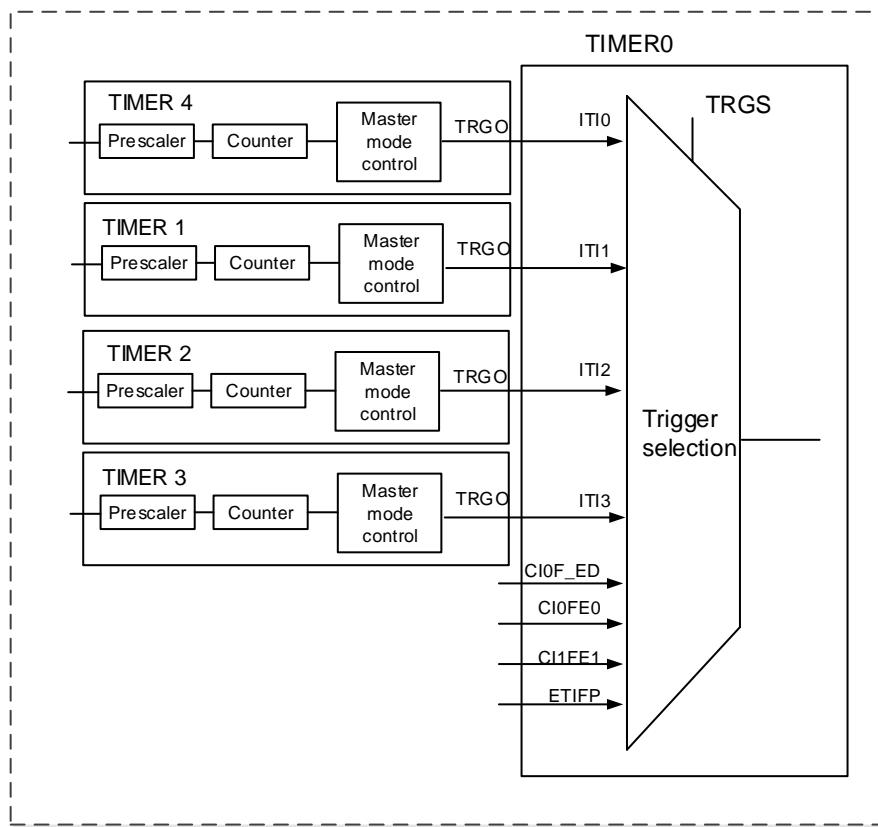


### Timers interconnection

Timer can be configured as interconnection, that is, one timer which operate in the master mode outputs TRGO signal to control another timer which operate in the slave mode, TRGO include reset event, start event, update event, capture/compare pulse event, compare event. slave timer received the ITIx and performs the corresponding mode, include internal clock mode, quadrature decoder mode, restart mode, pause mode, event mode, external clock mode.

[\*\*Figure 16-26. Timer0 master/slave mode timer example\*\*](#) shows the timer0 trigger selection when it is configured in slave mode.

**Figure 16-26. Timer0 master/slave mode timer example**



Other interconnection examples:

- Timer 2 as prescaler for timer 0

We configure Timer2 as a prescaler for Timer 0. Refer to [\*\*Figure 16-26. Timer0 master/slave mode timer example\*\*](#) for connections. Do as bellow:

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 TIMERx\_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 timer 0 with timer 2's Enable/Update signal

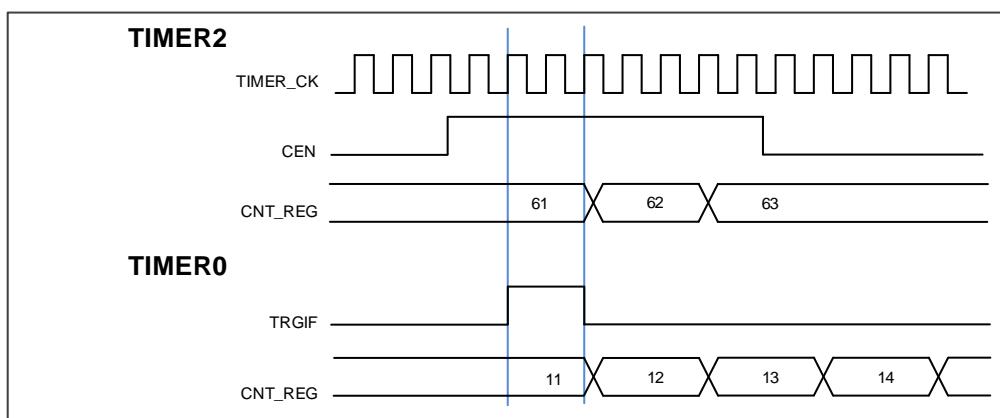
First, we enable Timer0 with the enable out of Timer2. Refer to [\*\*Figure 16-27. Triggering TIMER0 with enable signal of TIMER2\*\*](#). Timer0 starts counting from its current value on the

divided internal clock after trigger by timer2 enable output.

When Timer0 receives the trigger signal its CEN bit is 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 ( $SMC=3'b110$  in `TIMERx_SMCFG` register).
4. Start Timer2 by writing 1 in the CEN bit (`TIMER2_CTL0` register).

**Figure 16-27. Triggering TIMER0 with enable signal of TIMER2**



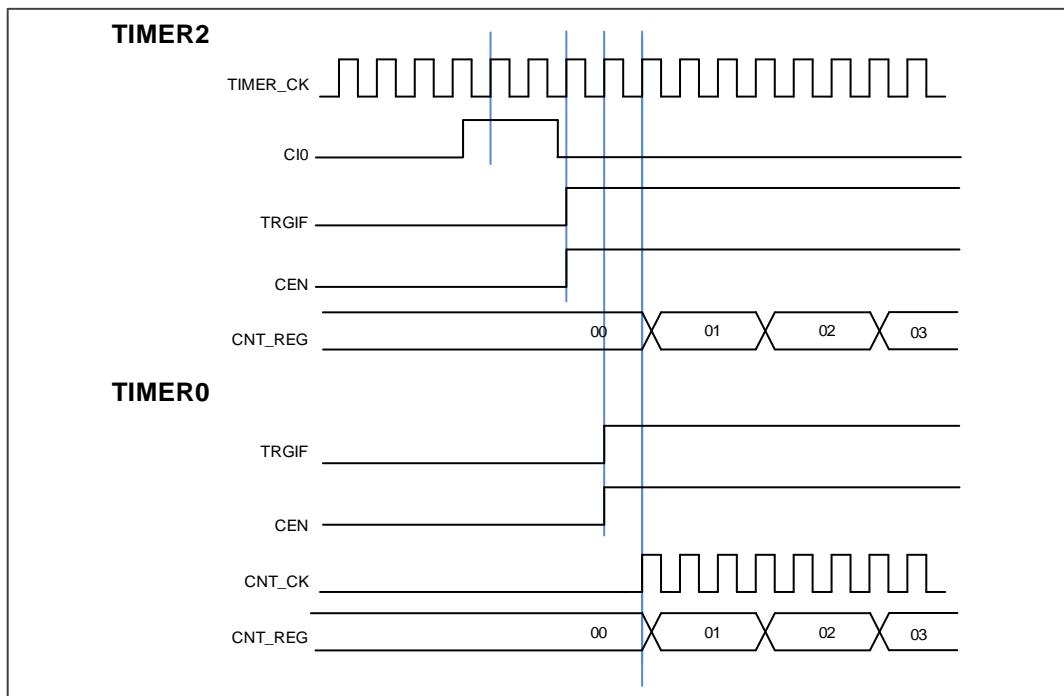
#### ■ Using an external trigger to start 2 timers synchronously

We configure the start of Timer0 is triggered by the enable 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 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 `TIMERx_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 16-28. Triggering TIMER0 and TIMER2 with TIMER2's CI0 input



### Timer DMA mode

Timer's DMA mode is the function that configures timer's register by DMA module. The relative registers are `TIMERx_DMACFG` and `TIMERx_DMATB`. Of course, you have to enable a DMA request which will be asserted by some internal event. When the interrupt event was asserted, `TIMERx` will send a request to DMA, which is configured to M2P mode and PADDR is `TIMERx_DMATB`, then DMA will access the `TIMERx_DMATB`. In fact, register `TIMERx_DMATB` is only a buffer; timer will map the `TIMERx_DMATB` to an internal register, appointed by the field of `DMATA` in `TIMERx_DMACFG`. If the field of `DMATC` in `TIMERx_DMACFG` is 0(1 transfer), then the timer's DMA request is finished. While if `TIMERx_DMATC` is not 0, such as 3( 4 transfers), then timer will send 3 more requests to DMA, and DMA will access timer's registers `DMATA+0x4`, `DMATA+0x8`, `DMATA+0xc` at the next 3 accesses to `TIMERx_DMATB`. In one word, one time DMA internal interrupt event assert, `DMATC+1` times request will be send by `TIMERx`.

If one more time DMA request event coming, `TIMERx` will repeat the process as above.

### Timer debug mode

When the Cortex®-M4 halted, and the `TIMERx_HOLD` configuration bit in `DBG_CTL0` register is set to 1, the `TIMERx` counter stops.

### 16.1.5. TIMERx registers(x=0, 7)

TIMER0 base address: 0x4001 2C00

TIMER7 base address: 0x4001 3400

#### 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 factor between the CK_TIMER and the dead-time and digital filter sample clock (DTS). 00: $f_{DTS}=f_{CK\_TIMER}$ 01: $f_{DTS}= f_{CK\_TIMER} /2$ 10: $f_{DTS}= f_{CK\_TIMER} /4$ 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 counting down, CHxF bit 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 counting up, CHxF bit 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 counting up and counting down, CHxF bit 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 If the timer work in center-aligned mode or encoder mode, this bit is read only.
3	SPM	Single pulse mode. 0: Single pulse mode disable. The counter continues after update event. 1: Single pulse mode enable. The counter counts until the next update event occurs.
2	UPS	Update source This bit is used to select the update event sources by software. 0: These events generate update interrupts or DMA requests: The UPG bit is set The counter generates an overflow or underflow event The restart mode generates an update event. 1: This event generates update interrupts or DMA requests: The counter generates an overflow or underflow event
1	UPDIS	Update disable. This bit is used to enable or disable the update event generation. 0: Update event enable. When an update event occurs, the corresponding shadow registers are loaded with their preloaded values. These events generate update event: The UPG bit is set The counter generates an overflow or underflow event The restart mode generates an update event. 1: Update event disable. <b>Note:</b> When this bit is set to 1, setting UPG bit or the restart mode does not generate an update event, but the counter and prescaler are initialized.
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.

### Control register 1 (TIMERx\_CTL1)

Address offset: 0x04

Reset value: 0x0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	ISO3	ISO2N	ISO2	ISO1N	ISO1	ISO0N	ISO0	TIO5		MMC[2:0]		DMAS	CCUC	Reserved	CCSE

<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: When a counter reset event occurs, a TRGO trigger signal is output. The counter reset source: Master timer generate a reset the UPG bit in the TIMERx_SWEVG register is set 001: Enable. When a counter start event occurs, a TRGO trigger signal is output. The counter start source : CEN control bit is set The trigger input in pause mode is high

010: When an update event occurs, a TRGO trigger signal is output. The update source depends on UPDIS bit and UPS bit.

011: When a capture or compare pulse event occurs in channel0, a TRGO trigger signal is output.

100: When a compare event occurs, a TRGO trigger signal is output. The compare source is from O0CPRE.

101: When a compare event occurs, a TRGO trigger signal is output. The compare source is from O1CPRE.

110: When a compare event occurs, a TRGO trigger signal is output. The compare source is from O2CPRE.

111: When a compare event occurs, a TRGO trigger signal is output. The compare source is from O3CPRE.

3	DMAS	DMA request source selection  0: When capture or compare event occurs, the DMA request of channel x is sent 1: When update event occurs, the DMA request of channel x is sent.
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 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
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

Bits	Fields	Descriptions											
31:16	Reserved	Must be kept at reset value											
15	ETP	<p>External trigger polarity</p> <p>This bit specifies the polarity of ETI signal</p> <p>0: ETI is active at rising edge or high level .</p> <p>1: ETI is active at falling edge or low level .</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>When the slave mode is configured as restart mode, pause mode or event mode, the timer can still work in the external clock 1 mode by setting this bit. But the TRGS bits must not be 3'b111 in this case.</p> <p>The clock source of the timer will be ETIFP if external clock mode 0 and external clock mode 1 are configured at the same time.</p> <p><b>Note:</b> External clock mode 0 enable is in this register's SMC[2:0] bit-field.</p>											
13:12	ETPSC[1:0]	<p>The prescaler of external trigger</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: The prescaler is 2.</p> <p>10: The prescaler is 4.</p> <p>11: The prescaler is 8.</p>											
11:8	ETFC[3:0]	<p>External trigger filter control</p> <p>The external trigger can be filtered by digital filter and this bit-field configure the filtering capability.</p> <p>Basic principle of digital filter: continuously sample the external trigger signal according to fsAMP and record the number of times of the same level of the signal. After reaching the filtering capacity configured by this bit-field, it is considered to be an effective level.</p> <p>The filtering capability configuration is as follows:</p> <table border="1"> <thead> <tr> <th>EXTFC[3:0]</th> <th>Times</th> <th>fsAMP</th> </tr> </thead> <tbody> <tr> <td>4'b0000</td> <td colspan="2">Filter disabled.</td></tr> <tr> <td>4'b0001</td> <td>2</td> <td rowspan="2">fCK_TIMER</td></tr> <tr> <td>4'b0010</td> <td>4</td></tr> </tbody> </table>	EXTFC[3:0]	Times	fsAMP	4'b0000	Filter disabled.		4'b0001	2	fCK_TIMER	4'b0010	4
EXTFC[3:0]	Times	fsAMP											
4'b0000	Filter disabled.												
4'b0001	2	fCK_TIMER											
4'b0010	4												

4'b0011	8	
4'b0100	6	$f_{DTS\_CK}/2$
4'b0101	8	
4'b0110	6	$f_{DTS\_CK}/4$
4'b0111	8	
4'b1000	6	$f_{DTS\_CK}/8$
4'b1001	8	
4'b1010	5	$f_{DTS\_CK}/16$
4'b1011	6	
4'b1100	8	
4'b1101	5	$f_{DTS\_CK}/32$
4'b1110	6	
4'b1111	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: ITI0
		001: ITI1
		010: ITI2
		011: ITI3
		100: CI0F_ED
		101: CI0FE0
		110: CI1FE1
		111: 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 CI0FE0 edge, while the direction depends on CI1FE1 level.
		010: Quadrature decoder mode 1. The counter counts on CI1FE1 edge, while the direction depends on CI0FE0 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 an update event is generated 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 clock when it is low.

110: Event mode. A rising edge of the trigger input enables the counter.

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	CH0IF	UPIF		

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	Reserved	Must be kept at reset value.
7	BRKIF	Break interrupt flag When the break input is inactive, the bit is set by hardware. When the break input is inactive, the bit can be cleared by software. 0: No active level break has been detected. 1: An active level has been detected.
6	TRGIF	Trigger interrupt flag This flag is set on trigger event and cleared by software. When in pause mode, both edges on trigger input generates a trigger event, otherwise, only an active edge on trigger input can generates a trigger event. 0: No trigger event occurred. 1: Trigger interrupt occurred.
5	CMTIF	Channel commutation interrupt flag This flag is set by hardware when channel's commutation event occurs, and cleared by software 0: No channel commutation interrupt occurred 1: Channel commutation interrupt occurred
4	CH3IF	Channel 3 's capture/compare interrupt flag Refer to CH0IF description
3	CH2IF	Channel 2 's capture/compare interrupt flag 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.

If Channel0 is set to input mode, this bit will be reset by reading TIMERx\_CH0CV.

0: No Channel 0 interrupt occurred

1: Channel 0 interrupt occurred

0	UPIF	Update interrupt flag  This bit is set by hardware on an update event and cleared by software. 0: No update interrupt occurred 1: Update interrupt occurred
---	------	---

## Software event generation register (TIMERx\_SWEVG)

Address offset: 0x14

Reset value: 0x0000

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

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	w	w	w	w	w	w	w

Bits	Fields	Descriptions
15:8	Reserved	Must be kept at reset value.
7	BRKG	Break event generation  This bit is set by software and cleared by hardware automatically. When this bit is set, the POEN bit is cleared and BRKIF flag is set, related interrupt or DMA transfer can occur if enabled. 0: No generate a break event 1: Generate a break event
6	TRGG	Trigger event generation  This bit is set by software and cleared by hardware automatically. When this bit is set, the TRGIF flag in TIMERx_INTF register is set, related interrupt or DMA transfer can occur if enabled. 0: No generate a trigger event 1: Generate a trigger event
5	CMTG	Channel commutation event generation  This bit is set by software and cleared by hardware automatically. When this bit is set, channel's capture/compare control registers (CHxEN, CHxNEN and CHxCOMCTL bits) are updated based on the value of CCSE (in the TIMERx_CTL1). 0: No affect 1: Generate channel's c/c control update event
4	CH3G	Channel 3's capture or compare event generation

		Refer to CH0G description
3	CH2G	Channel 2's capture or compare event generation Refer to CH0G description
2	CH1G	Channel 1's capture or compare event generation Refer to CH0G description
1	CH0G	Channel 0's capture or compare event generation This bit is set by software in order to generate a capture or compare event in channel 0, it is automatically cleared by hardware. When this bit is set, the CH0IF flag is set, the corresponding interrupt or DMA request is sent if enabled. In addition, if channel 1 is configured in input mode, the current value of the counter is captured in TIMERx_CH0CV register, and the CH0OF flag is set if the CH0IF flag was already high. 0: No generate a channel 1 capture or compare event 1: Generate a channel 1 capture or compare event
0	UPG	Update event generation This bit can be set by software, and cleared by hardware automatically. When this bit is set, the counter is cleared if the center-aligned or up counting mode is selected, else (down counting) it takes the auto-reload value. The prescaler counter is cleared at the same time. 0: No generate an update event 1: Generate an update event

### Channel control register 0 (TIMERx\_CHCTL0)

Address offset: 0x18

Reset value: 0x0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH1COM CEN	CH1COMCTL[2:0]	CH1COM SEN	CH1COM FEN	CH1MS[1:0]	CH0COM CEN	CH0COMCTL[2:0]	CH0COM SEN	CH0COM FEN	CH0MS[1:0]						
CH1CAPFLT[3:0]	CH1CAPPSC[1:0]					CH0CAPFLT[3:0]	CH0CAPPSC[1:0]								

rw rw rw rw rw 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	<p>Channel 1 output compare fast enable</p> <p>Refer to CH0COMFEN description</p>
9:8	CH1MS[1:0]	<p>Channel 1 mode selection</p> <p>This bit-field specifies the direction of the channel and the input signal selection.</p> <p>This bit-field is writable only when the channel is not active. (CH1EN bit in TIMERx_CHCTL2 register is reset).</p> <p>00: Channel 1 is programmed as output mode</p> <p>01: Channel 1 is programmed as input mode, IS1 is connected to CI1FE1</p> <p>10: Channel 1 is programmed as input mode, IS1 is connected to CI0FE1</p> <p>11: Channel 1 is programmed as input mode, IS1 is connected to ITS.</p> <p><b>Note:</b> When CH1MS[1:0]=11, it is necessary to select an internal trigger input through TRGS bits in TIMERx_SMCFG register.</p>
7	CH0COMCEN	<p>Channel 0 output compare clear enable.</p> <p>When this bit is set, if the ETIPP signal is detected as high level, the O0CPRE signal will be cleared.</p> <p>0: Channel 0 output compare clear disable</p> <p>1: Channel 0 output compare clear enable</p>
6:4	CH0COMCTL[2:0]	<p>Channel 0 compare output control</p> <p>This bit-field specifies the compare output mode of the the output prepare signal O0CPRE. In addition, the high level of O0CPRE is the active level, and CH0_O and CH0_ON channels polarity 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 is equals to the output compare register TIMERx_CH0CV.</p> <p>010: Clear the channel output. O0CPRE signal is forced low when the counter is equals to the output compare register TIMERx_CH0CV.</p> <p>011: Toggle on match. O0CPRE toggles when the counter is equals to the output compare register TIMERx_CH0CV.</p> <p>100: Force low. O0CPRE is forced to low level.</p> <p>101: Force high. O0CPRE is forced to high level.</p> <p>110: PWM mode0. When counting up, O0CPRE is high when the counter is smaller than TIMERx_CH0CV, and low otherwise. When counting down, O0CPRE is low when the counter is larger than TIMERx_CH0CV, and high otherwise.</p> <p>111: PWM mode1. When counting up, O0CPRE is low when the counter is smaller than TIMERx_CH0CV, and high otherwise. When counting down, O0CPRE is high when the counter is larger than TIMERx_CH0CV, and low otherwise.</p> <p>If configured in PWM mode, the O0CPRE level changes only when the output compare mode is adjusted from “Timing” mode to “PWM” mode or the comparison result changes.</p> <p>This bit cannot be modified when PROT [1:0] bit-field in TIMERx_CCHP register is</p>

11 and CH0MS bit-field is 00(COMPARE MODE).

3	CH0COMSEN	<p>Channel 0 compare output shadow enable</p> <p>When this bit is set, the shadow register of TIMERx_CH0CV register, which updates at each update event, will be enabled.</p> <p>0: Channel 0 output compare shadow disable</p> <p>1: Channel 0 output compare shadow enable</p> <p>The PWM mode can be used without verifying the shadow register only in single pulse mode (when SPM=1)</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.</p> <p>1: Channel 0 output quickly compare enable.</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 programmed as output mode</p> <p>01: Channel 0 is programmed as input mode, IS0 is connected to CI0FE0</p> <p>10: Channel 0 is programmed as input mode, IS0 is connected to CI1FE0</p> <p>11: Channel 0 is programmed as input mode, IS0 is connected to ITS</p> <p><b>Note:</b> When CH0MS[1:0]=11, it is necessary to select an internal trigger input through TRGS bits in TIMERx_SMCFG register.</p>

#### **Input capture mode:**

Bits	Fields	Descriptions
15:12	CH1CAPFLT[3:0]	<p>Channel 1 input capture filter control</p> <p>Refer to CH0CAPFLT description</p>
11:10	CH1CAPPSC[1:0]	<p>Channel 1 input capture prescaler</p> <p>Refer to CH0CAPPSC description</p>
9:8	CH1MS[1:0]	<p>Channel 1 mode selection</p> <p>Same as Output compare mode</p>
7:4	CH0CAPFLT[3:0]	<p>Channel 0 input capture filter control</p> <p>The CI0 input signal can be filtered by digital filter and this bit-field configure the filtering capability.</p>

Basic principle of digital filter: continuously sample the CI0 input signal according to  $f_{SAMP}$  and record the number of times of the same level of the signal. After reaching the filtering capacity configured by this bit, it is considered to be an effective level.

The filtering capability configuration is as follows:

<b>CH0CAPFLT [3:0]</b>	<b>Times</b>	<b><math>f_{SAMP}</math></b>
4'b0000	Filter disabled.	
4'b0001	2	$f_{CK\_TIMER}$
4'b0010	4	
4'b0011	8	
4'b0100	6	$f_{DTS}/2$
4'b0101	8	
4'b0110	6	$f_{DTS}/4$
4'b0111	8	
4'b1000	6	$f_{DTS}/8$
4'b1001	8	
4'b1010	5	$f_{DTS}/16$
4'b1011	6	
4'b1100	8	
4'b1101	5	$f_{DTS}/32$
4'b1110	6	
4'b1111	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, input capture occurs on every channel input edge

01: The input capture occurs on every 2 channel input edges

10: The input capture occurs on every 4 channel input edges

11: The input capture occurs on 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]	CH2CAPPSC[1:0]	CH2CAPPSC[1:0]	CH2CAPPSC[1:0]	CH2CAPPSC[1:0]	CH2CAPPSC[1:0]	CH2CAPPSC[1:0]
CH3CAPFLT[3:0]	CH3CAPPSC[1:0]				CH2CAPFLT[3:0]										

rw

rw

rw

rw

rw

rw

### Output compare mode:

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
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 programmed as output mode 01: Channel 3 is programmed as input mode, IS3 is connected to CI3FE3 10: Channel 3 is programmed as input mode, IS3 is connected to CI2FE3 11: Channel 3 is programmed as input mode, IS3 is connected to ITS. <b>Note:</b> When CH3MS[1:0]=11, it is necessary to select an internal trigger input through TRGS bits in TIMERx_SMCFG register.
7	CH2COMCEN	Channel 2 output compare clear enable. When this bit is set, if the ETIIP signal is detected as high level, the O2CPRE signal will be cleared. 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 specifies the compare output mode of the the output prepare signal O0CPRE. In addition, the high level of O0CPRE is the active level, and CH0_O and CH0_ON channels polarity depends on CH0P and CH0NP 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 is equals to the output compare register TIMERx_CH2CV. 010: Clear the channel output. O2CPRE signal is forced low when the counter is equals to the output compare register TIMERx_CH2CV. 011: Toggle on match. O2CPRE toggles when the counter is equals to the output compare register TIMERx_CH2CV. 100: Force low. O2CPRE is forced to low level. 101: Force high. O2CPRE is forced to high level. 110: PWM mode 0. When counting up, O2CPRE is high when the counter is smaller than TIMERx_CH2CV, and low otherwise. When counting down, O2CPRE is low

when the counter is larger than TIMERx\_CH2CV, and high otherwise.

111: PWM mode 1. When counting up, O2CPRE is low when the counter is smaller than TIMERx\_CH2CV, and high otherwise. When counting down, O2CPRE is high when the counter is larger than TIMERx\_CH2CV, and low otherwise.

If configured in PWM mode, the O2CPRE level changes only when the output compare mode is adjusted from “Timing” mode to “PWM” mode or the comparison result changes.

This bit cannot be modified when PROT [1:0] bit-field in TIMERx\_CCHP register is 11 and CH2MS bit-field 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</p> <p>1: Channel 2 output compare shadow enable</p> <p>The PWM mode can be used without verifying the shadow register only in single pulse mode (when SPM=1)</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	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.</p> <p>1: Channel 2 output quickly compare enable.</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> <p>This bit-field is writable only when the channel is not active. (CH2EN bit in TIMERx_CHCTL2 register is reset.).</p> <p>00: Channel 2 is programmed as output mode</p> <p>01: Channel 2 is programmed as input mode, IS2 is connected to CI2FE2</p> <p>10: Channel 2 is programmed as input mode, IS2 is connected to CI3FE2</p> <p>11: Channel 2 is programmed as input mode, IS2 is connected to ITS.</p> <p><b>Note:</b> When CH2MS[1:0]=11, it is necessary to select an internal trigger input through TRGS bits in TIMERx_SMCFG register.</p>

#### **Input capture mode:**

Bits	Fields	Descriptions
15:12	CH3CAPFLT[3:0]	<p>Channel 3 input capture filter control</p> <p>Refer to CH0CAPFLT description</p>

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  The CI2 input signal can be filtered by digital filter and this bit-field configure the filtering capability.  Basic principle of digital filter: continuously sample the CI2 input signal according to $f_{SAMP}$ and record the number of times of the same level of the signal. After reaching the filtering capacity configured by this bit, it is considered to be an effective level.  The filtering capability configuration is as follows:																																										
	<table border="1"> <thead> <tr> <th>CH2CAPFLT [3:0]</th><th>Times</th><th><math>f_{SAMP}</math></th></tr> </thead> <tbody> <tr> <td>4'b0000</td><td colspan="2">Filter disabled.</td></tr> <tr> <td>4'b0001</td><td>2</td><td rowspan="3"><math>f_{CK\_TIMER}</math></td></tr> <tr> <td>4'b0010</td><td>4</td></tr> <tr> <td>4'b0011</td><td>8</td></tr> <tr> <td>4'b0100</td><td>6</td><td rowspan="2"><math>f_{DTS}/2</math></td></tr> <tr> <td>4'b0101</td><td>8</td></tr> <tr> <td>4'b0110</td><td>6</td><td rowspan="2"><math>f_{DTS}/4</math></td></tr> <tr> <td>4'b0111</td><td>8</td></tr> <tr> <td>4'b1000</td><td>6</td><td rowspan="2"><math>f_{DTS}/8</math></td></tr> <tr> <td>4'b1001</td><td>8</td></tr> <tr> <td>4'b1010</td><td>5</td><td rowspan="3"><math>f_{DTS}/16</math></td></tr> <tr> <td>4'b1011</td><td>6</td></tr> <tr> <td>4'b1100</td><td>8</td></tr> <tr> <td>4'b1101</td><td>5</td><td rowspan="3"><math>f_{DTS}/32</math></td></tr> <tr> <td>4'b1110</td><td>6</td></tr> <tr> <td>4'b1111</td><td>8</td></tr> </tbody> </table>		CH2CAPFLT [3:0]	Times	$f_{SAMP}$	4'b0000	Filter disabled.		4'b0001	2	$f_{CK\_TIMER}$	4'b0010	4	4'b0011	8	4'b0100	6	$f_{DTS}/2$	4'b0101	8	4'b0110	6	$f_{DTS}/4$	4'b0111	8	4'b1000	6	$f_{DTS}/8$	4'b1001	8	4'b1010	5	$f_{DTS}/16$	4'b1011	6	4'b1100	8	4'b1101	5	$f_{DTS}/32$	4'b1110	6	4'b1111	8
CH2CAPFLT [3:0]	Times	$f_{SAMP}$																																										
4'b0000	Filter disabled.																																											
4'b0001	2	$f_{CK\_TIMER}$																																										
4'b0010	4																																											
4'b0011	8																																											
4'b0100	6	$f_{DTS}/2$																																										
4'b0101	8																																											
4'b0110	6	$f_{DTS}/4$																																										
4'b0111	8																																											
4'b1000	6	$f_{DTS}/8$																																										
4'b1001	8																																											
4'b1010	5	$f_{DTS}/16$																																										
4'b1011	6																																											
4'b1100	8																																											
4'b1101	5	$f_{DTS}/32$																																										
4'b1110	6																																											
4'b1111	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, input capture occurs on every channel input edge 01: The input capture occurs on every 2 channel input edges 10: The input capture occurs on every 4 channel input edges 11: The input capture occurs on 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 complementary output high level is active level 1: Channel 0 complementary output low level is active level When channel 0 is configured in input mode, together with CH0P, this bit is used to define the polarity of CI0. This bit cannot be modified when PROT [1:0] bit-field in TIMERx_CCHP register is 11 or 10.
2	CH0NEN	Channel 0 complementary output enable

When channel 0 is configured in output mode, setting this bit enables the complementary output in channel0.

- 0: Channel 0 complementary output disabled
- 1: Channel 0 complementary output enabled

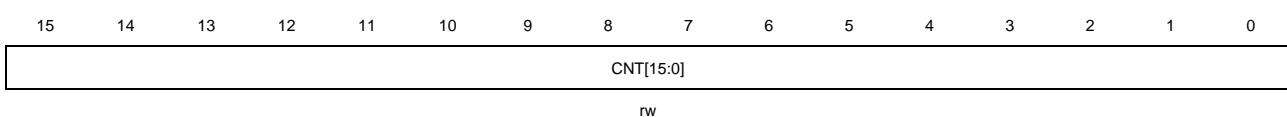
1	CH0P	<p>Channel 0 capture/compare function polarity</p> <p>When channel 0 is configured in output mode, this bit specifies the output signal polarity.</p> <ul style="list-style-type: none"> <li>0: Channel 0 high level is active level</li> <li>1: Channel 0 low level is active level</li> </ul> <p>When channel 0 is configured in input mode, this bit specifies the CI0 signal polarity.</p> <p>[CH0NP, CH0P] will select the active trigger or capture polarity for CI0FE0 or CI1FE0.</p> <p>[CH0NP==0, CH0P==0]: CIxFE0's rising edge is the active signal for capture or trigger operation in slave mode. And CIxFE0 will not be inverted.</p> <p>[CH0NP==0, CH0P==1]: CIxFE0's falling edge is the active signal for capture or trigger operation in slave mode. And CIxFE0 will be inverted.</p> <p>[CH0NP==1, CH0P==0]: Reserved.</p> <p>[CH0NP==1, CH0P==1]: Reserved.</p> <p>This bit cannot be modified when PROT [1:0] bit-field in TIMERx_CCHP register is 11 or 10.</p>
0	CH0EN	<p>Channel 0 capture/compare function enable</p> <p>When channel 0 is configured in output mode, setting this bit enables CH0_O signal in active state. When channel 0 is configured in input mode, setting this bit enables the capture event in channel0.</p> <ul style="list-style-type: none"> <li>0: Channel 0 disabled</li> <li>1: Channel 0 enabled</li> </ul>

## Counter register (TIMERx\_CNT)

Address offset: 0x24

Reset value: 0x0000

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



Bits	Fields	Descriptions
15:0	CNT[15:0]	This bit-field indicates the current counter value. Writing to this bit-field can change the value of the counter.

### **Prescaler register (TIMERx\_PSC)**

Address offset: 0x28

Reset value: 0x0000

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

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 TIMER_CK clock is divided by (PSC+1) to generate the counter clock. The value of this bit-field will be loaded to the corresponding shadow register at every update event.</p>

### **Counter auto reload register (TIMERx\_CAR)**

Address offset: 0x2C

Reset value: 0x0000

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

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><b>Note:</b> 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																							

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
15:8	Reserved	Must be kept at reset value.
7:0	CREP[7:0]	<p>Counter repetition value</p> <p>This bit-field 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-field 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															

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

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

shadow register updates every update event.

### **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 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>The bit can be set to 1 by:</p> <ul style="list-style-type: none"> <li>- Write 1 to this bit</li> <li>- If OAEN is set to 1, this bit is set to 1 at the next update event.</li> </ul> <p>The bit can be cleared to 0 by:</p> <ul style="list-style-type: none"> <li>- Write 0 to this bit</li> <li>- Valid fault input (asynchronous).</li> </ul> <p>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: Disable channel outputs (CHxO or CHxON). 1: Enabled channel outputs (CHxO or CHxON).</p> <p><b>Note:</b> This bit is only valid when CHxMS=2'b00.</p>
14	OAEN	<p>Output automatic enable</p> <p>0: The POEN bit can only be set by software. 1: POEN can be set at the next update event, if the break input is not active.</p> <p>This bit can be modified only when PROT [1:0] bit-filed in TIMERx_CCHP register is 00.</p>
13	BRKP	<p>Break polarity</p> <p>This bit specifies the polarity of the BRKIN input signal.</p> <p>0: BRKIN input active low 1: BRKIN input active high</p>
12	BRKEN	<p>Break enable</p> <p>This bit can be set to enable the BRKIN and CCS clock failure event inputs.</p> <p>0: Break inputs disabled 1: Break inputs enabled</p> <p>This bit can be modified only when PROT [1:0] bit-filed in TIMERx_CCHP register is 00.</p>
11	ROS	<p>Run mode off-state configure</p> <p>When POEN bit is set, this bit specifies the output state for the channels which has a complementary output and has been configured in output mode.</p> <p>0: When POEN bit is set, the channel output signals (CHx_O/CHx_ON) are disabled. 1: When POEN bit is set, the channel output signals (CHx_O/CHx_ON) are enabled, with relationship to CHxEN/CHxNEN bits in TIMERx_CHCTL2 register.</p> <p>This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is</p>

10 or 11.

10	IOS	Idle mode off-state configure When POEN bit is reset, this bit specifies the output state for the channels which has been configured in output mode. 0: When POEN bit is reset, the channel output signals (CHx_O/CHx_ON) are disabled. 1: When POEN bit is reset, the channel output signals (CHx_O/CHx_ON) are enabled, with relationship to CHxEN/CHxNEN bits in TIMERx_CHCTL2 register. This bit cannot be modified when PROT [1:0] bit-field in TIMERx_CCHP register is 10 or 11.
9:8	PROT[1:0]	Complementary register protect control This bit-field specifies the write protection property of registers. 00: protect disable. No write protection. 01: PROT mode 0. The ISOx/ISOxN bits in TIMERx_CTL1 register and the BRKEN/BRKP/OAEN/DTCFG bits in TIMERx_CCHP register are writing protected. 10: PROT mode 1. In addition of the registers in PROT mode 0, the CHxP/CHxNP bits in TIMERx_CHCTL2 register (if related channel is configured in output mode) and the ROS/IOS bits in TIMERx_CCHP register are writing protected. 11: PROT mode 2. In addition of the registers in PROT mode 1, the CHxCOMCTL/CHxCOMSEN bits in TIMERx_CHCTL0/1 registers (if the related channel is configured in output) are writing protected. This bit-field can be written only once after the reset. Once the TIMERx_CCHP register has been written, this bit-field will be writing protected.
7:0	DTCFG[7:0]	Dead time configure The relationship between DTVAL value and the duration of dead-time is as follow:

DTCFG[7:5]	The duration of dead-time
3'b0xx	DTCFG[7:0] * tDTS_CK
3'b10x	(64 + DTCFG[5:0]) * tDTS_CK * 2
3'b110	(32 + DTCFG[4:0]) * tDTS_CK * 8
3'b111	(32 + DTCFG[4:0]) * tDTS_CK * 16

**Note:**

1. tDTS\_CK is the period of DTS\_CK which is configured by CKDIV[1:0] in TIMERx\_CTL0.
2. This bit can be modified only when PROT [1:0] bit-field in TIMERx\_CCHP register is 00.

### DMA configuration register (TIMERx\_DMACFG)

Address offset: 0x48

Reset value: 0x0000

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

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 filed defines the number(n) of the register that DMA will access(R/W), n = (DMATC [4:0] +1). DMATC [4:0] is from 5'b0_0000 to 5'b1_0001.
7:5	Reserved	Must be kept at reset value.
4:0	DMATA [4:0]	DMA transfer access start address This filed define the first address for the DMA access the TIMERx_DMATB. When access is done through the TIMERx_DMA address first time, this bit-field specifies the address you just access. And then the second access to the TIMERx_DMATB, you will access the address of start address + 0x4.

### 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]	DMA transfer buffer When a read or write operation is assigned to this register, the register located at the address range (Start Addr + Transfer Timer* 4) will be accessed. The transfer Timer is calculated by hardware, and ranges from 0 to DMATC.

### Configuration register (TIMERx\_CFG)

Address offset: 0xFC

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												CHVSEL	OUTSEL		

rw

rw

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
15:2	Reserved	Must be kept at reset value.
1	CHVSEL	<p>Write CHxVAL register selection</p> <p>This bit-field set and reset by software.</p> <p>1: If write the CHxVAL register, the write value is same as the CHxVAL value, the write access ignored</p> <p>0: No effect</p>
0	OUTSEL	<p>The output value selection</p> <p>This bit-field set and reset by software</p> <p>1: If POEN and IOS is 0, the output disabled</p> <p>0: No effect</p>

## 16.2. General level0 timer (TIMERx, x=1, 2, 3, 4)

### 16.2.1. Overview

The general level0 timer module (Timer1, 2, 3, 4) is a four-channel timer that supports input capture, output compare. They can generate PWM signals to control motor or be used for power management applications. The general level0 time reference is a 16-bit 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.

Timer and timer are completely independent, but there may be synchronized to provide a larger timer with their counters incrementing in unison.

### 16.2.2. Characteristics

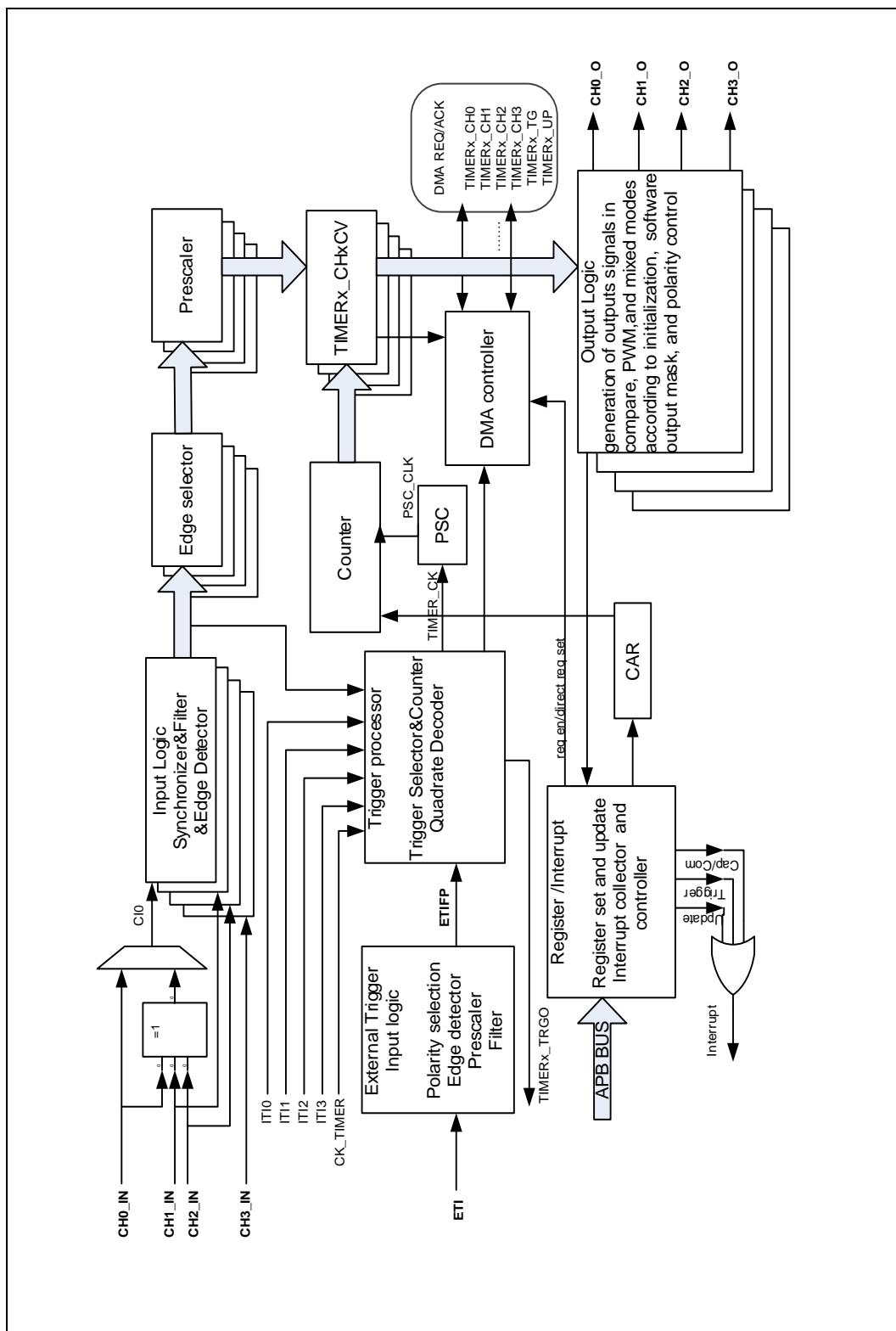
- Total channel num: 4.
- Counter width: 16 bits.
- Source of count clock is selectable: internal clock, internal trigger, external input, external trigger.
- Multiple counter modes: count up, count down, count up/down.
- Quadrature decoder: used to track motion and determine both rotation direction and position.
- Hall sensor: for 3-phase motor control.
- Programmable prescaler: 16 bits. Factor can be changed on the go.
- Each channel is user-configurable:
  - Input capture mode, output compare mode, programmable PWM mode, single pulse mode
- Auto-reload function.
- Interrupt output or DMA request on: update, trigger event, and compare/capture event.
- Daisy chaining of timer modules to allow a single timer to initiate multiple timing events.
- Timer synchronization allows selected timers to start counting on the same clock cycle.
- Timer master-slave management.

### 16.2.3. Block diagram

[Figure 16-29. General Level 0 timer block diagram](#) provides details on the internal

configuration of the general level0 timer.

**Figure 16-29. General Level 0 timer block diagram**



## 16.2.4. Function overview

### Clock source configuration

The general level0 TIMER has the capability of being clocked by either the CK\_TIMER or an alternate clock source controlled by SMC (TIMERx\_SMCFG bit [2:0]).

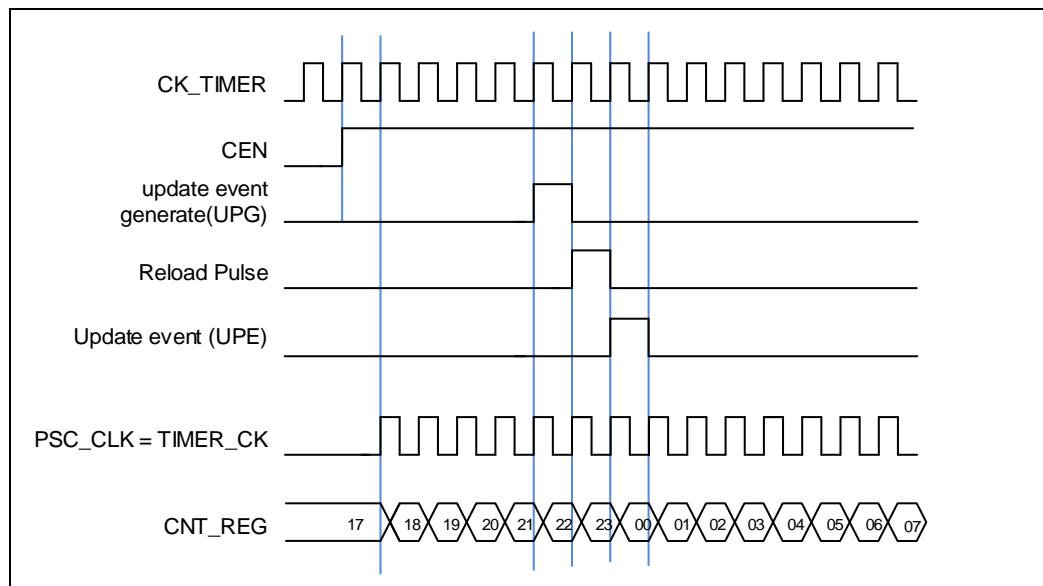
- SMC [2:0] == 3'b000. Internal timer clock CK\_TIMER which is from module RCU.

The default internal clock source is the CK\_TIMER used to drive the counter prescaler when the SMC [2:0] == 3'b000. When the CEN is set, the CK\_TIMER will be divided by PSC value to generate PSC\_CLK.

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

If the SMC [2:0] in the TIMERx\_SMCFG register are setting to an available value including 0x1, 0x2, 0x3 and 0x7, the prescaler is clocked by other clock sources selected by the TRGS [2:0] in the TIMERx\_SMCFG register and described as follows. When the SMC [2:0] bits are set to 0x4, 0x5 or 0x6, the internal clock CK\_TIMER is the counter prescaler driving clock source.

**Figure 16-30. Timing chart of internal clock divided by 1**



- SMC [2:0] == 3'b111(external clock mode 0). External input pin 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\_CI0/TIMERx\_CI1. This mode can be selected by setting SMC [2:0] to 0x7 and the TRGS [2:0] to 0x4, 0x5 or 0x6.

And, the counter prescaler can also be driven by rising edge on the internal trigger input pin ITI0/1/2/3. This mode can be selected by setting SMC [2:0] to 0x7 and the TRGS [2:0] to 0x0, 0x1, 0x2 or 0x3.

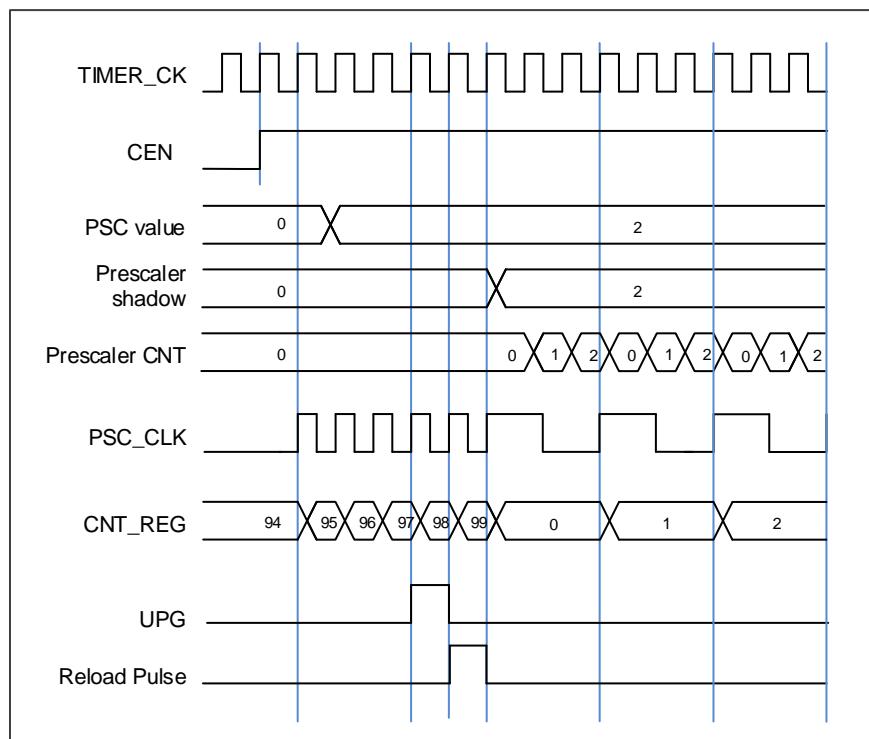
- SMC1== 1'b1(external clock mode 1). External input pin source (ETI)

The TIMER\_CK, 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 to set the SMC [2:0] to 0x7 and the TRGS [2:0] to 0x7 respectively. Note that the ETI signal is derived from the ETI pin sampled by a digital filter. When the ETI signal is selected as clock source, the trigger controller including the edge detection circuitry will generate a clock pulse on each ETI signal rising edge to clock the counter prescaler.

### Clock prescaler

The counter clock (PSC\_CK) is obtained by the TIMER\_CK through the prescaler, and the prescale factor can be configured from 1 to 65536 through the prescaler register (TIMERx\_PSC). The new written prescaler value will not take effect until the next update event.

**Figure 16-31. Timing chart of PSC value change from 0 to 2**



### Counter up counting

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 will start counting up from 0 again. The update event is generated at each counter overflow. The counting direction bit DIR in the TIMERx\_CTL1 register should be set to 0 for the up counting mode.

When the update event is set by the UPG bit in the TIMERx\_SWEVG register, the counter

value will be initialized to 0 and generates an update event.

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

When an update event occurs, all the shadow registers (counter autoreload register, prescaler register) are updated.

**Figure 16-32. Timing chart of up counting mode, PSC=0/2** show some examples of the counter behavior for different clock prescaler factor when TIMERx\_CAR=0x99.

**Figure 16-32. Timing chart of up counting mode, PSC=0/2**

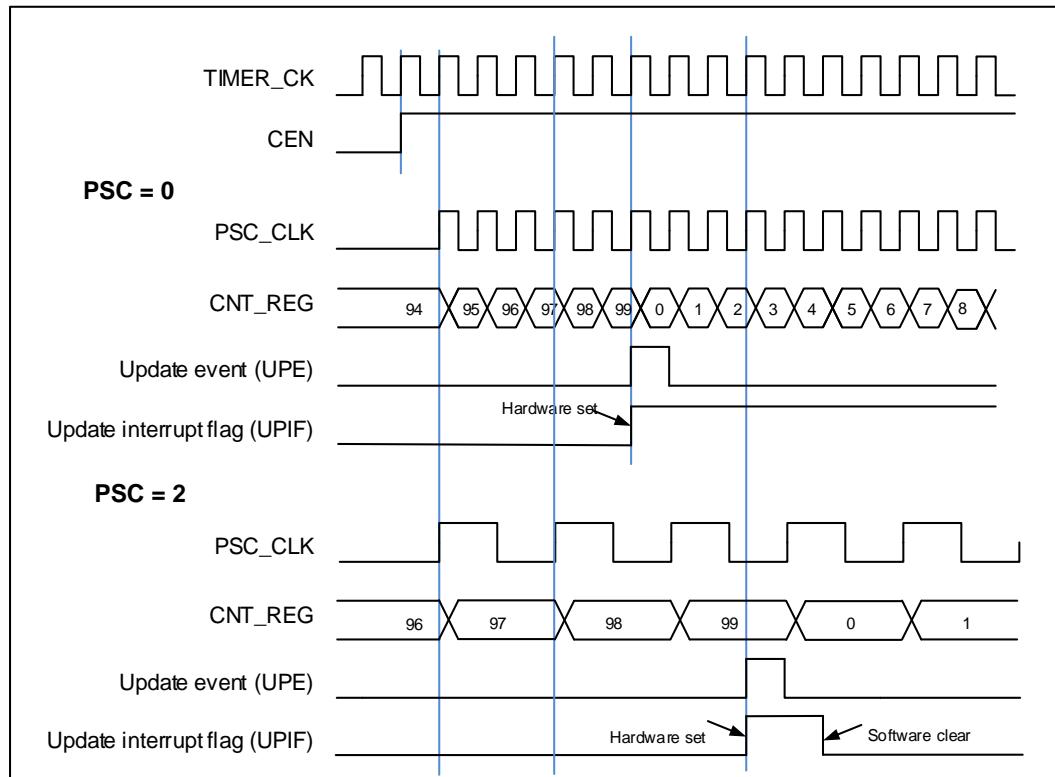
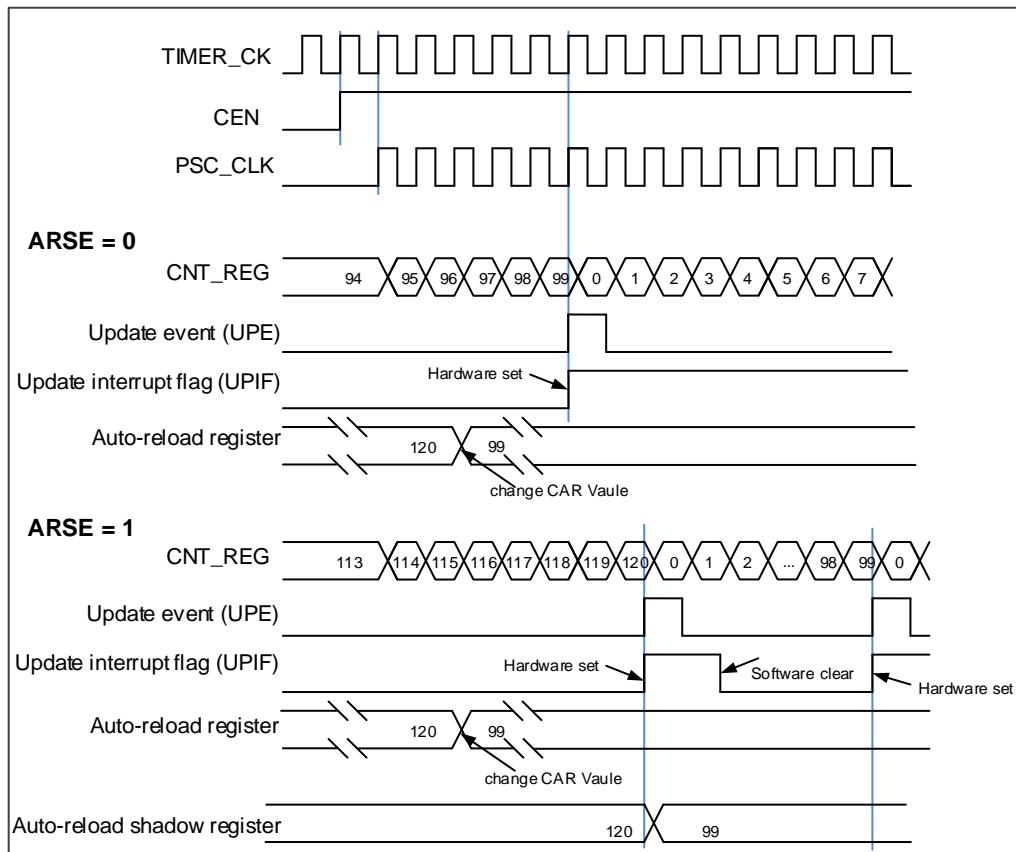


Figure 16-33. Timing chart of up counting mode, change TIMERx\_CAR ongoing



### Counter down counting

In this mode, the counter counts down continuously from the counter-reload value, which is defined in the TIMERx\_CAR register, to 0 in a count-down direction. Once the counter reaches to 0, the counter will start counting down from the counter-reload value. The update event is generated at each counter underflow. The counting direction bit DIR in the TIMERx\_CTL0 register should be set to 1 for the down-counting mode.

When the update event is set by the UPG bit in the TIMERx\_SWEVG register, the counter value will be initialized to the counter-reload value and generates an update event.

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

When an update event occurs, all the shadow registers (counter autoreload register, prescaler register) are updated.

**Figure 16-34. Timing chart of down counting mode, PSC=0/2** show some examples of the counter behavior for different clock frequencies when TIMERx\_CAR=0x99.

Figure 16-34. Timing chart of down counting mode, PSC=0/2

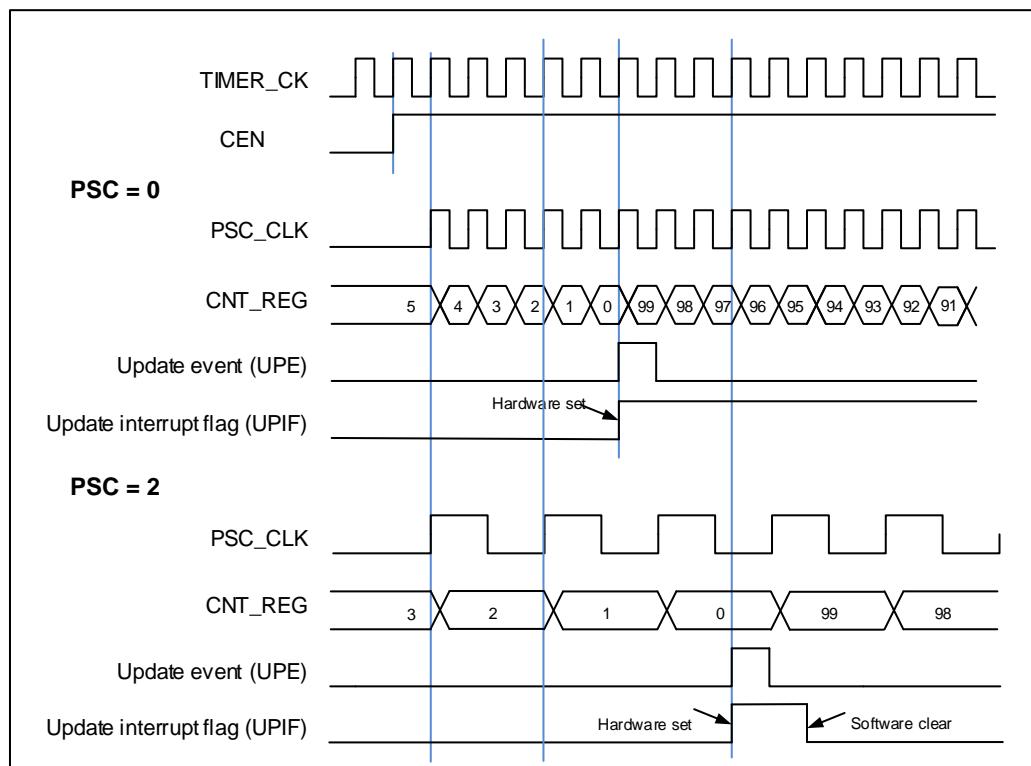
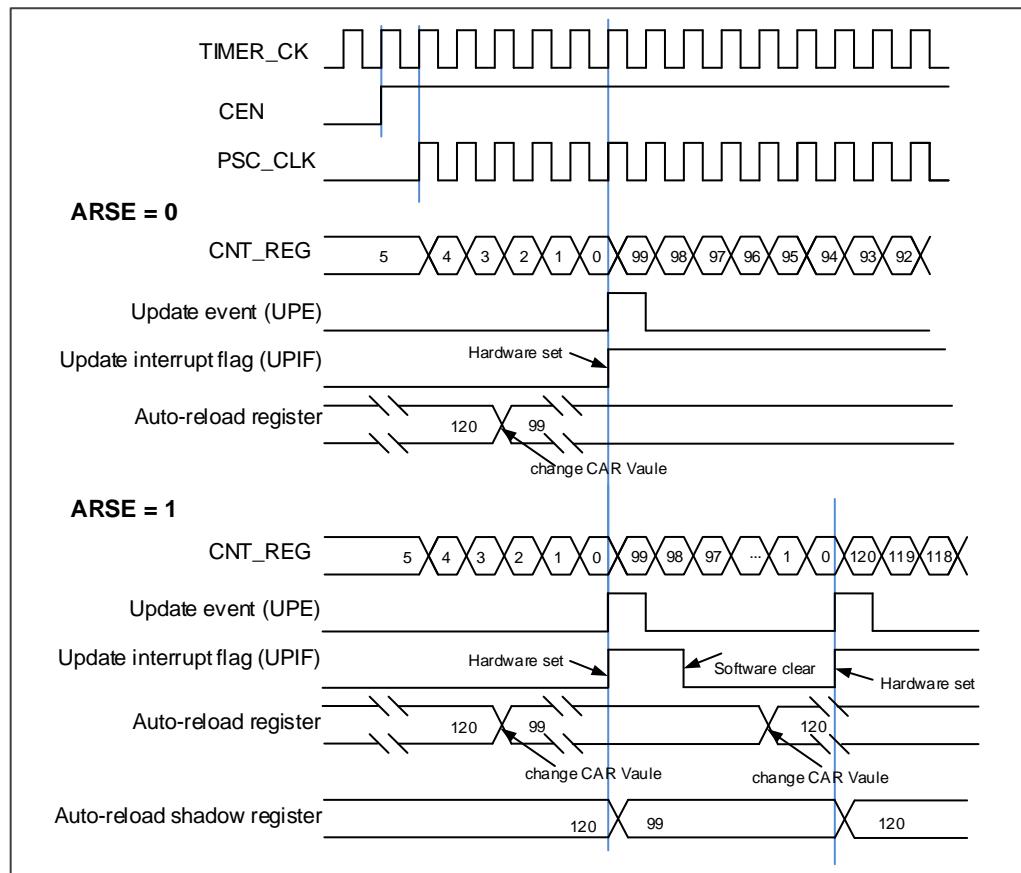


Figure 16-35. Timing chart of down counting mode, change TIMERx\_CAR ongoing



### Counter center-aligned counting

In this mode, the counter counts up from 0 to the counter-reload value and then counts down to 0 alternatively. The Timer module generates an overflow event when the counter counts to the counter-reload value subtract 1 in the up-counting direction and generates an underflow event when the counter counts to 1 in the down-counting mode. The counting direction bit DIR in the TIMERx\_CTL0 register is read-only and indicates the counting direction when in the center-aligned mode.

Setting the UPG bit in the TIMERx\_SWEVG register will initialize the counter value to 0 and generates an update event irrespective of whether the counter is counting up or down in the center-align counting mode.

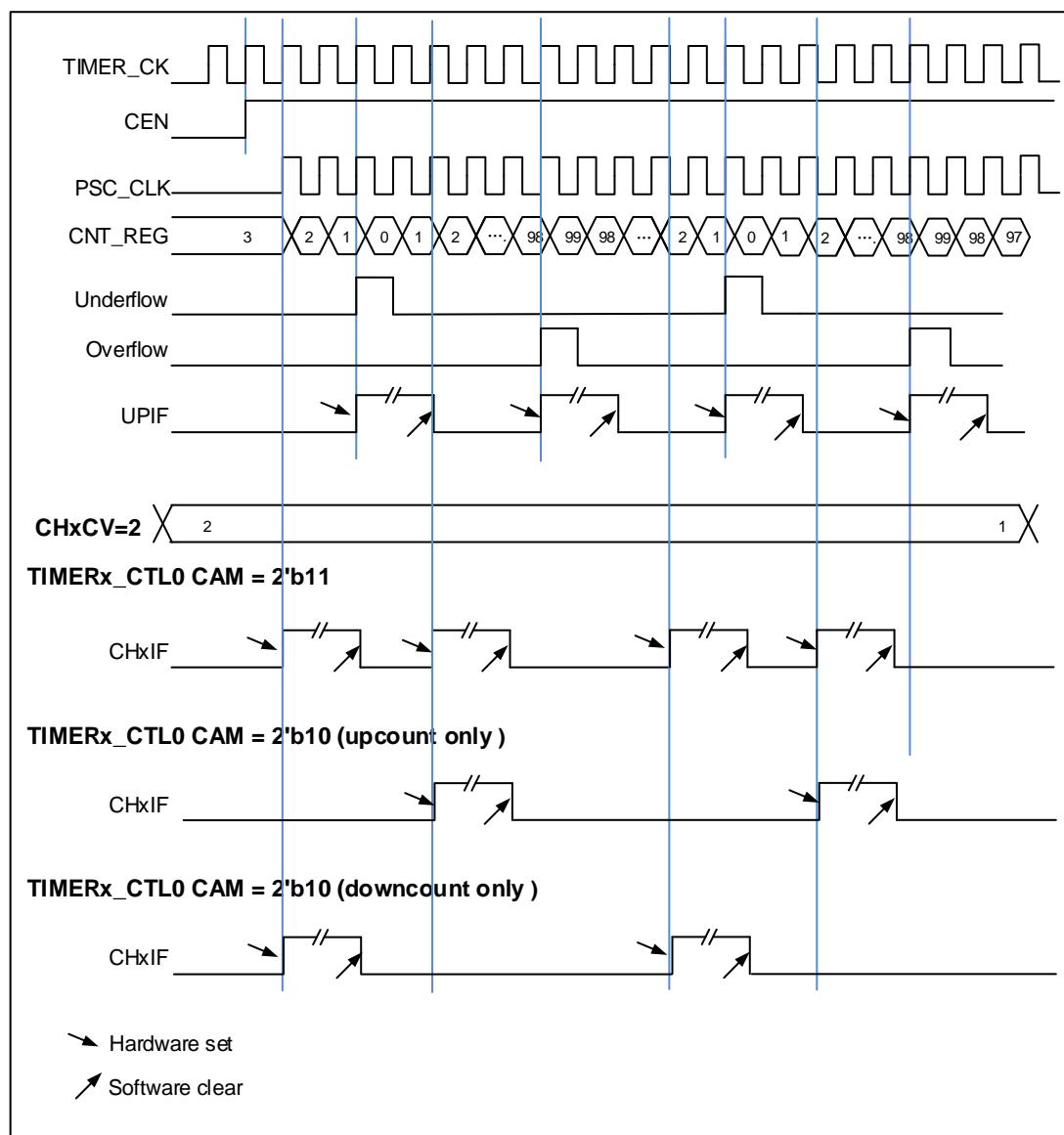
The UPIF bit in the TIMERx\_INTF register can be set to 1 either when an underflow event or an overflow event occurs. While the CHxIF bit is associated with the value of CAM in TIMERx\_CTL0. The details refer to [Figure 16-36. Timing chart of center-aligned counting mode](#)

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

When an update event occurs, all the shadow registers (counter autoreload register, prescaler register) are updated.

[Figure 16-36. Timing chart of center-aligned counting mode](#) show some examples of the counter behavior when TIMERx\_CAR=0x99, TIMERx\_PSC=0x0

Figure 16-36. Timing chart of center-aligned counting mode



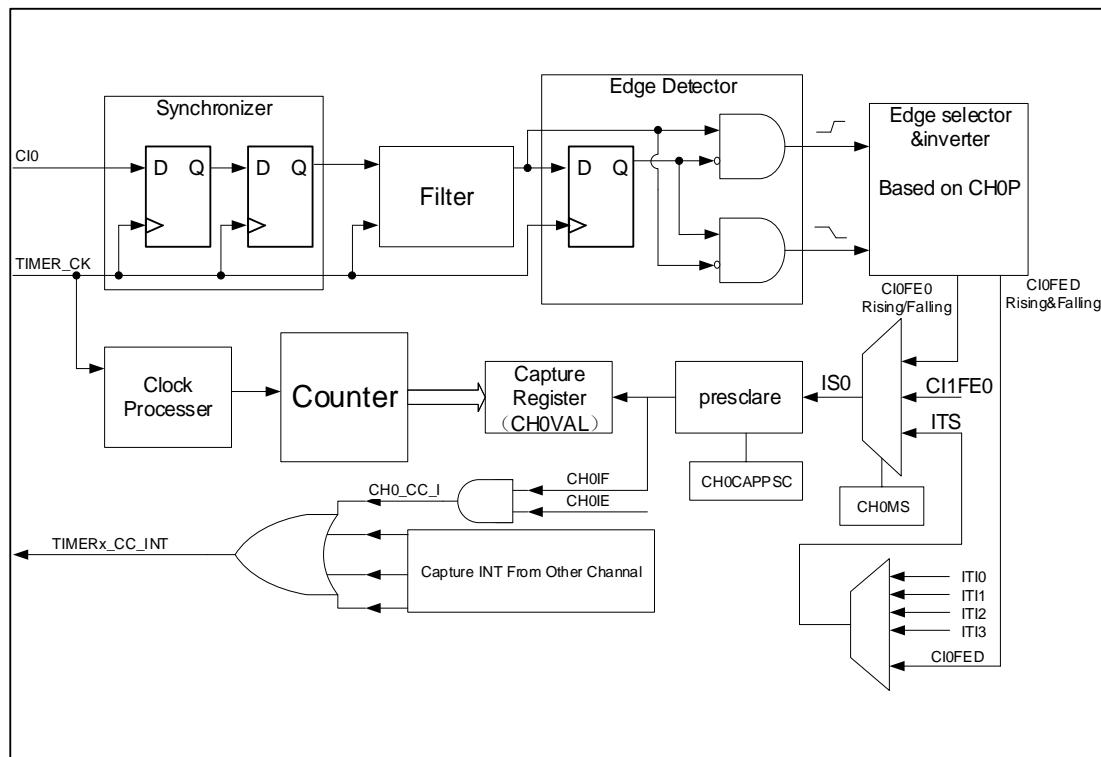
### Input capture and output compare channels

The general level0 Timer has four independent channels which can be used as capture inputs or compare match outputs. Each channel is built around a channel capture compare register including an input stage, channel controller and an output stage.

#### ■ Channel input capture function

Capture mode allows the channel to perform measurements such as pulse timing, frequency, period, duty cycle and so on. The input stage consists of a digital filter, a channel polarity selection, edge detection and a channel prescaler. When a selected edge occurs on the channel input, the current value of the counter is captured into the **TIMERx\_CHxCV** register, at the same time the **CHxIF** bit is set and the channel interrupt is generated if enabled by **CHxIE** = 1.

Figure 16-37. Channel input capture principle



One of channels' input signals (Clx) can be chosen from the TIMERx\_CHx signal or the Exclusive-OR function of the TIMERx\_CH0, TIMERx\_CH1 and TIMERx\_CH2 signals. First, the channel input signal (Clx) is synchronized to TIMER\_CK domain, and then sampled by a digital filter to generate a filtered input signal. Then through the edge detector, the rising and falling edge are detected. You can select one of them by CHxP. One more selector is for the other channel and trig, controlled by CHxMS. The IC\_prescaler make several the input event generate one effective capture event. On the capture event, TIMERx\_CHxCV will restore the value of Counter.

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

**Step1:** Filter Configuration. (CHxCAPFLT in TIMERx\_CHCTL0)

Based on the input signal and requested signal quality, configure compatible CHxCAPFLT.

**Step2:** Edge Selection. (CHxP in TIMERx\_CHCTL2)

Rising or falling edge, choose one by CHxP.

**Step3:** Capture source Selection. (CHxMS in TIMERx\_CHCTL0)

As soon as you select one input capture source by CHxMS, you have set the channel to input mode ( CHxMS!=0x0) and TIMERx\_CHxCV cannot be written any more.

**Step4:** Interrupt enable. (CHxIE and CHxDEN in TIMERx\_DMAINTEN)

Enable the related interrupt enable; you can got the interrupt and DMA request.

**Step5:** Capture enables. (CHxEN in TIMERx\_CHCTL2)

**Result:** When you wanted input signal is got, TIMERx\_CHxCV will be set by counter's value.

And CHxIF is asserted. If the CHxIF is high, the CHxOF will be asserted also. The interrupt and DMA request will be asserted based on the your configuration of CHxIE and CHxDEN in TIMERx\_DMINTEN

**Direct generation:** If you want to generate a DMA request or interrupt, you can set CHxG by software directly.

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

## ■ Channel output compare function

In channel output compare function, the TIMERx can generate timed pulses with programmable position, polarity, duration, and frequency. When the counter matches the value in the CHxVAL register of an output compare channel, the channel (n) output can be set, cleared, or toggled based on CHxCOMCTL. when the counter reaches the value in the CHxVAL register, the CHxIF bit is set and the channel (n) interrupt is generated if CHxIE = 1. And the DMA request will be assert, if CxCDE=1.

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

**Step1:** Clock configuration. Such as clock source, clock prescaler and so on.

**Step2:** Compare mode configuration.

- \* Set the shadow enable mode by CHxCOMSEN
- \* Set the output mode (Set/Clear/Toggle) by CHxCOMCTL.
- \* Select the active high polarity by CHxP
- \* Enable the output by CHxEN

**Step3:** Interrupt/DMA-request enables configuration by CHxIE/CxCDE

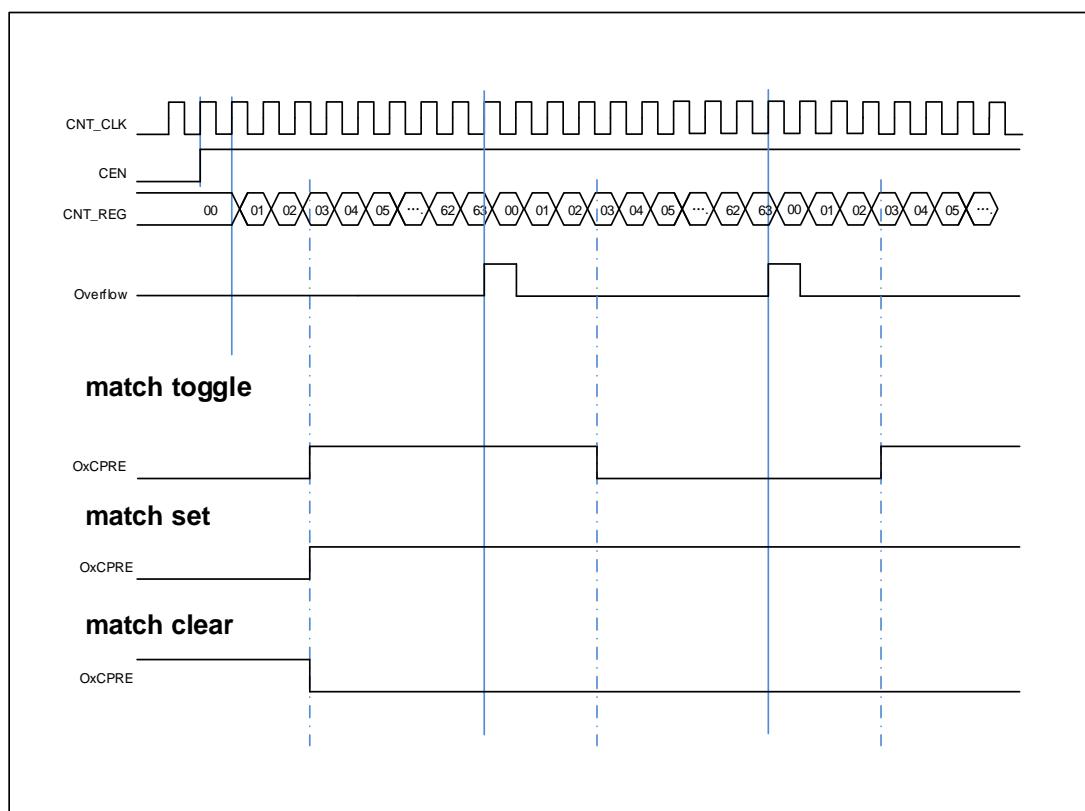
**Step4:** Compare output timing configuration by TIMERx\_CAR and TIMERx\_CHxCV.

About the CHxVAL, you can change it on the go to meet the waveform you expected.

**Step5:** Start the counter by CEN.

The timechart below show the three compare modes toggle/set/clear. CAR=0x63, CHxVAL=0x3

Figure 16-38. Output-compare under three modes



### Output PWM function

In the output PWM function (by setting the CHxCOMCTL bits to 3'b110 (PWM mode0) or to 3'b 111(PWM mode1), the channel can outputs PWM waveform according to the TIMERx\_CAR registers and TIMERx\_CHxCV registers.

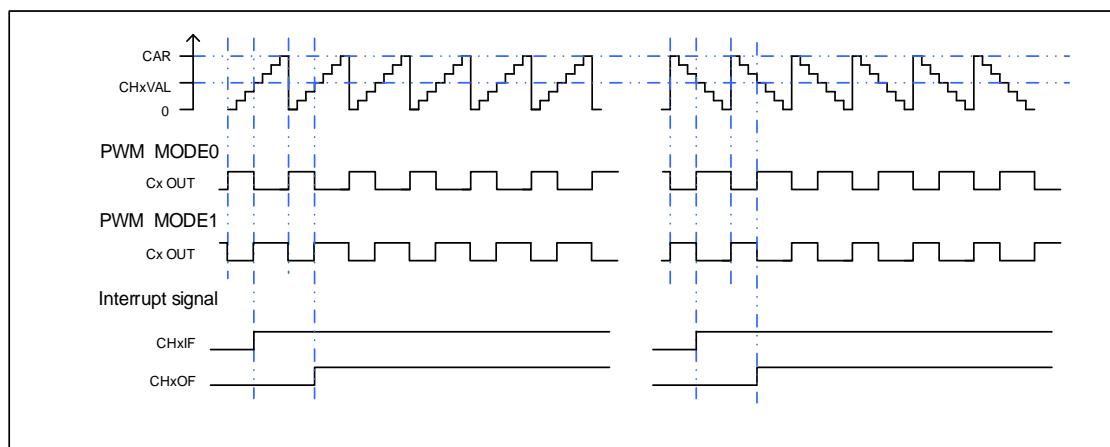
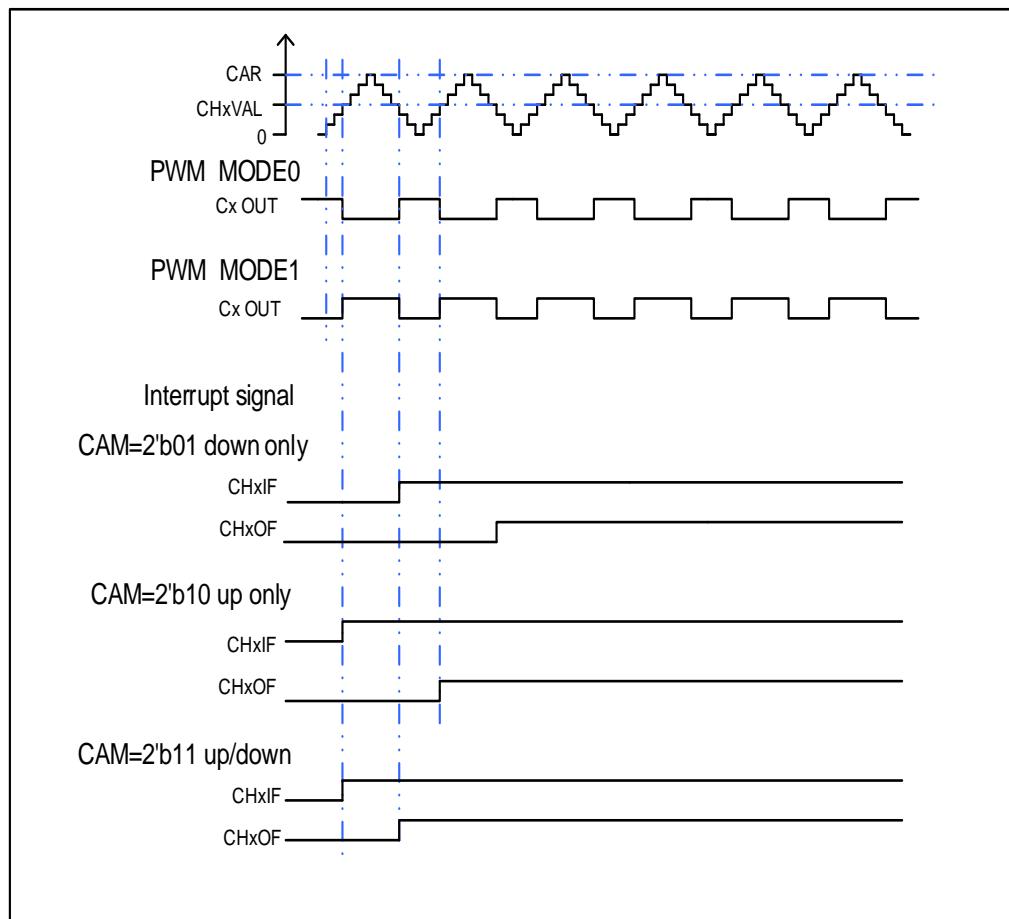
Based on the counter mode, we have can also divide PWM into EAPWM (Edge aligned PWM) and CAPWM (Centre aligned PWM).

The EAPWM period is determined by TIMERx\_CAR and duty cycle is by TIMERx\_CHxCV. [Figure 16-39. 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 16-40. 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 16-39. EAPWM timechart**

**Figure 16-40. 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 ETIFE signal is derived from the external ETI pin and when it is set to a high level by setting the CHxCOMCEN bit to 1 in the TIMERx\_CHCTL0 register. The OxCPRE signal will not return to its active level until the next update event occurs.

### **Quadrature decoder**

Refer to [\*\*Quadrature decoder\*\*](#).

### **Hall sensor function**

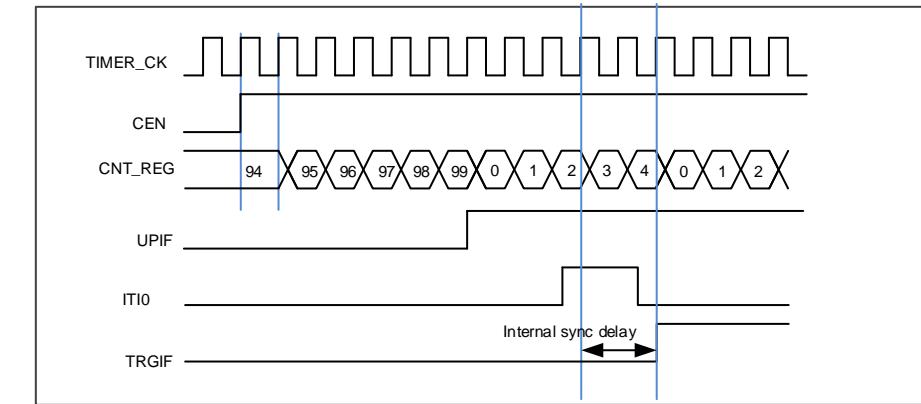
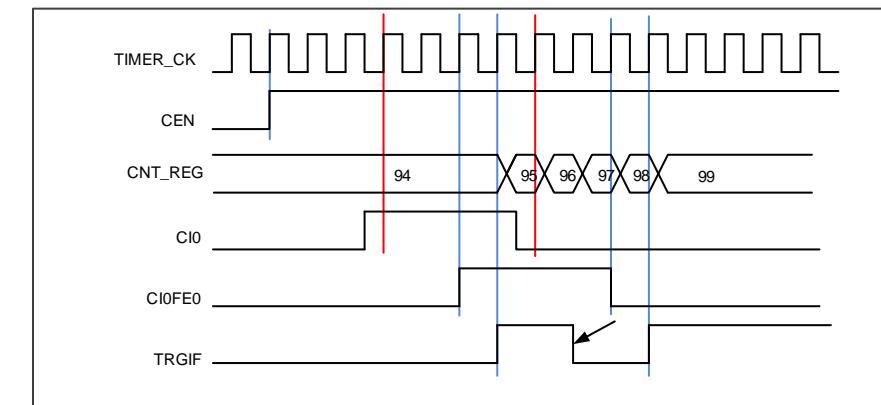
Refer to [\*\*Hall sensor function\*\*](#).

### **Master-slave management**

The TIMERx can be synchronized with a trigger in several modes including the restart mode, the pause mode and the event mode which is selected by the SMC[2:0] in the TIMERx\_SMCFG register. The trigger input of these modes can be selected by the TRGS[2:0] in the TIMERx\_SMCFG register.

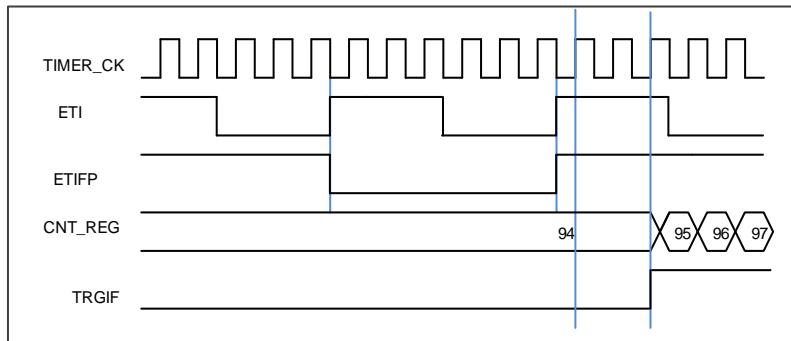
**Table 16-5. Examples of slave mode**

	<b>Mode Selection</b>	<b>Source Selection</b>	<b>Polarity Selection</b>	<b>Filter and Prescaler</b>
LIST	SMC[2:0] 3'b100 (restart mode) 3'b101 (pause mode) 3'b110 (event mode)	TRGS[2:0] 000: ITI0 001: ITI1 010: ITI2 011: ITI3 100: CI0F_ED 101: CI0FE0 110: CI1FE1 111: ETIFF	If you choose the CI0FE0 or CI1FE1, configure the CHxP for the polarity selection and inversion. If you choose the ETIF, configure the ETP for polarity selection and inversion.	For the ITIx no filter and prescaler can be used. For the CIx, configure Filter by CHxCAPFLT, no prescaler can be used. For the ETIF, configure Filter by ETFC and Prescaler by ETPSC.

	Mode Selection	Source Selection	Polarity Selection	Filter and Prescaler
Exam1	Restart mode  The counter can be clear and restart when a rising trigger input.	TRGS[2:0]=3'b000  ITI0 is the selection.	-  For ITI0, no polarity selector can be used.	-  For the ITI0, no filter and prescaler can be used.
<b>Figure 16-41. Restart mode</b>				
			 <p>The diagram shows the timing of various signals over time. The CNT_REG signal is a sequence of hex digits: 94, X, 95, 96, 97, 98, 99, 0, 1, 2, 3, 4, 0, 1, 2. The ITI0 signal is asserted at the start of the sequence. The TRGIF signal is asserted after a delay relative to the ITI0 signal.</p>	
Exam2	Pause mode  The counter can be paused when the trigger input is low.	TRGS[2:0]=3'b101  CI0FE0 is the selection.	TI0S=0. ( Non-xor ), CH0P==0, no inverted.  Capture will be sensitive to the rising edge only.	Filter is bypass in this example.
<b>Figure 16-42. Pause mode</b>				
			 <p>The diagram shows the timing of various signals over time. The CNT_REG signal is a sequence of hex digits: 94, 95, 96, 97, 98, 99. The CI0 signal is asserted during the pause period. The TRGIF signal is asserted after a delay relative to the CI0 signal.</p>	

	<b>Mode Selection</b>	<b>Source Selection</b>	<b>Polarity Selection</b>	<b>Filter and Prescaler</b>
Exam3	Event mode The counter will start to count when a rising trigger input. <small>ETIF is the selection.</small>	TRGS[2:0]=3'b11 1	ETP = 0 no polarity change.	ETPSC = 1, divided by 2. ETFC = 0 , no filter

**Figure 16-43. Event mode**



### Single pulse mode

Refer to [Single pulse mode](#).

### Timers interconnection

Refer to [Advanced timer \(TIMERx, x=0, 7\)](#).

### Timer DMA mode

Timer's DMA mode is the function that configures timer's register by DMA module. The relative registers are TIMERx\_DMACFG and TIMERx\_DMATB; Of course, you have to enable a DMA request which will be asserted by some internal interrupt event. When the interrupt event was asserted, TIMERx will send a request to DMA, which is configured to M2P mode and PADDR is TIMERx\_DMATB, then DMA will access the TIMERx\_DMATB. In fact, register TIMERx\_DMATB is only a buffer; timer will map the TIMERx\_DMATB to an internal register, appointed by the field of DMATA in TIMERx\_DMACFG . If the field of DMATC in TIMERx\_DMACFG is 0(1 transfer), then the timer's DMA request is finished. While if TIMERx\_DMATC is not 0, such as 3( 4 transfers), then timer will send 3 more requests to DMA, and DMA will access timer's registers DMASAR+0x4, DMASAR+0x8, DMASAR+0xc at the next 3 accesses to TIMERx\_DMATB. In one word, one time DMA internal interrupt event assert, DMATC+1 times request will be send by TIMERx.

If one more time DMA request event coming, TIMERx will repeat the process as above.

### Timer debug mode

When the Cortex®-M4 halted, and the TIMERx\_HOLD configuration bit in DBG\_CTL0 register

set to 1, the TIMERx counter stops.

### 16.2.5. TIMERx registers(x=1, 2, 3, 4)

TIMER1 base address: 0x4000 0000

TIMER2 base address: 0x4000 0400

TIMER3 base address: 0x4000 0800

TIMER4 base address: 0x4000 0C00

#### 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 factor between the CK_TIMER and the dead-time and digital filter sample clock (DTS). 00: $f_{DTS}=f_{CK\_TIMER}$ 01: $f_{DTS}= f_{CK\_TIMER} /2$ 10: $f_{DTS}= f_{CK\_TIMER} /4$ 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 counting down, CHxF bit 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 counting up, CHxF bit 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 counting up and counting down, CHxF bit

		can be set. After the counter is enabled, cannot be switched from 0x00 to non 0x00.
4	DIR	<p>Direction</p> <p>0: Count up</p> <p>1: Count down</p> <p>If the timer work in center-aligned mode or encoder mode, this bit is read only.</p>
3	SPM	<p>Single pulse mode.</p> <p>0: Single pulse mode disable. The counter continues after update event.</p> <p>1: Single pulse mode enable. The counter counts until the next update event occurs.</p>
2	UPS	<p>Update source</p> <p>This bit is used to select the update event sources by software.</p> <p>0: These events generate update interrupts or DMA requests:</p> <ul style="list-style-type: none"> <li>The UPG bit is set</li> <li>The counter generates an overflow or underflow event</li> <li>The restart mode generates an update event.</li> </ul> <p>1: This event generates update interrupts or DMA requests:</p> <p>The counter generates an overflow or underflow event</p>
1	UPDIS	<p>Update disable.</p> <p>This bit is used to enable or disable the update event generation.</p> <p>0: Update event enable. When an update event occurs, the corresponding shadow registers are loaded with their preloaded values. These events generate update event:</p> <ul style="list-style-type: none"> <li>The UPG bit is set</li> <li>The counter generates an overflow or underflow event</li> <li>The restart mode generates an update event.</li> </ul> <p>1: Update event disable.</p> <p><b>Note:</b> When this bit is set to 1, setting UPG bit or the restart mode does not generate an update event, but the counter and prescaler are initialized.</p>
0	CEN	<p>Counter enable</p> <p>0: Counter disable</p> <p>1: Counter enable</p> <p>The CEN bit must be set by software when timer works in external clock, pause mode and encoder mode.</p>

### **Control register 1 (TIMERx\_CTL1)**

Address offset: 0x04

Reset value: 0x0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				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: When a counter reset event occurs, a TRGO trigger signal is output. The counter reset source:</p> <ul style="list-style-type: none"> <li>Master timer generate a reset</li> <li>the UPG bit in the TIMERx_SWEVG register is set</li> </ul> <p>001: Enable. When a counter start event occurs, a TRGO trigger signal is output. The counter start source :</p> <ul style="list-style-type: none"> <li>CEN control bit is set</li> <li>The trigger input in pause mode is high</li> </ul> <p>010: When an update event occurs, a TRGO trigger signal is output. The update source depends on UPDIS bit and UPS bit.</p> <p>011: When a capture or compare pulse event occurs in channel0, a TRGO trigger signal is output.</p> <p>100: When a compare event occurs, a TRGO trigger signal is output. The compare source is from O0CPRE.</p> <p>101: When a compare event occurs, a TRGO trigger signal is output. The compare source is from O1CPRE.</p> <p>110: When a compare event occurs, a TRGO trigger signal is output. The compare source is from O2CPRE.</p> <p>111: When a compare event occurs, a TRGO trigger signal is output. The compare source is from O3CPRE.</p>
3	DMAS	<p>DMA request source selection</p> <p>0: When capture or compare event occurs, the DMA request of channel x is sent</p> <p>1: When update event occurs, the DMA request of channel x is sent.</p>
2:0	Reserved	Must be kept at reset value.

### Slave mode configuration register (TIMERx\_SMCFG)

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
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

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15	ETP	<p>External trigger polarity</p> <p>This bit specifies the polarity of ETI signal</p> <p>0: ETI is active at rising edge or high level .</p> <p>1: ETI is active at falling edge or low level .</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>When the slave mode is configured as restart mode, pause mode or event mode, the timer can still work in the external clock 1 mode by setting this bit. But the TRGS bits must not be 3'b111 in this case.</p> <p>The clock source of the timer will be ETIFP if external clock mode 0 and external clock mode 1 are configured at the same time.</p> <p><b>Note:</b> External clock mode 0 enable is in this register's SMC[2:0] bit-field.</p>
13:12	ETPSC[1:0]	<p>The prescaler of external trigger</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: The prescaler is 2.</p> <p>10: The prescaler is 4.</p> <p>11: The prescaler is 8.</p>
11:8	ETFC[3:0]	<p>External trigger filter control</p> <p>The external trigger can be filtered by digital filter and this bit-field configure the filtering capability.</p> <p>Basic principle of digital filter: continuously sample the external trigger signal according to <math>f_{SAMP}</math> and record the number of times of the same level of the signal.</p> <p>After reaching the filtering capacity configured by this bit-field, it is considered to be an effective level.</p> <p>The filtering capability configuration is as follows:</p>

<b>EXTFC[3:0]</b>	<b>Times</b>	<b>f<sub>SAMP</sub></b>
4'b0000	Filter disabled.	
4'b0001	2	f <sub>CK_TIMER</sub>
4'b0010	4	
4'b0011	8	
4'b0100	6	f <sub>DTS_CK</sub> /2
4'b0101	8	
4'b0110	6	f <sub>DTS_CK</sub> /4
4'b0111	8	
4'b1000	6	f <sub>DTS_CK</sub> /8
4'b1001	8	
4'b1010	5	f <sub>DTS_CK</sub> /16
4'b1011	6	
4'b1100	8	
4'b1101	5	f <sub>DTS_CK</sub> /32
4'b1110	6	
4'b1111	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: ITI0 001: ITI1 010: ITI2 011: ITI3 100: CI0F_ED 101: CI0FE0 110: CI1FE1 111: 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 CI0FE0 edge, while the direction depends on CI1FE1 level. 010: Quadrature decoder mode 1. The counter counts on CI1FE1 edge, while the

direction depends on CI0FE0 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 an update event is generated 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 clock when it is low.

110: Event mode. A rising edge of the trigger input enables the counter.

111: External clock mode 0. The counter counts on the rising edges of the selected trigger.

### DMA and interrupt enable register (TIMERx\_DMAINTEN)

Address offset: 0x0C

Reset value: 0x0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	TRGDEN	Reserved	CH3DEN	CH2DEN	CH1DEN	CH0DEN	UPDEN	Reserved	TRGIE	Reserved	CH3IE	CH2IE	CH1IE	CH0IE	UPIE

rw                    rw

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

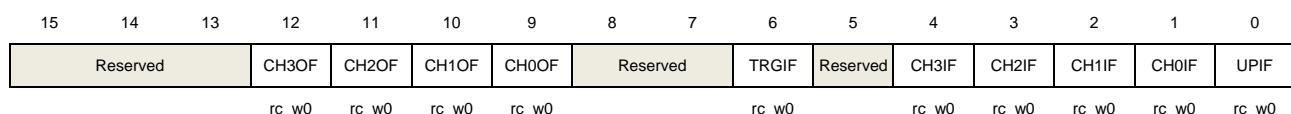
		0: disabled 1: enabled
7	Reserved	Must be kept at reset value.
6	TRGIE	Trigger interrupt enable 0: disabled 1: enabled
5	Reserved	Must be kept at reset value.
4	CH3IE	Channel 3 capture/compare interrupt enable 0: disabled 1: enabled
3	CH2IE	Channel 2 capture/compare interrupt enable 0: disabled 1: enabled
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)



<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
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 on trigger event and cleared by software. When in pause mode, both edges on trigger input generates a trigger event, otherwise, only an active edge on trigger input can 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. If Channel0 is set to input mode, this bit will be reset by reading TIMERx_CH0CV. 0: No Channel 0 interrupt occurred 1: Channel 0 interrupt occurred
0	UPIF	Update interrupt flag This bit is set by hardware on an update event and cleared by software. 0: No update interrupt occurred 1: Update interrupt occurred

### Software event generation register (TIMERx\_SWEVG)

Address offset: 0x14

Reset value: 0x0000

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

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

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]	CH0CAPFLT[3:0]	CH0CAPFLT[3:0]	CH0COM SEN	CH0COM FEN	CH0MS[1:0]	CH0CAPPSC[1:0]	CH0CAPPSC[1:0]	CH0MS[1:0]	CH0MS[1:0]
CH1CAPFLT[3:0]	CH1CAPPSC[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 CH0COMFEN 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 programmed as output mode 01: Channel 1 is programmed as input mode, IS1 is connected to CI1FE1 10: Channel 1 is programmed as input mode, IS1 is connected to CI0FE1 11: Channel 1 is programmed as input mode, IS1 is connected to ITS.  <b>Note:</b> When CH1MS[1:0]=11, it is necessary to select an internal trigger input through TRGS bits in TIMERx_SMCFG register.
7	CH0COMCEN	Channel 0 output compare clear enable.  When this bit is set, if the ETIPP signal is detected as high level, the O0CPRE signal will be cleared. 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 specifies the compare output mode of the the output prepare signal O0CPRE. In addition, the high level of O0CPRE is the active level, and CH0_O and CH0_ON channels polarity depends on CH0P and CH0NP 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 is equals to the output compare register TIMERx_CH0CV. 010: Clear the channel output. O0CPRE signal is forced low when the counter is

equals to the output compare register TIMERx\_CH0CV.

011: Toggle on match. O0CPRE toggles when the counter is equals to the output compare register TIMERx\_CH0CV.

100: Force low. O0CPRE is forced to low level.

101: Force high. O0CPRE is forced to high level.

110: PWM mode0. When counting up, O0CPRE is high when the counter is smaller than TIMERx\_CH0CV, and low otherwise. When counting down, O0CPRE is low when the counter is larger than TIMERx\_CH0CV, and high otherwise.

111: PWM mode1. When counting up, O0CPRE is low when the counter is smaller than TIMERx\_CH0CV, and high otherwise. When counting down, O0CPRE is high when the counter is larger than TIMERx\_CH0CV, and low otherwise.

If configured in PWM mode, the O0CPRE level changes only when the output compare mode is adjusted from “Timing” mode to “PWM” mode or the comparison result changes.

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 verifying the shadow register only in single pulse mode (when SPM=1)</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. 1: Channel 0 output quickly compare enable.</p>
1:0	CH0MS[1:0]	<p>Channel 0 I/O mode selection</p> <p>This bit-field specifies the work mode of the channel and the input signal selection. This bit-field is writable only when the channel is not active. (CH0EN bit in TIMERx_CHCTL2 register is reset).).</p> <p>00: Channel 0 is programmed as output mode 01: Channel 0 is programmed as input mode, IS0 is connected to CI0FE0 10: Channel 0 is programmed as input mode, IS0 is connected to CI1FE0 11: Channel 0 is programmed as input mode, IS0 is connected to ITS</p> <p><b>Note:</b> When CH0MS[1:0]=11, it is necessary to select an internal trigger input through TRGS bits in TIMERx_SMCFG register.</p>

#### **Input capture mode:**

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

15:12	CH1CAPFLT[3:0]	Channel 1 input capture filter control Refer to CH0CAPFLT description																																										
11:10	CH1CAPPSC[1:0]	Channel 1 input capture prescaler Refer to CH0CAPPSC description																																										
9:8	CH1MS[1:0]	Channel 1 mode selection Same as Output compare mode																																										
7:4	CH0CAPFLT[3:0]	Channel 0 input capture filter control The CI0 input signal can be filtered by digital filter and this bit-field configure the filtering capability. Basic principle of digital filter: continuously sample the CI0 input signal according to $f_{SAMP}$ and record the number of times of the same level of the signal. After reaching the filtering capacity configured by this bit, it is considered to be an effective level. The filtering capability configuration is as follows:																																										
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr style="background-color: #a0c0ff;"> <th style="padding: 2px;">CH0CAPFLT [3:0]</th> <th style="padding: 2px;">Times</th> <th style="padding: 2px;"><math>f_{SAMP}</math></th> </tr> </thead> <tbody> <tr> <td style="padding: 2px;">4'b0000</td> <td style="padding: 2px;"></td> <td style="padding: 2px;">Filter disabled.</td> </tr> <tr> <td style="padding: 2px;">4'b0001</td> <td style="padding: 2px; text-align: center;">2</td> <td rowspan="3" style="padding: 2px; vertical-align: middle;"><math>f_{CK\_TIMER}</math></td> </tr> <tr> <td style="padding: 2px;">4'b0010</td> <td style="padding: 2px; text-align: center;">4</td> </tr> <tr> <td style="padding: 2px;">4'b0011</td> <td style="padding: 2px; text-align: center;">8</td> </tr> <tr> <td style="padding: 2px;">4'b0100</td> <td style="padding: 2px; text-align: center;">6</td> <td rowspan="2" style="padding: 2px; vertical-align: middle;"><math>f_{DTS}/2</math></td> </tr> <tr> <td style="padding: 2px;">4'b0101</td> <td style="padding: 2px; text-align: center;">8</td> </tr> <tr> <td style="padding: 2px;">4'b0110</td> <td style="padding: 2px; text-align: center;">6</td> <td rowspan="2" style="padding: 2px; vertical-align: middle;"><math>f_{DTS}/4</math></td> </tr> <tr> <td style="padding: 2px;">4'b0111</td> <td style="padding: 2px; text-align: center;">8</td> </tr> <tr> <td style="padding: 2px;">4'b1000</td> <td style="padding: 2px; text-align: center;">6</td> <td rowspan="2" style="padding: 2px; vertical-align: middle;"><math>f_{DTS}/8</math></td> </tr> <tr> <td style="padding: 2px;">4'b1001</td> <td style="padding: 2px; text-align: center;">8</td> </tr> <tr> <td style="padding: 2px;">4'b1010</td> <td style="padding: 2px; text-align: center;">5</td> <td rowspan="3" style="padding: 2px; vertical-align: middle;"><math>f_{DTS}/16</math></td> </tr> <tr> <td style="padding: 2px;">4'b1011</td> <td style="padding: 2px; text-align: center;">6</td> </tr> <tr> <td style="padding: 2px;">4'b1100</td> <td style="padding: 2px; text-align: center;">8</td> </tr> <tr> <td style="padding: 2px;">4'b1101</td> <td style="padding: 2px; text-align: center;">5</td> <td rowspan="3" style="padding: 2px; vertical-align: middle;"><math>f_{DTS}/32</math></td> </tr> <tr> <td style="padding: 2px;">4'b1110</td> <td style="padding: 2px; text-align: center;">6</td> </tr> <tr> <td style="padding: 2px;">4'b1111</td> <td style="padding: 2px; text-align: center;">8</td> </tr> </tbody> </table>	CH0CAPFLT [3:0]	Times	$f_{SAMP}$	4'b0000		Filter disabled.	4'b0001	2	$f_{CK\_TIMER}$	4'b0010	4	4'b0011	8	4'b0100	6	$f_{DTS}/2$	4'b0101	8	4'b0110	6	$f_{DTS}/4$	4'b0111	8	4'b1000	6	$f_{DTS}/8$	4'b1001	8	4'b1010	5	$f_{DTS}/16$	4'b1011	6	4'b1100	8	4'b1101	5	$f_{DTS}/32$	4'b1110	6	4'b1111	8	
CH0CAPFLT [3:0]	Times	$f_{SAMP}$																																										
4'b0000		Filter disabled.																																										
4'b0001	2	$f_{CK\_TIMER}$																																										
4'b0010	4																																											
4'b0011	8																																											
4'b0100	6	$f_{DTS}/2$																																										
4'b0101	8																																											
4'b0110	6	$f_{DTS}/4$																																										
4'b0111	8																																											
4'b1000	6	$f_{DTS}/8$																																										
4'b1001	8																																											
4'b1010	5	$f_{DTS}/16$																																										
4'b1011	6																																											
4'b1100	8																																											
4'b1101	5	$f_{DTS}/32$																																										
4'b1110	6																																											
4'b1111	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, input capture occurs on every channel input edge 01: The input capture occurs on every 2 channel input edges 10: The input capture occurs on every 4 channel input edges 11: The input capture occurs on 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:**

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
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 programmed as output mode 01: Channel 3 is programmed as input mode, IS3 is connected to CI3FE3 10: Channel 3 is programmed as input mode, IS3 is connected to CI2FE3 11: Channel 3 is programmed as input mode, IS3 is connected to ITS.  <b>Note:</b> When CH3MS[1:0]=11, it is necessary to select an internal trigger input through TRGS bits in TIMERx_SMCFG register.
7	CH2COMCEN	Channel 2 output compare clear enable.  When this bit is set, if the ETIFP signal is detected as high level, the O2CPRE signal will be cleared.  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 specifies the compare output mode of the the output prepare signal O0CPRE. In addition, the high level of O0CPRE is the active level, and CH0_O and CH0_ON channels polarity depends on CH0P and CH0NP 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 is equals to the output compare register `TIMERx_CH2CV`.

010: Clear the channel output. `O2CPRE` signal is forced low when the counter is equals to the output compare register `TIMERx_CH2CV`.

011: Toggle on match. `O2CPRE` toggles when the counter is equals to the output compare register `TIMERx_CH2CV`.

100: Force low. `O2CPRE` is forced to low level.

101: Force high. `O2CPRE` is forced to high level.

110: PWM mode 0. When counting up, `O2CPRE` is high when the counter is smaller than `TIMERx_CH2CV`, and low otherwise. When counting down, `O2CPRE` is low when the counter is larger than `TIMERx_CH2CV`, and high otherwise.

111: PWM mode 1. When counting up, `O2CPRE` is low when the counter is smaller than `TIMERx_CH2CV`, and high otherwise. When counting down, `O2CPRE` is high when the counter is larger than `TIMERx_CH2CV`, and low otherwise.

If configured in PWM mode, the `O2CPRE` level changes only when the output compare mode is adjusted from “Timing” mode to “PWM” mode or the comparison result changes.

3	<b>CH2COMSEN</b>	<p>Channel 2 compare output shadow enable</p> <p>When this bit is set, the shadow register of <code>TIMERx_CH2CV</code> register, which updates at each update event will be enabled.</p> <p>0: Channel 2 output compare shadow disable 1: Channel 2 output compare shadow enable</p> <p>The PWM mode can be used without verifying the shadow register only in single pulse mode (when <code>SPM=1</code>)</p>
2	<b>CH2COMFEN</b>	<p>Channel 2 output compare fast enable</p> <p>When this bit is set, the effect of an event on the trigger in input on the capture/compare output will be accelerated if the channel is configured in PWM1 or PWM2 mode. The output channel will treat an active edge on the trigger input as a compare match, and <code>CH2_O</code> is set to the compare level independently from the result of the comparison.</p> <p>0: Channel 2 output quickly compare disable. 1: Channel 2 output quickly compare enable.</p>
1:0	<b>CH2MS[1:0]</b>	<p>Channel 2 I/O mode selection</p> <p>This bit-field specifies the work mode of the channel and the input signal selection. This bit-field is writable only when the channel is not active. (<code>CH2EN</code> bit in <code>TIMERx_CHCTL2</code> register is reset).).</p> <p>00: Channel 2 is programmed as output mode 01: Channel 2 is programmed as input mode, <code>IS2</code> is connected to <code>CI2FE2</code> 10: Channel 2 is programmed as input mode, <code>IS2</code> is connected to <code>CI3FE2</code> 11: Channel 2 is programmed as input mode, <code>IS2</code> is connected to <code>ITS</code>.</p>

**Note:** When CH2MS[1:0]=11, it is necessary to select an internal trigger input 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 The CI2 input signal can be filtered by digital filter and this bit-field configure the filtering capability. Basic principle of digital filter: continuously sample the CI2 input signal according to $f_{SAMP}$ and record the number of times of the same level of the signal. After reaching the filtering capacity configured by this bit, it is considered to be an effective level. The filtering capability configuration is as follows:																																										
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr style="background-color: #d9e1f2;"> <th>CH2CAPFLT [3:0]</th> <th>Times</th> <th><math>f_{SAMP}</math></th> </tr> </thead> <tbody> <tr> <td>4'b0000</td> <td colspan="2" style="text-align: center;">Filter disabled.</td> </tr> <tr> <td>4'b0001</td> <td style="text-align: center;">2</td> <td rowspan="3" style="vertical-align: middle; text-align: center;"><math>f_{CK\_TIMER}</math></td></tr> <tr> <td>4'b0010</td> <td style="text-align: center;">4</td> </tr> <tr> <td>4'b0011</td> <td style="text-align: center;">8</td> </tr> <tr> <td>4'b0100</td> <td style="text-align: center;">6</td> <td rowspan="2" style="vertical-align: middle; text-align: center;"><math>f_{DTS}/2</math></td></tr> <tr> <td>4'b0101</td> <td style="text-align: center;">8</td> </tr> <tr> <td>4'b0110</td> <td style="text-align: center;">6</td> <td rowspan="2" style="vertical-align: middle; text-align: center;"><math>f_{DTS}/4</math></td></tr> <tr> <td>4'b0111</td> <td style="text-align: center;">8</td> </tr> <tr> <td>4'b1000</td> <td style="text-align: center;">6</td> <td rowspan="2" style="vertical-align: middle; text-align: center;"><math>f_{DTS}/8</math></td></tr> <tr> <td>4'b1001</td> <td style="text-align: center;">8</td> </tr> <tr> <td>4'b1010</td> <td style="text-align: center;">5</td> <td rowspan="3" style="vertical-align: middle; text-align: center;"><math>f_{DTS}/16</math></td></tr> <tr> <td>4'b1011</td> <td style="text-align: center;">6</td> </tr> <tr> <td>4'b1100</td> <td style="text-align: center;">8</td> </tr> <tr> <td>4'b1101</td> <td style="text-align: center;">5</td> <td rowspan="3" style="vertical-align: middle; text-align: center;"><math>f_{DTS}/32</math></td></tr> <tr> <td>4'b1110</td> <td style="text-align: center;">6</td> </tr> <tr> <td>4'b1111</td> <td style="text-align: center;">8</td> </tr> </tbody> </table>		CH2CAPFLT [3:0]	Times	$f_{SAMP}$	4'b0000	Filter disabled.		4'b0001	2	$f_{CK\_TIMER}$	4'b0010	4	4'b0011	8	4'b0100	6	$f_{DTS}/2$	4'b0101	8	4'b0110	6	$f_{DTS}/4$	4'b0111	8	4'b1000	6	$f_{DTS}/8$	4'b1001	8	4'b1010	5	$f_{DTS}/16$	4'b1011	6	4'b1100	8	4'b1101	5	$f_{DTS}/32$	4'b1110	6	4'b1111	8
CH2CAPFLT [3:0]	Times	$f_{SAMP}$																																										
4'b0000	Filter disabled.																																											
4'b0001	2	$f_{CK\_TIMER}$																																										
4'b0010	4																																											
4'b0011	8																																											
4'b0100	6	$f_{DTS}/2$																																										
4'b0101	8																																											
4'b0110	6	$f_{DTS}/4$																																										
4'b0111	8																																											
4'b1000	6	$f_{DTS}/8$																																										
4'b1001	8																																											
4'b1010	5	$f_{DTS}/16$																																										
4'b1011	6																																											
4'b1100	8																																											
4'b1101	5	$f_{DTS}/32$																																										
4'b1110	6																																											
4'b1111	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, input capture occurs on every channel input edge 01: The input capture occurs on every 2 channel input edges 10: The input capture occurs on every 4 channel input edges																																										

11: The input capture occurs on 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 high level is active level 1: Channel 0 low level is active level When channel 0 is configured in input mode, this bit specifies the CI0 signal polarity.

[CH0P==0]: ClxFE0's rising edge is the active signal for capture or trigger operation in slave mode. And ClxFE0 will not be inverted.

[CH0P==1]: ClxFE0's falling edge is the active signal for capture or trigger operation in slave mode. And ClxFE0 will be 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															

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 TIMER_CK 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><b>Note:</b> 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-filed indicates the counter value corresponding to the last capture event. And this bit-filed is read-only.</p> <p>When channel 0 is configured in output mode, this bit-filed contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates every update event.</p>

### Channel 1 capture/compare value register (TIMERx\_CH1CV)

Address offset: 0x38

Reset value: 0x0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH1VAL[15:0]															

rw

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

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

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

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 filed defines the number(n) of the register that DMA will access(R/W), n = (DMATC [4:0] +1). DMATC [4:0] is from 5'b0_0000 to 5'b1_0001.
7:5	Reserved	Must be kept at reset value.
4:0	DMATA [4:0]	DMA transfer access start address This filed define the first address for the DMA access the TIMERx_DMATB. When access is done through the TIMERx_DMA address first time, this bit-field specifies the address you just access. And then the second access to the TIMERx_DMATB, you will access the address of start address + 0x4.

### 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															

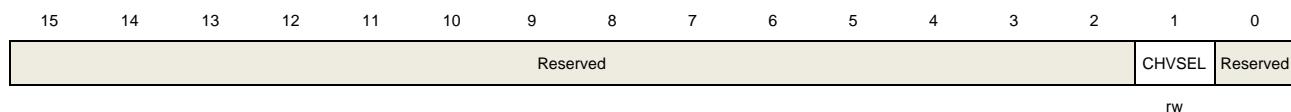
Bits	Fields	Descriptions
15:0	DMATB[15:0]	DMA transfer buffer When a read or write operation is assigned to this register, the register located at the address range (Start Addr + Transfer Timer* 4) will be accessed. The transfer Timer is calculated by hardware, and ranges from 0 to DMATC.

### **Configuration register (TIMERx\_CFG )**

Address offset: 0xFC

Reset value: 0x0000

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



<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
15:2	Reserved	Must be kept at reset value.
1	CHVSEL	Write CHxVAL register selection This bit-field set and reset by software. 1: If write the CHxVAL register, the write value is same as the CHxVAL value, the write access ignored 0: No effect
0	Reserved	Must be kept at reset value.

## 16.3. General level1 timer (TIMERx, x=8, 11)

### 16.3.1. Overview

The general level1 timer module (Timer8, 11) is a two-channel timer that supports input capture, output compare. They can generate PWM signals to control motor or be used for power management applications. The general level1 time reference is a 16-bit counter that can be used as an unsigned counter.

In addition, the general level1 timers can be programmed and be used to count or time external events that drive other Timers.

Timer and timer are completely independent, but there may be synchronized to provide a larger timer with their counters incrementing in unison.

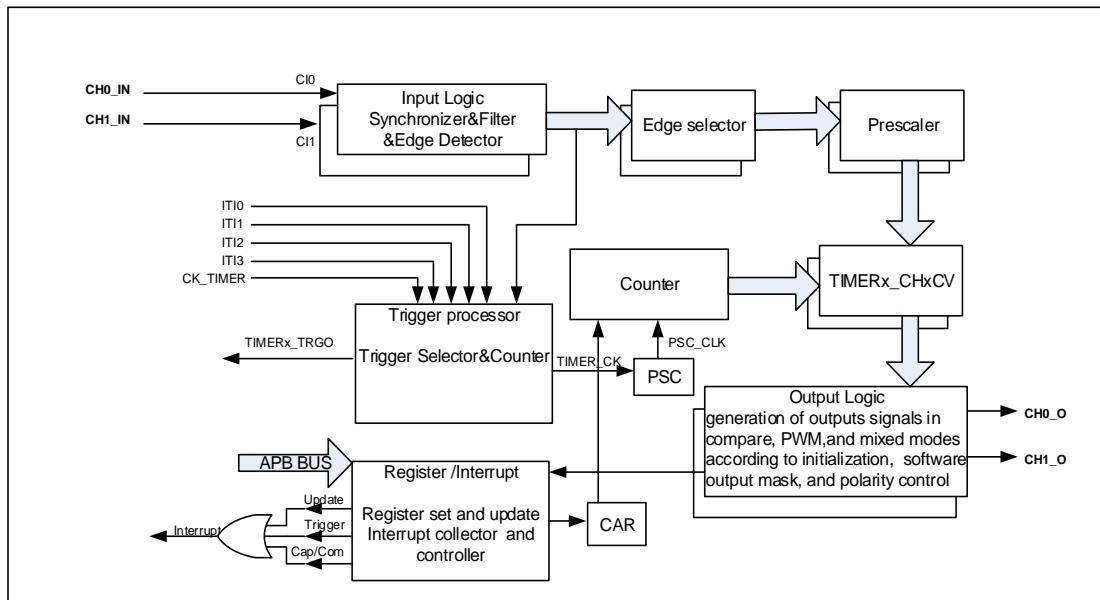
### 16.3.2. Characteristics

- Total channel num: 2.
- Counter width: 16 bits.
- Source of count clock is selectable: internal clock, internal trigger, external input.
- counter mode: count up only.
- Programmable prescaler: 16 bit. Factor can be changed on the go.
- Each channel is user-configurable: Input capture mode, Output compare mode, Programmable PWM mode, Single pulse mode
- Auto-reload function.
- Interrupt output on: update, trigger event, and compare/capture event.
- Daisy chaining of timer modules to allow a single timer to initiate multiple timing events.
- Timer synchronization allows selected timers to start counting on the same clock cycle.
- Timer master-slave management.

### 16.3.3. Block diagram

[\*\*Figure 16-44. General level1 timer block diagram\*\*](#) provides details on the internal configuration of the general level1 timer.

**Figure 16-44. General level1 timer block diagram**



### 16.3.4. Function overview

#### Clock source configuration

The general level1 TIMER has the capability of being clocked by either the CK\_TIMER or an alternate clock source controlled by SMC (TIMERx\_SMCFG bit [2:0]).

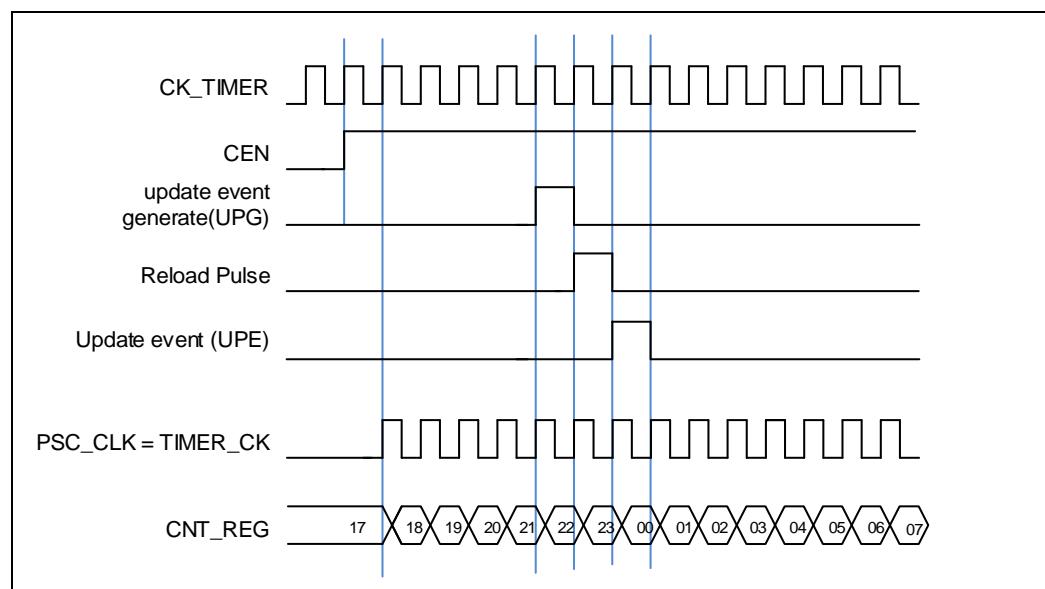
- SMC [2:0] == 3'b000. Internal timer clock CK\_TIMER which is from module RCU.

The default internal clock source is the CK\_TIMER used to drive the counter prescaler when SMC [2:0] == 3'b000. When the CEN is set, the CK\_TIMER will be divided by PSC value to generate PSC\_CLK.

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

If the SMC [2:0] in the TIMERx\_SMCFG register are setting to an available value including 0x1, 0x2, 0x3 and 0x7, the prescaler is clocked by other clock sources selected by the TRGS [2:0] in the TIMERx\_SMCFG register and described as follows. When the SMC bits are set to 0x4, 0x5 or 0x6, the internal clock CK\_TIMER is the counter prescaler driving clock source.

**Figure 16-45. Timing chart of internal clock divided by 1**



- SMC [2:0] == 3'b111 (external clock mode 0). External input pin source

The TIMER\_CK, driven counter's prescaler to count, can be triggered by the event of rising or falling edge on the external pin TIMERx\_CI0/TIMERx\_CI1. This mode can be selected by setting SMC [2:0] to 0x7 and the TRGS [2:0] to 0x4, 0x5 or 0x6.

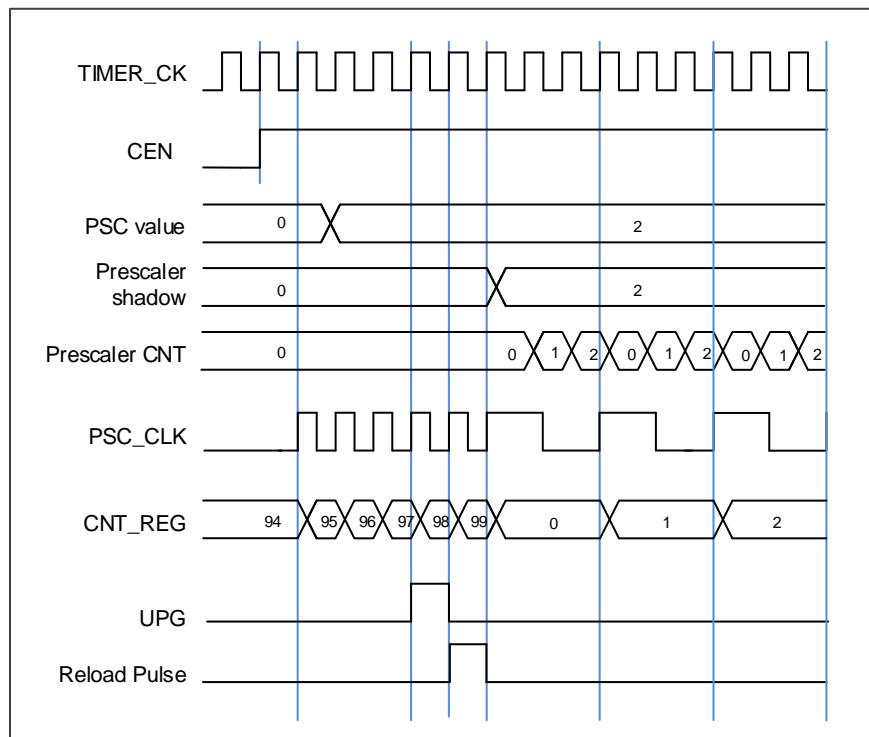
And, the counter prescaler can also be driven by rising edge on the internal trigger input pin ITI0/1/2/3. This mode can be selected by setting SMC [2:0] to 0x7 and the TRGS [2:0] to 0x0,

0x1, 0x2 or 0x3.

### Clock prescaler

The counter clock (PSC\_CK) is obtained by the TIMER\_CK through the prescaler, and the prescale factor can be configured from 1 to 65536 through the prescaler register (TIMERx\_PSC). The new written prescaler value will not take effect until the next update event.

**Figure 16-46. Timing chart of PSC value change from 0 to 2**



### Counter up counting

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 will start counting up from 0 again. The update event is generated at each counter overflow. The counting direction bit DIR in the TIMERx\_CTL1 register should be set to 0 for the up counting mode.

When the update event is set by the UPG bit in the TIMERx\_SWEVG register, the counter value will be initialized to 0 and generates an update event.

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

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

The following figures show some examples of the counter behavior for different clock prescaler factor when TIMERx\_CAR=0x99.

Figure 16-47. Timing chart of up counting mode, PSC=0/2

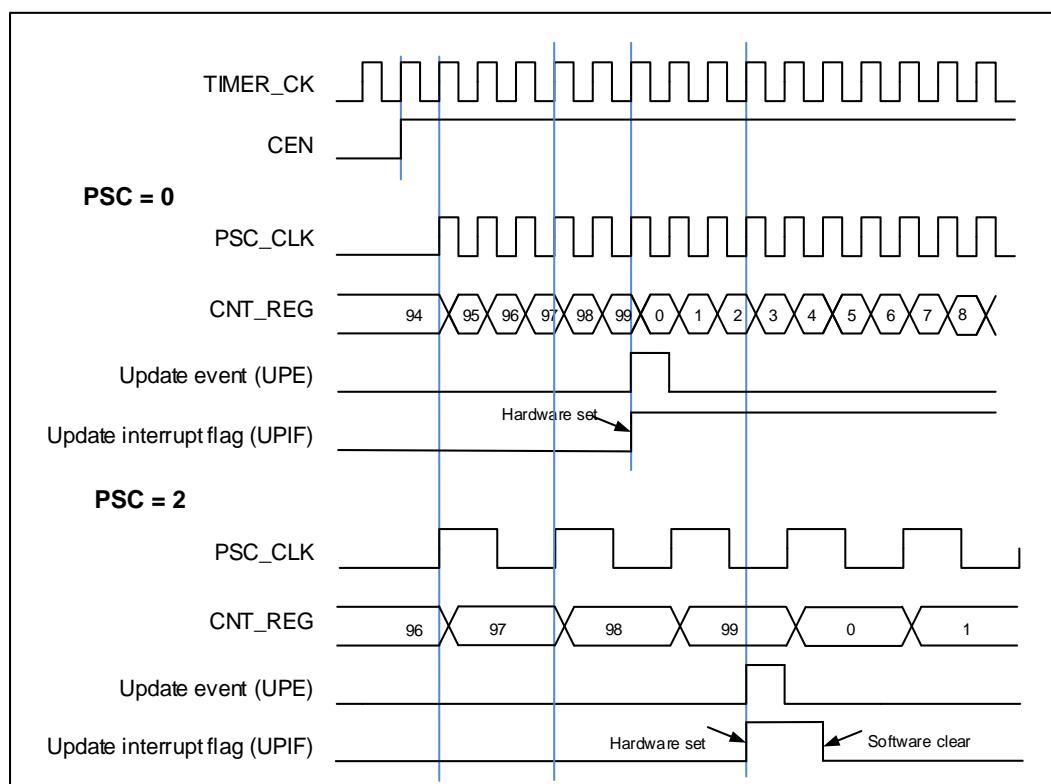
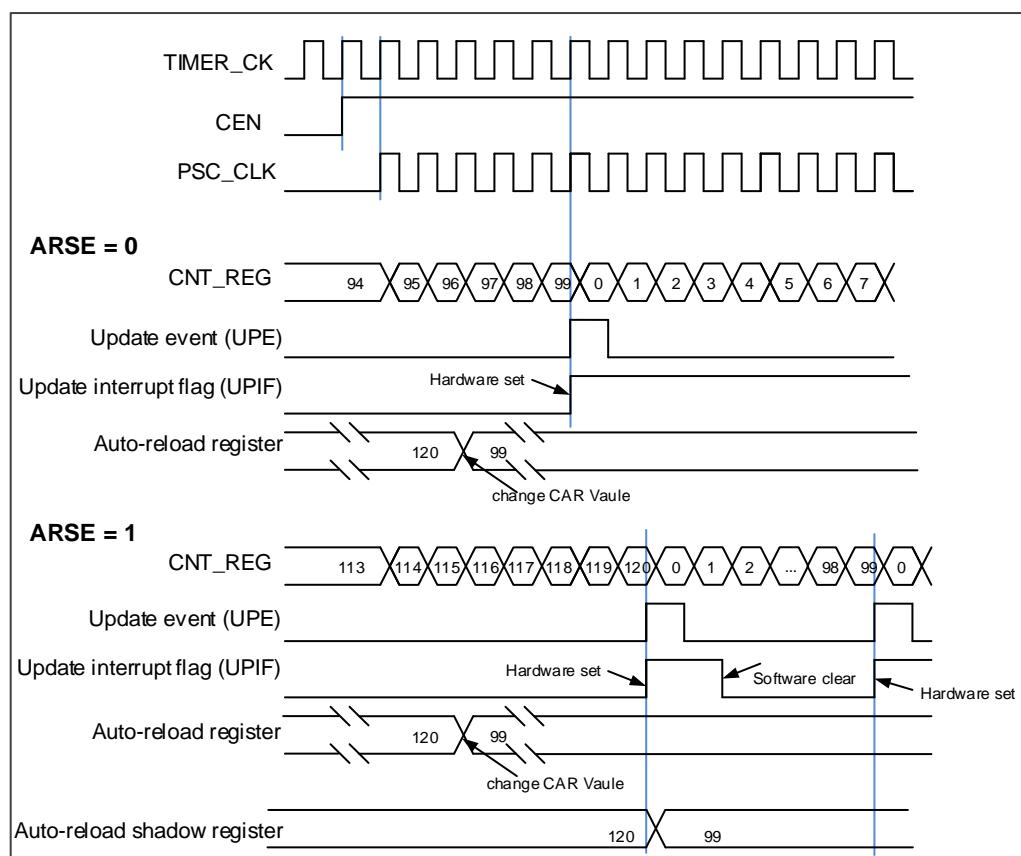


Figure 16-48. Timing chart of up counting mode, change TIMERx\_CAR ongoing



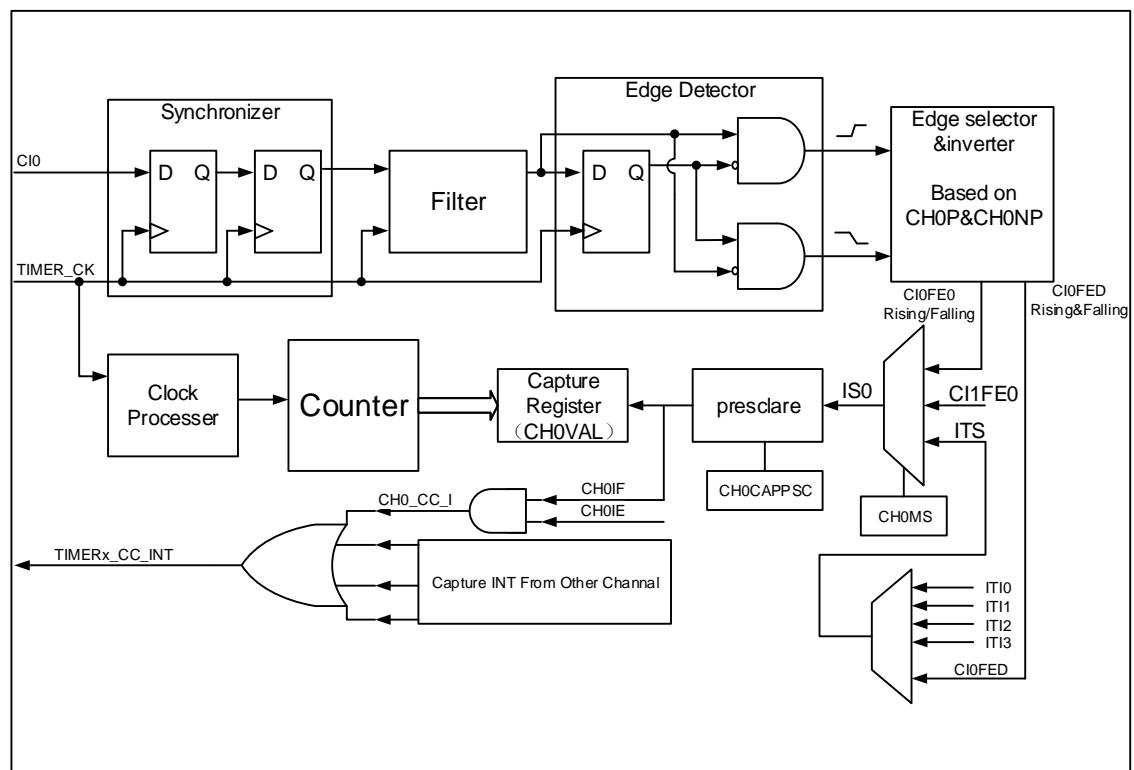
## Input capture and output compare channels

The general level1 timer has two independent channels which can be used as capture inputs or compare match outputs. Each channel is built around a channel capture compare register including an input stage, channel controller and an output stage.

### ■ Channel input capture function

Channel input capture function allows the channel to perform measurements such as pulse timing, frequency, period, duty cycle and so on. The input stage consists of a digital filter, a channel polarity selection, edge detection and a channel prescaler. When a selected edge occurs on the channel input, the current value of the counter is captured into the TIMERx\_CHxCV register, at the same time the CHxIF bit is set and the channel interrupt is generated if enabled by CHxIE = 1.

**Figure 16-49. Channel input capture principle**



First, the channel input signal (CIx) is synchronized to TIMER\_CK domain, and then sampled by a digital filter to generate a filtered input signal. Then through the edge detector, the rising and fall edge are detected. You can select one of them by CHxP. One more selector is for the other channel and trig, controlled by CHxMS. The IC\_prescaler make several the input event generate one effective capture event. On the capture event, TIMERx\_CHxCV will restore the value of counter.

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

**Step1:** Filter configuration. (CHxCAPFLT in TIMERx\_CHCTL0)

Based on the input signal and requested signal quality, configure compatible CHxCAPFLT.

**Step2:** Edge selection. (CHxP/CHxNP in TIMERx\_CHCTL2)

Rising or falling edge, choose one by CHxP/CHxNP.

**Step3:** Capture source selection. (CHxMS in TIMERx\_CHCTL0)

As soon as you select one input capture source by CHxMS, you have set the channel to input mode (CHxMS!=0x0) and TIMERx\_CHxCV cannot be written any more.

**Step4:** Interrupt enable. (CHxIE and CHxDEN in TIMERx\_DMAINTEN)

Enable the related interrupt enable; you can get the interrupt and DMA request.

**Step5:** Capture enables. (CHxEN in TIMERx\_CHCTL2)

**Result:** When you wanted input signal is got, TIMERx\_CHxCV will be set by Counter's value. And CHxIF is asserted. If the CHxIF is high, the CHxOF will be asserted also. The interrupt and DMA request will be asserted based on your configuration of CHxIE and CHxDEN in TIMERx\_DMAINTEN

**Direct generation:** If you want to generate a DMA request or Interrupt, you can set CHxG by software directly.

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

## ■ Channel output compare function

In channel output compare function, the TIMERx can generate timed pulses with programmable position, polarity, duration, and frequency. When the counter matches the value in the CHxVAL register of an output compare channel, the channel (n) output can be set, cleared, or toggled based on CHxCOMCTL. When the counter reaches the value in the CHxVAL register, the CHxIF bit is set and the channel (n) interrupt is generated if CHxIE = 1. And the DMA request will be assert, if CxCDE=1.

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

**Step1:** Clock configuration. Such as clock source, clock prescaler and so on.

**Step2:** Compare mode configuration.

- \* Set the shadow enable mode by CHxCOMSEN
- \* Set the output mode (Set/Clear/Toggle) by CHxCOMCTL.
- \* Select the active high polarity by CHxP/CHxNP
- \* Enable the output by CHxEN

**Step3:** Interrupt/DMA-request enables configuration by CHxIE/CxCDE

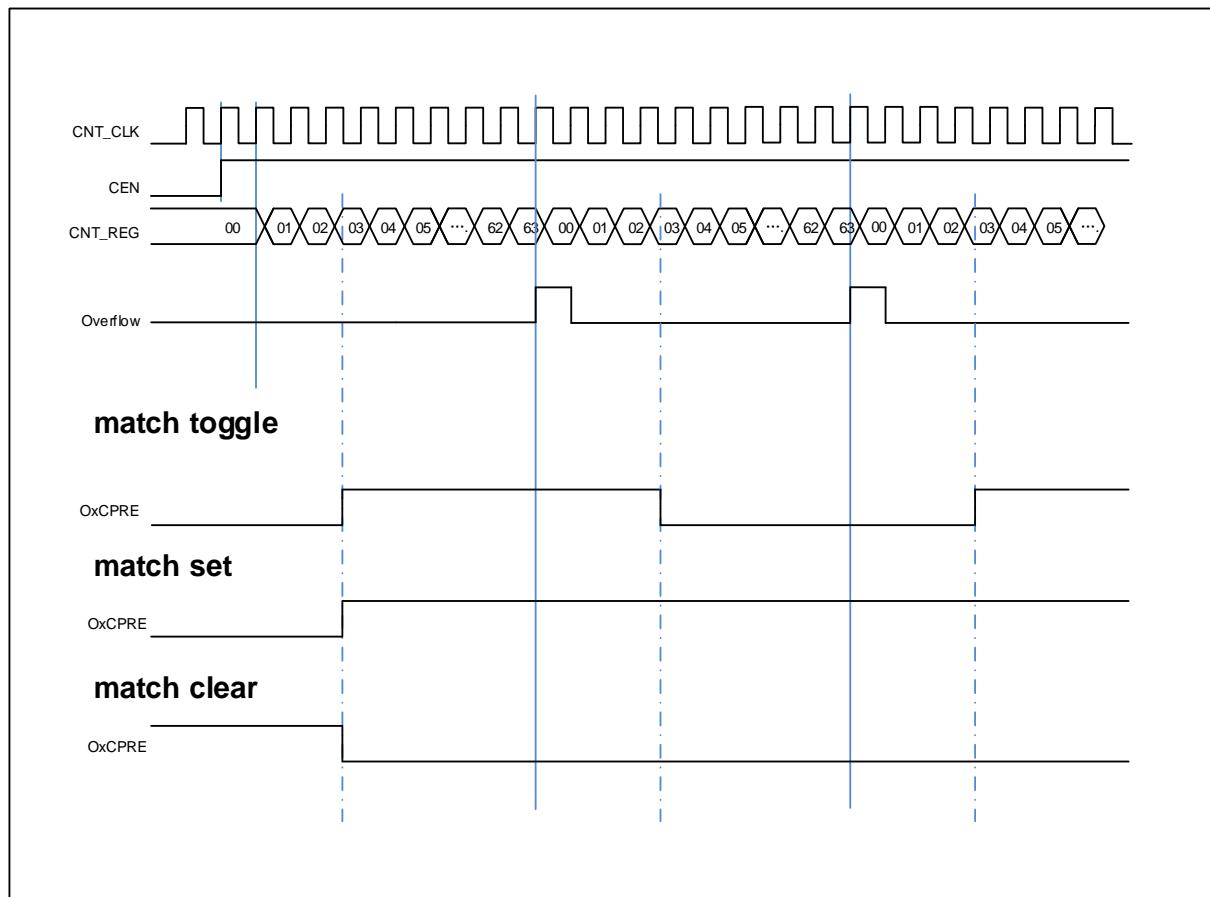
**Step4:** Compare output timing configuration by TIMERx\_CAR and TIMERx\_CHxCV.

About the CHxVAL, you can change it on the go to meet the waveform you expected.

**Step5:** Start the counter by CEN.

The timechart below show the three compare modes toggle/set/clear. CAR=0x63, CHxVAL=0x3

**Figure 16-50. Output-compare under three modes**



### Output PWM function

In the output PWM function (by setting the CHxCOMCTL bits to 3'b110 (PWM mode0) or to 3'b 111(PWM mode1), the channel can outputs PWM waveform according to the TIMERx\_CAR registers and TIMERx\_CHxCV registers.

Based on the counter mode, we have can also divide PWM into EAPWM (Edge aligned PWM) and CAPWM (Centre aligned PWM).

The EAPWM period is determined by TIMERx\_CAR and duty cycle is by TIMERx\_CHxCV. [\*\*Figure 16-51. EAPWM timechart\*\*](#) shows the EAPWM output and interrupts waveform.

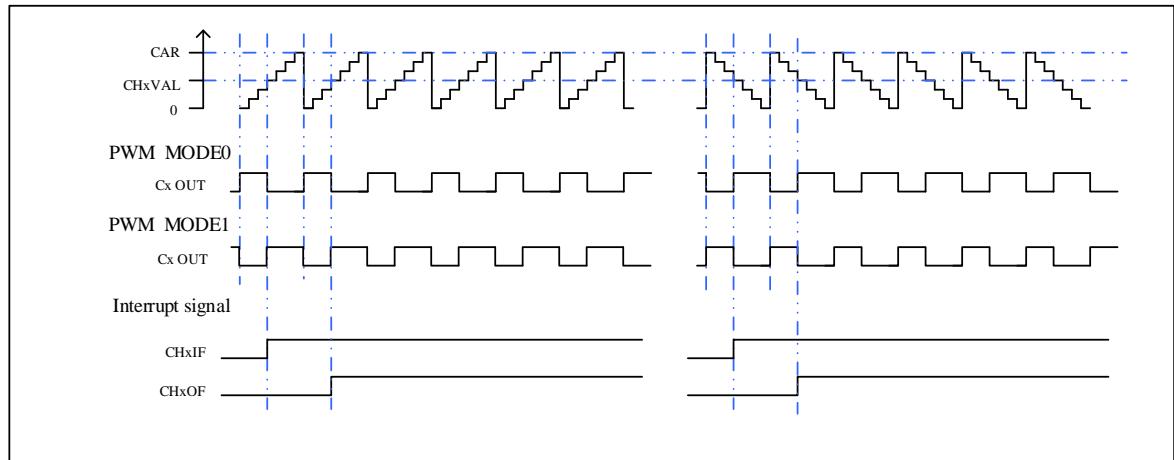
The CAPWM period is determined by 2\*TIMERx\_CAR, and duty cycle is determined by 2\*TIMERx\_CHxCV. [\*\*Figure 16-52. CAPWM timechart\*\*](#) shows the CAPWM output and interrupt waveform.

If TIMERx\_CHxCV is greater than TIMERx\_CAR, the output will be always active under PWM

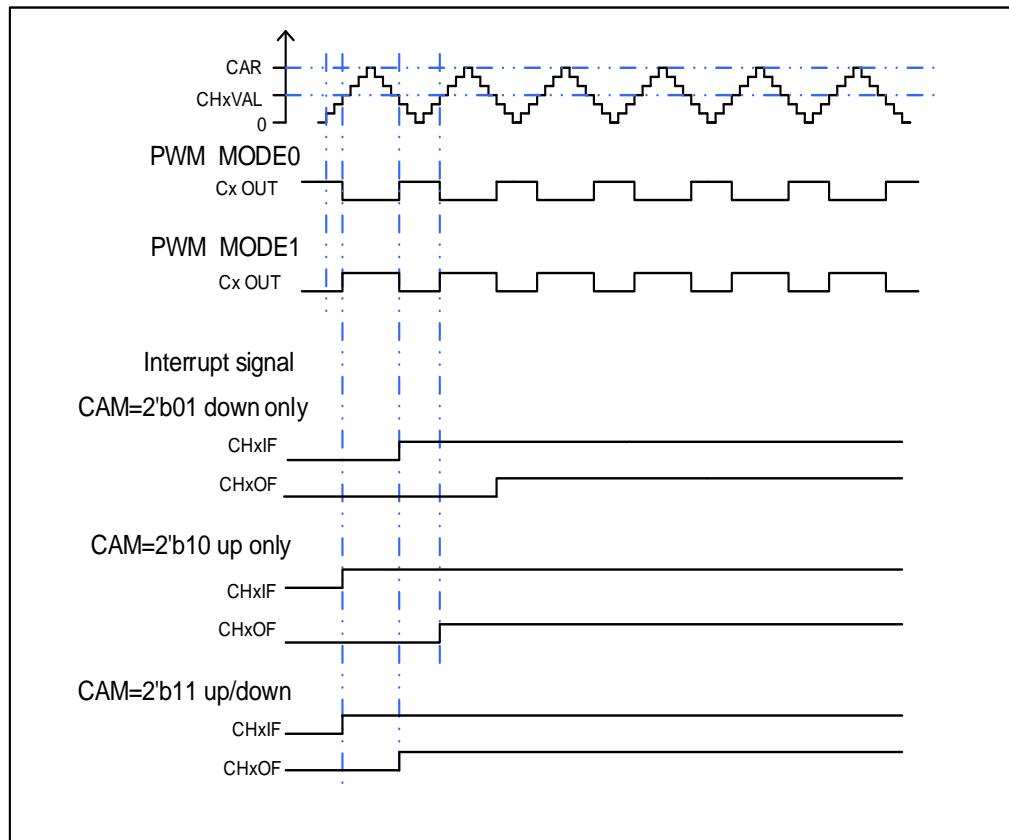
mode0 ( $\text{CHxCOMCTL} == 3'b110$ ).

And if  $\text{TIMERx_CHxCV}$  is equal to zero, the output will be always inactive under PWM mode0 ( $\text{CHxCOMCTL} == 3'b110$ ).

**Figure 16-51. EAPWM timechart**



**Figure 16-52. CAPWM timechart**



### Channel output prepare signal

When the  $\text{TIMERx}$  is used in the compare match output mode, the  $\text{OxCPRE}$  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.

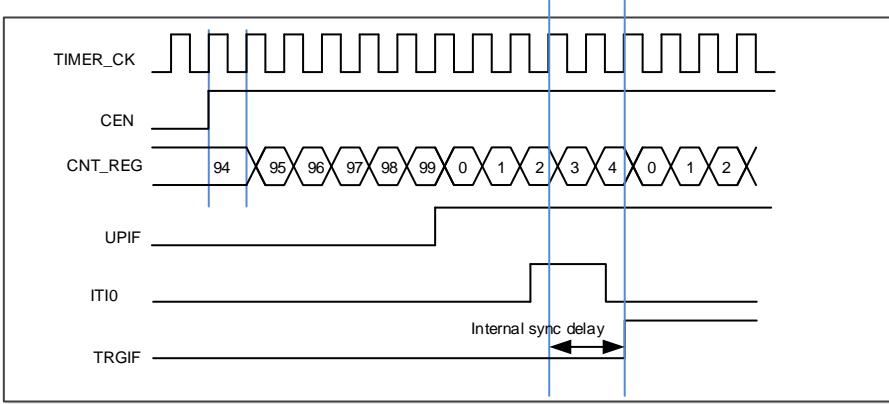
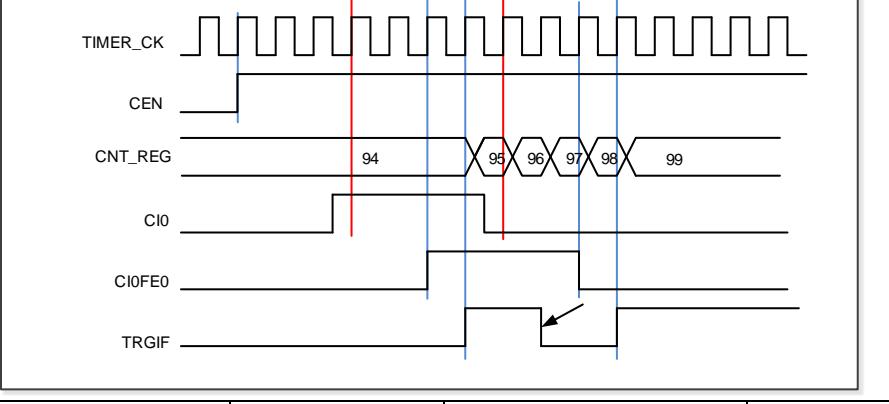
Another special function of the OxCPRE signal is a forced output which can be achieved by setting the CHxCOMCTL field to 0x04/0x05. Here the output can be forced to an inactive/active level irrespective of the comparison condition between the counter and the TIMERx\_CHxCV values.

### **Master-slave management**

The TIMERx can be synchronized with a trigger in several modes including the Restart mode, the Pause mode and the Event mode which is selected by the SMC [2:0] in the TIMERx\_SMCFG register. The trigger input of these modes can be selected by the TRGS [2:0] in the TIMERx\_SMCFG register.

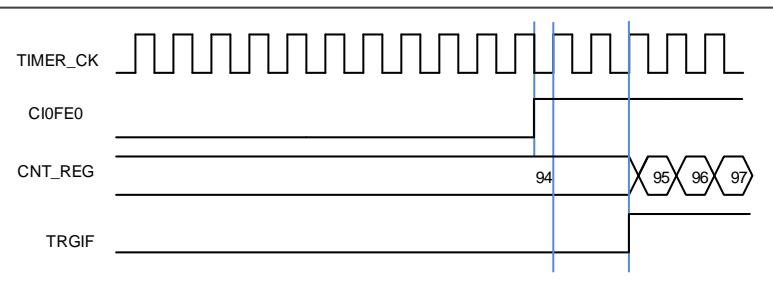
**Table 16-6. Examples of slave mode**

	<b>Mode Selection</b>	<b>Source Selection</b>	<b>Polarity Selection</b>	<b>Filter and Prescaler</b>
LIST	SMC[2:0] 3'b100 (restart mode) 3'b101 (pause mode) 3'b110 (event mode)	TRGS[2:0] 000: ITI0 001: ITI1 010: ITI2 011: ITI3 100: CI0F_ED 101: CI0FE0 110: CI1FE1 111: Reserved	If you choose the CI0FE0 or CI1FE1, configure the CHxP and CHxNP for the polarity selection and inversion.	For the ITIx no filter and prescaler can be used.  For the CIx, configure Filter by CHxCAPFLT, no prescaler can be used.
Exam1	Restart mode  The counter can be clear and restart when a rising trigger input.	TRGS[2:0]=3'b000  ITI0 is the selection.	-  For ITI0, no polarity selector can be used.	-  For the ITI0, no filter and prescaler can be used.

	<b>Mode Selection</b>	<b>Source Selection</b>	<b>Polarity Selection</b>	<b>Filter and Prescaler</b>
	<b>Figure 16-53. Restart mode</b>			
		 <p>The diagram illustrates the timing sequence for a counter restart. The <b>CEN</b> signal (Counter Enable) goes high at the start of the sequence. The <b>CNT_REG</b> signal shows the count from 94 to 99, then back to 0, 1, 2, 3, 4, 0, 1, 2. The <b>UPIF</b> signal (Update Flag) is asserted during the count. The <b>ITIO</b> signal (Interrupt Output) is asserted after the count reaches 4. The <b>TRGIF</b> signal (Trigger Flag) is asserted after the count reaches 1. A double-headed arrow labeled "Internal sync delay" indicates the time between the rising edge of CEN and the assertion of TRGIF.</p>		
Exam2	Pause mode The counter can be paused when the trigger input is low.	TRGS[2:0]=3'b101 CI0FE0 is the selection.	CH0P==0, no inverted. Capture will be sensitive to the rising edge only.	Filter is bypass in this example.
	<b>Figure 16-54. Pause mode</b>			
		 <p>The diagram illustrates the timing sequence for pause mode. The <b>CEN</b> signal (Counter Enable) goes high at the start of the sequence. The <b>CNT_REG</b> signal shows the count from 94 to 99. The <b>CI0</b> signal (Capture Input 0) is asserted during the count. The <b>CI0FE0</b> signal (Capture Selection 0) is asserted during the count. The <b>TRGIF</b> signal (Trigger Flag) is asserted after the count reaches 99. Red vertical lines indicate the points where the counter is captured by CI0 and CI0FE0.</p>		
Exam3	Event mode The counter will start to count when a rising trigger input.	TRGS[2:0]=3'b101 CI0FE0 is the selection.	CH0P==0, no inverted.	Filter is bypass in this example.

	Mode Selection	Source Selection	Polarity Selection	Filter and Prescaler
	<b>Figure 16-55. Event mode</b>			

**Figure 16-55. Event mode**



### Single pulse mode

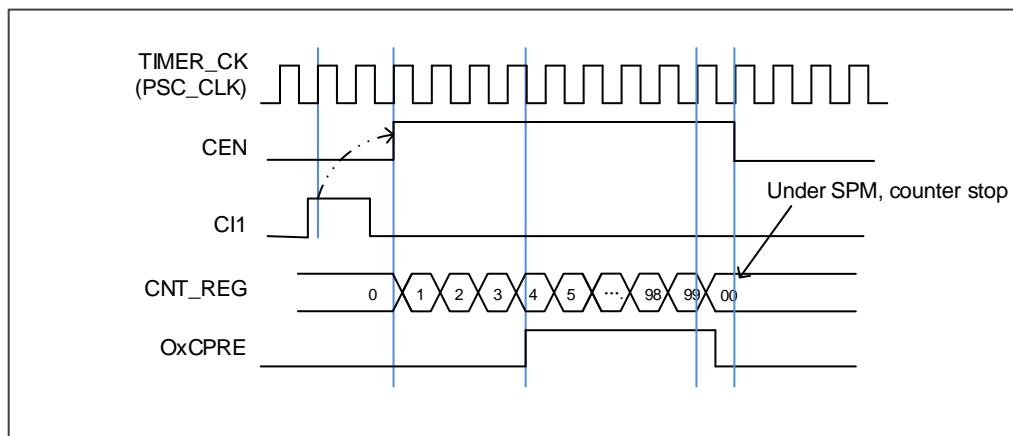
Single pulse mode is opposite to the repetitive mode, which can be enabled by setting SPM in **TIMERx\_CTL0**. When you set SPM, the counter will be clear and stop when the next update event. In order to get pulse waveform, you can set the **TIMERx** to PWM mode or compare by **CHxCOMCTL**.

Once the timer is set to operate in the single pulse mode, it is not necessary to set the timer enable bit CEN in the **TIMERx\_CTL0** register to 1 to enable the counter. The trigger to generate a pulse can be sourced from the trigger signals edge or by setting the CEN bit to 1 using software. Setting the CEN bit to 1 or a trigger from the trigger signals edge can generate a pulse and then keep the CEN bit at a high state until the update event occurs or the CEN bit is written to 0 by software. If the CEN bit is cleared to 0 using software, the counter will be stopped and its value held.

In the single pulse mode, the trigger active edge which sets the CEN bit to 1 will enable the counter. However, there exist several clock delays to perform the comparison result between the counter value and the **TIMERx\_CHxCV** value. In order to reduce the delay to a minimum value, the user can set the **CHxCOMFEN** bit in each **TIMERx\_CHCTL0/1** register. After a trigger rising occurs in the single pulse mode, the **OxCPRE** signal will immediately be forced to the state which the **OxCPRE** signal will change to, as the compare match event occurs without taking the comparison result into account. The **CHxCOMFEN** bit is available only when the output channel is configured to operate in the PWM0 or PWM1 output mode and the trigger source is derived from the trigger signal.

[\*\*Figure 16-56. Single pulse mode \*\*TIMERx\\_CHxCV = 4\*\* \*\*TIMERx\\_CAR=99\*\*\*\*](#) shows an example.

Figure 16-56. Single pulse mode TIMERx\_CHxCV = 4 TIMERx\_CAR=99



### Timers interconnection

Refer to [Advanced timer \(TIMERx, x=0, 7\)](#).

### Timer debug mode

When the Cortex®-M4 halted, and the TIMERx\_HOLD configuration bit in DBG\_CTL0 register set to 1, the TIMERx counter stops.

### 16.3.5. **TIMERx registers(x=8, 11)**

TIMER8 base address: 0x4001 4C00

TIMER11 base address: 0x4000 1800

#### **Control register 0 (TIMERx\_CTL0)**

Address offset: 0x00

Reset value: 0x0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				CKDIV[1:0]		ARSE		Reserved			SPM	UPS	UPDIS	CEN	r

rw                    rw                    rw                    rw                    rw                    r

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 factor between the CK_TIMER and the dead-time and digital filter sample clock (DTS). 00: $f_{DTS} = f_{CK\_TIMER}$ 01: $f_{DTS} = f_{CK\_TIMER} / 2$ 10: $f_{DTS} = f_{CK\_TIMER} / 4$ 11: Reserved
7	ARSE	Auto-reload shadow enable 0: The shadow register for TIMERx_CAR register is disabled 1: The shadow register for TIMERx_CAR register is enabled
6:4	Reserved	Must be kept at reset value.
3	SPM	Single pulse mode. 0: Single pulse mode disable. The counter continues after update event. 1: Single pulse mode enable. The counter counts until the next update event occurs.
2	UPS	Update source This bit is used to select the update event sources by software. 0: These events generate update interrupts or DMA requests: The UPG bit is set The counter generates an overflow or underflow event The restart mode generates an update event. 1: This event generates update interrupts or DMA requests: The counter generates an overflow or underflow event

---

1	UPDIS	<p>Update disable.</p> <p>This bit is used to enable or disable the update event generation.</p> <p>0: Update event enable. When an update event occurs, the corresponding shadow registers are loaded with their preloaded values. These events generate update event:</p> <ul style="list-style-type: none"> <li>The UPG bit is set</li> <li>The counter generates an overflow or underflow event</li> <li>The restart mode generates an update event.</li> </ul> <p>1: Update event disable.</p> <p><b>Note:</b> When this bit is set to 1, setting UPG bit or the restart mode does not generate an update event, but the counter and prescaler are initialized.</p>
0	CEN	<p>Counter enable</p> <p>0: Counter disable</p> <p>1: Counter enable</p> <p>The CEN bit must be set by software when timer works in external clock, pause mode and encoder mode.</p>

### 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
Reserved			MSM			TRGS[2:0]			Reserved			SMC[2:0]			
rw			rw			rw			rw			rw			

Bits	Fields	Descriptions
15:8	Reserved	Must be kept at reset value.
7	MSM	<p>Master-slave mode</p> <p>This bit can be used to synchronize selected timers to begin counting at the same time. The TRGI is used as the start event, and through TRGO, timers are connected together.</p> <p>0: Master-slave mode disable</p> <p>1: Master-slave mode enable</p>
6:4	TRGS[2:0]	<p>Trigger selection</p> <p>This bit-field specifies which signal is selected as the trigger input, which is used to synchronize the counter.</p> <p>000: ITI0</p> <p>001: ITI1</p> <p>010: ITI2</p> <p>011: ITI3</p>

		100: CI0F_ED
		101: CI0FE0
		110: CI1FE1
		111: Reserved.
		These bits must not be changed when slave mode is enabled.
3	Reserved	Must be kept at reset value.
2:0	SMC[2:0]	<p>Slave mode control</p> <p>000: Disable mode. The slave mode is disabled; The prescaler is clocked directly by the internal clock (TIMER_CK) when CEN bit is set high.</p> <p>001: Reserved.</p> <p>010: Reserved.</p> <p>011: Reserved.</p> <p>100: Restart mode. The counter is reinitialized and an update event is generated on the rising edge of the selected trigger input.</p> <p>101: Pause mode. The trigger input enables the counter clock when it is high and disables the counter clock when it is low.</p> <p>110: Event mode. A rising edge of the trigger input enables the counter.</p> <p>111: External clock mode0. The counter counts on the rising edges of the selected trigger.</p>

## 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		TRGIE		Reserved		CH1IE	CH0IE	UPIE

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
15:7	Reserved	Must be kept at reset value.
6	TRGIE	Trigger interrupt enable 0: disabled 1: enabled
5:3	Reserved	Must be kept at reset value.
2	CH1IE	Channel 1 capture/compare interrupt enable 0: disabled 1: enabled
1	CH0IE	Channel 0 capture/compare interrupt enable

0: disabled

1: enabled

0	UPIE	Update interrupt enable
		0: disabled
		1: enabled

### Interrupt flag register (TIMERx\_INTF)

Address offset: 0x10

Reset value: 0x0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		CH1OF	CH0OF	Reserved	TRGIF	Reserved			rc_w0	rc_w0	rc_w0	rc_w0	CH1IF	CH0IF	UPIF

Bits	Fields	Descriptions
15:11	Reserved	Must be kept at reset value.
10	CH1OF	Channel 1 over capture flag Refer to CH0OF description
9	CH0OF	Channel 0 over capture flag When channel 0 is configured in input mode, this flag is set by hardware when a capture event occurs while CH0IF flag has already been set. This flag is cleared by software. 0: No over capture interrupt occurred 1: Over capture interrupt occurred
8:7	Reserved	Must be kept at reset value.
6	TRGIF	Trigger interrupt flag This flag is set on trigger event and cleared by software. When in pause mode, both edges on trigger input generates a trigger event, otherwise, only an active edge on trigger input can generates a trigger event. 0: No trigger event occurred. 1: Trigger interrupt occurred.
5:3	Reserved	Must be kept at reset value.
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.

If Channel0 is set to input mode, this bit will be reset by reading TIMERx\_CH0CV.

0: No Channel 0 interrupt occurred

1: Channel 0 interrupt occurred

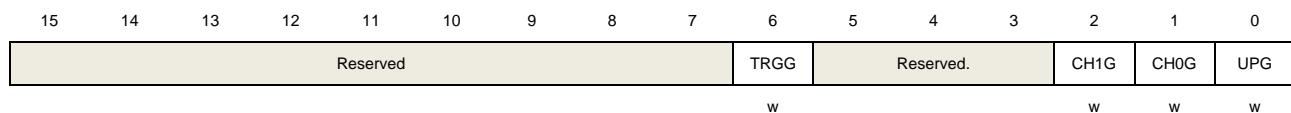
0	UPIF	Update interrupt flag This bit is set by hardware on an update event and cleared by software. 0: No update interrupt occurred 1: Update interrupt occurred
---	------	---

## Software event generation register (TIMERx\_SWEVG)

Address offset: 0x14

Reset value: 0x0000

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



Bits	Fields	Descriptions
15:7	Reserved	Must be kept at reset value.
6	TRGG	Trigger event generation This bit is set by software and cleared by hardware automatically. When this bit is set, the TRGIF flag in TIMERx_STAT register is set, related interrupt or DMA transfer can occur if enabled. 0: No generate a trigger event 1: Generate a trigger event
5:3	Reserved	Must be kept at reset value.
2	CH1G	Channel 1's capture or compare event generation Refer to CH0G description
1	CH0G	Channel 0's capture or compare event generation This bit is set by software in order to generate a capture or compare event in channel 0, it is automatically cleared by hardware. When this bit is set, the 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. 0: No generate a channel 1 capture or compare event 1: Generate a channel 1 capture or compare event
0	UPG	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.

- 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
Reserved	CH1COMCTL[2:0]	CH1COM SEN	CH1COM FEN	CH1MS[1:0]	Reserved	CH0COMCTL[2:0]	CH0COM SEN	CH0COM FEN	CH0MS[1:0]	CH0CAPFLT[3:0]	CH0CAPPSC[1:0]	CH0CAPPSC[1:0]	CH0MS[1:0]	CH0MS[1:0]	CH0MS[1:0]
CH1CAPFLT[3:0]	CH1CAPPSC[1:0]					CH0CAPFLT[3:0]									

rw

rw

rw

rw

rw

rw

#### **Output compare mode:**

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
15	Reserved	Must be kept at reset value.
14:12	CH1COMCTL[2:0]	Channel 1 compare output control Refer to CH0COMCTL description
11	CH1COMSEN	Channel 1 output compare shadow enable Refer to CH0COMSEN description
10	CH1COMFEN	Channel 1 output compare fast enable Refer to CH0COMFEN 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 programmed as output mode 01: Channel 1 is programmed as input mode, IS1 is connected to CI1FE1 10: Channel 1 is programmed as input mode, IS1 is connected to CI0FE1 11: Channel 1 is programmed as input mode, IS1 is connected to ITS. <b>Note:</b> When CH1MS[1:0]=11, it is necessary to select an internal trigger input through TRGS bits in TIMERx_SMCFG register.
7	Reserved	Must be kept at reset value.
6:4	CH0COMCTL[2:0]	Channel 0 compare output control This bit-field specifies the compare output mode of the the output prepare signal O0CPRE. In addition, the high level of O0CPRE is the active level, and CH0_O and CH0_ON channels polarity depends on CH0P and CH0NP 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 is equals to the output compare register TIMERx\_CH0CV.

010: Clear the channel output. O0CPRE signal is forced low when the counter is equals to the output compare register TIMERx\_CH0CV.

011: Toggle on match. O0CPRE toggles when the counter is equals to the output compare register TIMERx\_CH0CV.

100: Force low. O0CPRE is forced to low level.

101: Force high. O0CPRE is forced to high level.

110: PWM mode0. When counting up, O0CPRE is high when the counter is smaller than TIMERx\_CH0CV, and low otherwise. When counting down, O0CPRE is low when the counter is larger than TIMERx\_CH0CV, and high otherwise.

111: PWM mode1. When counting up, O0CPRE is low when the counter is smaller than TIMERx\_CH0CV, and high otherwise. When counting down, O0CPRE is high when the counter is larger than TIMERx\_CH0CV, and low otherwise.

If configured in PWM mode, the O0CPRE level changes only when the output compare mode is adjusted from “Timing” mode to “PWM” mode or the comparison result changes.

This bit cannot be modified when PROT [1:0] bit-filed in TIMERx\_CCHP register is 11 and CH0MS bit-filed is 00(COMPARE MODE).

3	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 verifying the shadow register only in single pulse mode (when SPM=1)</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 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. 1: Channel 0 output quickly compare enable.</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>

00: Channel 0 is programmed as output mode

01: Channel 0 is programmed as input mode, IS0 is connected to CI0FE0

10: Channel 0 is programmed as input mode, IS0 is connected to CI1FE0

11: Channel 0 is programmed as input mode, IS0 is connected to ITS

**Note:** When CH0MS[1:0]=11, it is necessary to select an internal trigger input 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 The CI0 input signal can be filtered by digital filter and this bit-field configure the filtering capability. Basic principle of digital filter: continuously sample the CI0 input signal according to $f_{SAMP}$ and record the number of times of the same level of the signal. After reaching the filtering capacity configured by this bit, it is considered to be an effective level. The filtering capability configuration is as follows:																																											
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr style="background-color: #ADD8E6;"> <th>CH0CAPFLT [3:0]</th> <th>Times</th> <th><math>f_{SAMP}</math></th> </tr> </thead> <tbody> <tr> <td>4'b0000</td> <td colspan="2" style="text-align: center;">Filter disabled.</td></tr> <tr> <td>4'b0001</td> <td style="text-align: center;">2</td> <td rowspan="3" style="vertical-align: middle; text-align: center;"><math>f_{CK\_TIMER}</math></td></tr> <tr> <td>4'b0010</td> <td style="text-align: center;">4</td></tr> <tr> <td>4'b0011</td> <td style="text-align: center;">8</td></tr> <tr> <td>4'b0100</td> <td style="text-align: center;">6</td> <td rowspan="2" style="vertical-align: middle; text-align: center;"><math>f_{DTS}/2</math></td></tr> <tr> <td>4'b0101</td> <td style="text-align: center;">8</td></tr> <tr> <td>4'b0110</td> <td style="text-align: center;">6</td> <td rowspan="2" style="vertical-align: middle; text-align: center;"><math>f_{DTS}/4</math></td></tr> <tr> <td>4'b0111</td> <td style="text-align: center;">8</td></tr> <tr> <td>4'b1000</td> <td style="text-align: center;">6</td> <td rowspan="2" style="vertical-align: middle; text-align: center;"><math>f_{DTS}/8</math></td></tr> <tr> <td>4'b1001</td> <td style="text-align: center;">8</td></tr> <tr> <td>4'b1010</td> <td style="text-align: center;">5</td> <td rowspan="3" style="vertical-align: middle; text-align: center;"><math>f_{DTS}/16</math></td></tr> <tr> <td>4'b1011</td> <td style="text-align: center;">6</td></tr> <tr> <td>4'b1100</td> <td style="text-align: center;">8</td></tr> <tr> <td>4'b1101</td> <td style="text-align: center;">5</td> <td rowspan="2" style="vertical-align: middle; text-align: center;"><math>f_{DTS}/32</math></td></tr> <tr> <td>4'b1110</td> <td style="text-align: center;">6</td></tr> <tr> <td>4'b1111</td> <td style="text-align: center;">8</td><td></td></tr> </tbody> </table>		CH0CAPFLT [3:0]	Times	$f_{SAMP}$	4'b0000	Filter disabled.		4'b0001	2	$f_{CK\_TIMER}$	4'b0010	4	4'b0011	8	4'b0100	6	$f_{DTS}/2$	4'b0101	8	4'b0110	6	$f_{DTS}/4$	4'b0111	8	4'b1000	6	$f_{DTS}/8$	4'b1001	8	4'b1010	5	$f_{DTS}/16$	4'b1011	6	4'b1100	8	4'b1101	5	$f_{DTS}/32$	4'b1110	6	4'b1111	8	
CH0CAPFLT [3:0]	Times	$f_{SAMP}$																																											
4'b0000	Filter disabled.																																												
4'b0001	2	$f_{CK\_TIMER}$																																											
4'b0010	4																																												
4'b0011	8																																												
4'b0100	6	$f_{DTS}/2$																																											
4'b0101	8																																												
4'b0110	6	$f_{DTS}/4$																																											
4'b0111	8																																												
4'b1000	6	$f_{DTS}/8$																																											
4'b1001	8																																												
4'b1010	5	$f_{DTS}/16$																																											
4'b1011	6																																												
4'b1100	8																																												
4'b1101	5	$f_{DTS}/32$																																											
4'b1110	6																																												
4'b1111	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, input capture occurs on every channel input edge
- 01: The input capture occurs on every 2 channel input edges
- 10: The input capture occurs on every 4 channel input edges
- 11: The input capture occurs on every 8 channel input edges

1:0	CH0MS[1:0]	Channel 0 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			CH1NP	Reserved	CH1P	CH1EN	CH0NP	Reserved	CH0P	CH0EN

rw                    rw                    rw                    rw                    rw                    rw

Bits	Fields	Descriptions
15:8	Reserved	Must be kept at reset value.
7	CH1NP	Channel 1 complementary output polarity Refer to CH0NP description
6	Reserved	Must be kept at reset value.
5	CH1P	Channel 1 capture/compare function polarity Refer to CH0P description
4	CH1EN	Channel 1 capture/compare function enable Refer to CH1EN description
3	CH0NP	Channel 0 complementary output polarity When channel 0 is configured in output mode, this bit should be keep reset value. When channel 0 is configured in input mode, together with CH0P, this bit is used to define the polarity of CI0. This bit cannot be modified when PROT [1:0] bit-field in TIMERx_CCHP register is 11 or 10.
2	Reserved	Must be kept at reset value.
1	CH0P	Channel 0 capture/compare function polarity When channel 0 is configured in output mode, this bit specifies the output signal polarity. 0: Channel 0 high level is active level 1: Channel 0 low level is active level

When channel 0 is configured in input mode, this bit specifies the CI0 signal polarity.

[CH0NP, CH0P] will select the active trigger or capture polarity for CI0FE0 or CI1FE0.

[CH0NP==0, CH0P==0]: ClxFE0's rising edge is the active signal for capture or trigger operation in slave mode. And ClxFE0 will not be inverted.

[CH0NP==0, CH0P==1]: ClxFE0's falling edge is the active signal for capture or trigger operation in slave mode. And ClxFE0 will be inverted.

[CH0NP==1, CH0P==0]: Reserved.

[CH0NP==1, CH0P==1]: Reserved.

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-field indicates the current counter value. Writing to this bit-field can change the value of the counter.

## Prescaler register (TIMERx\_PSC)

Address offset: 0x28

Reset value: 0x0000

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

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 TIMER\_CK 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>This bit-filed specifies the auto reload value of the counter.</p> <p><b>Note:</b> 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-filed indicates the counter value corresponding to the last capture event. And this bit-filed is read-only.</p> <p>When channel 0 is configured in output mode, this bit-filed contains value to be compared to the counter. When the corresponding shadow register is enabled, the shadow register updates every update event.</p>

### Channel 1 capture/compare value register (TIMERx\_CH1CV)

Address offset: 0x38

Reset value: 0x0000

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

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>

### Configuration register (TIMERx\_CFG )

Address offset: 0xFC

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														CHVSEL	Reserved

rw

Bits	Fields	Descriptions
15:2	Reserved	Must be kept at reset value.
1	CHVSEL	<p>Write CHxVAL register selection</p> <p>This bit-field set and reset by software.</p> <p>1: If write the CHxVAL register, the write value is same as the CHxVAL value, the write access ignored</p> <p>0: No effect</p>
0	Reserved	Must be kept at reset value.

## 16.4. General level2 timer (TIMERx, x=9, 10, 12, 13)

### 16.4.1. Overview

The general level2 timer module (Timer9, 10, 12, 13) is a one-channel timer that supports input capture, output compare. They can generate PWM signals to control motor or be used for power management applications. The general level2 time reference is a 16-bit counter that can be used as an unsigned counter.

In addition, the general level2 timers can be programmed and be used to count or time external events that drive other Timers.

### 16.4.2. Characteristics

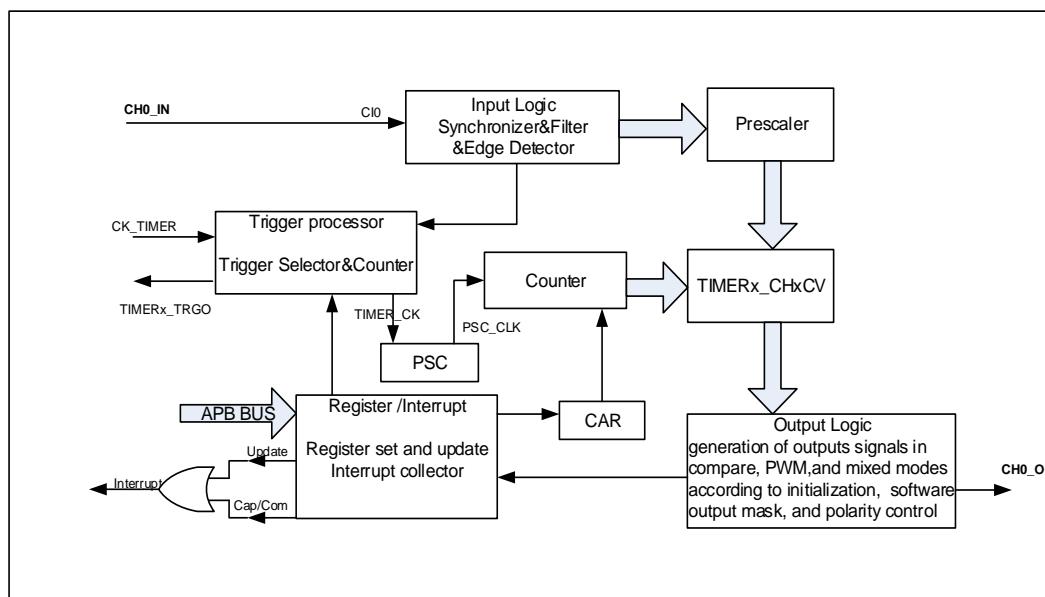
- Total channel num: 1.
- Counter width: 16 bits.
- Source of count clock: internal clock.
- Counter mode: count up only.
- Programmable prescaler: 16 bit. Factor can be changed on the go.
- Each channel is user-configurable:  
Input capture mode, output compare mode, programmable and PWM mode.
- Auto-reload function.
- Interrupt output on: update , compare/capture event.

### 16.4.3. Block diagram

[Figure 16-57. General level2 timer block diagram](#) provides details on the internal

configuration of the general level2 timer.

**Figure 16-57. General level2 timer block diagram**



#### 16.4.4. Function overview

##### Clock source configuration

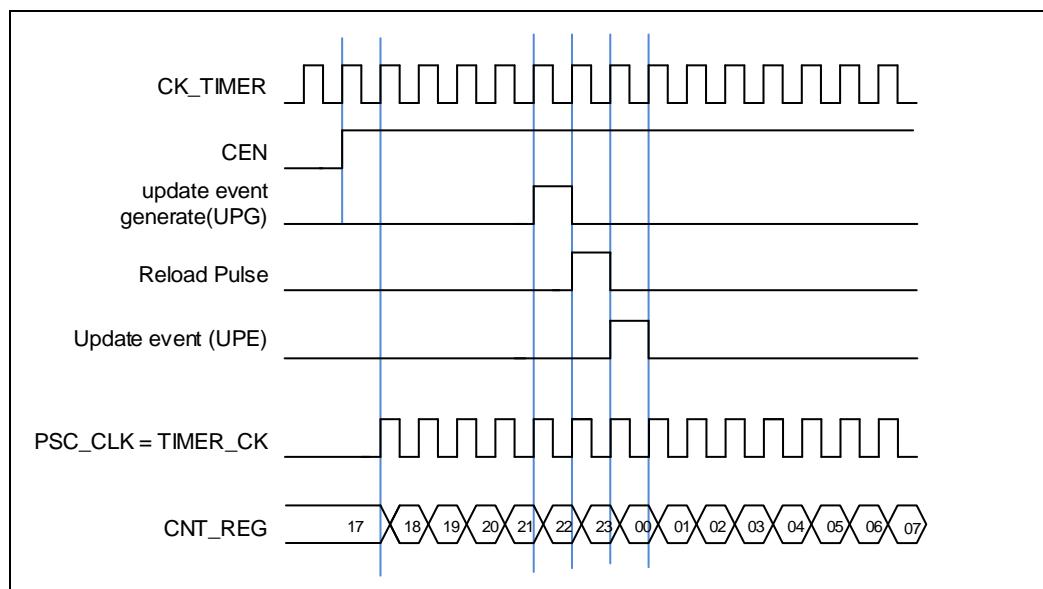
The general level2 TIMER can only being clocked by the CK\_TIMER.

- Internal timer clock CK\_TIMER which is from module RCU

The general level2 TIMER has only one clock source which is the internal CK\_TIMER, used to drive the counter prescaler. When the CEN is set, the CK\_TIMER will be divided by PSC value to generate PSC\_CLK.

The TIMER\_CK, driven counter's prescaler to count, is equal to CK\_TIMER which is from RCU

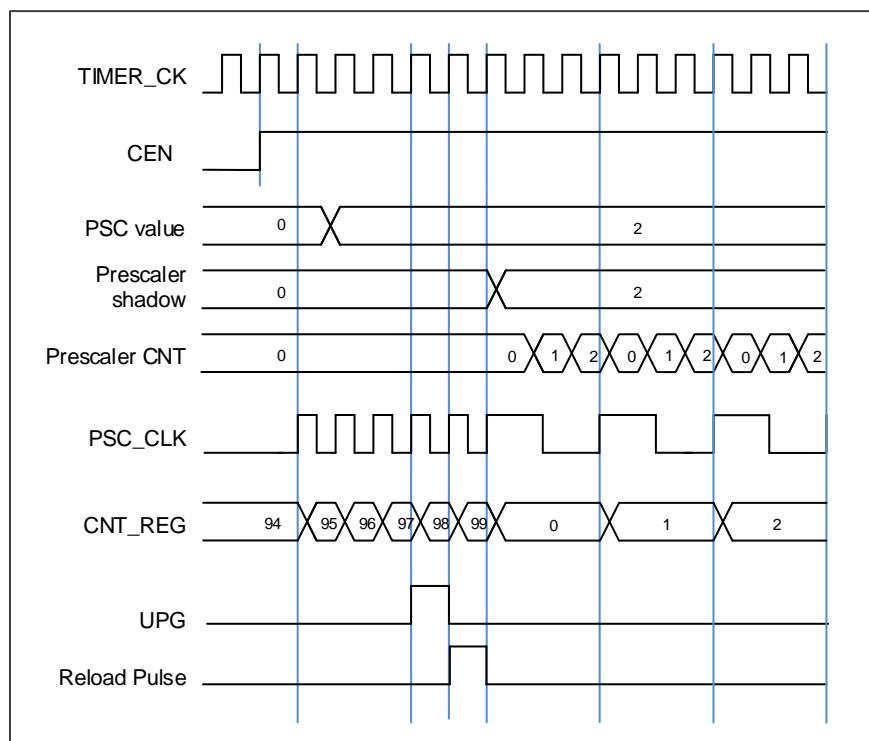
**Figure 16-58. Timing chart of internal clock divided by 1**



##### Clock prescaler

The counter clock (PSC\_CLK) is obtained by the TIMER\_CK through the prescaler, and the prescale factor can be configured from 1 to 65536 through the prescaler register (TIMERx\_PSC). The new written prescaler value will not take effect until the next update event.

Figure 16-59. Timing chart of PSC value change from 0 to 2



### Counter up counting

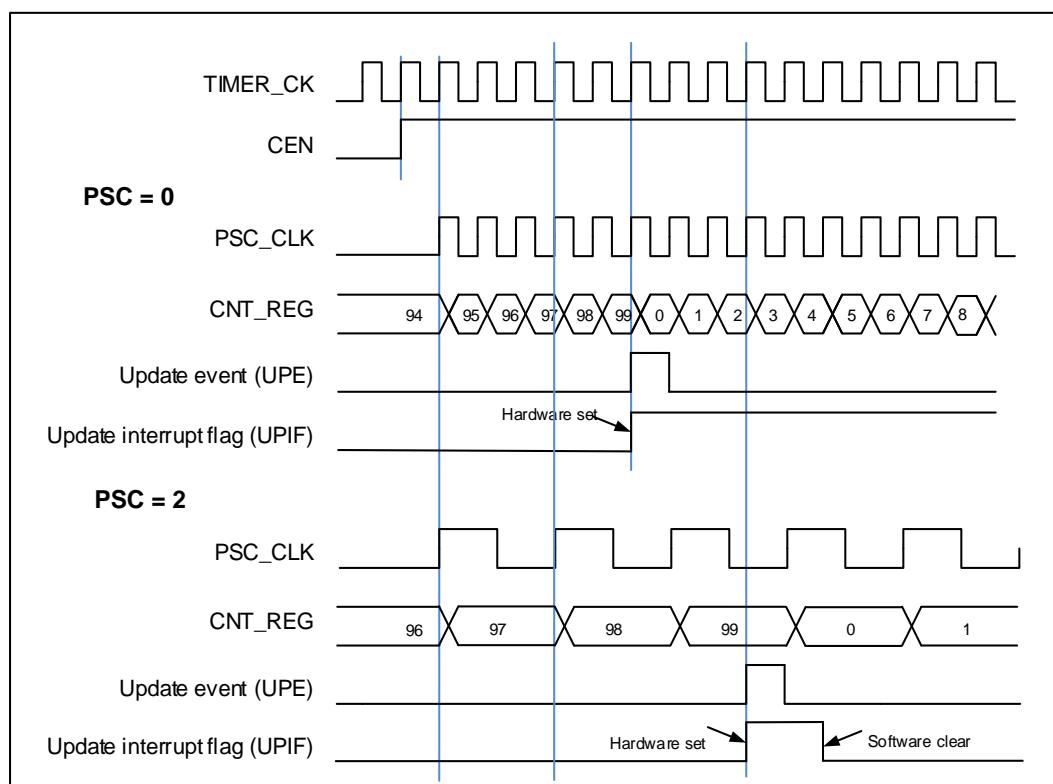
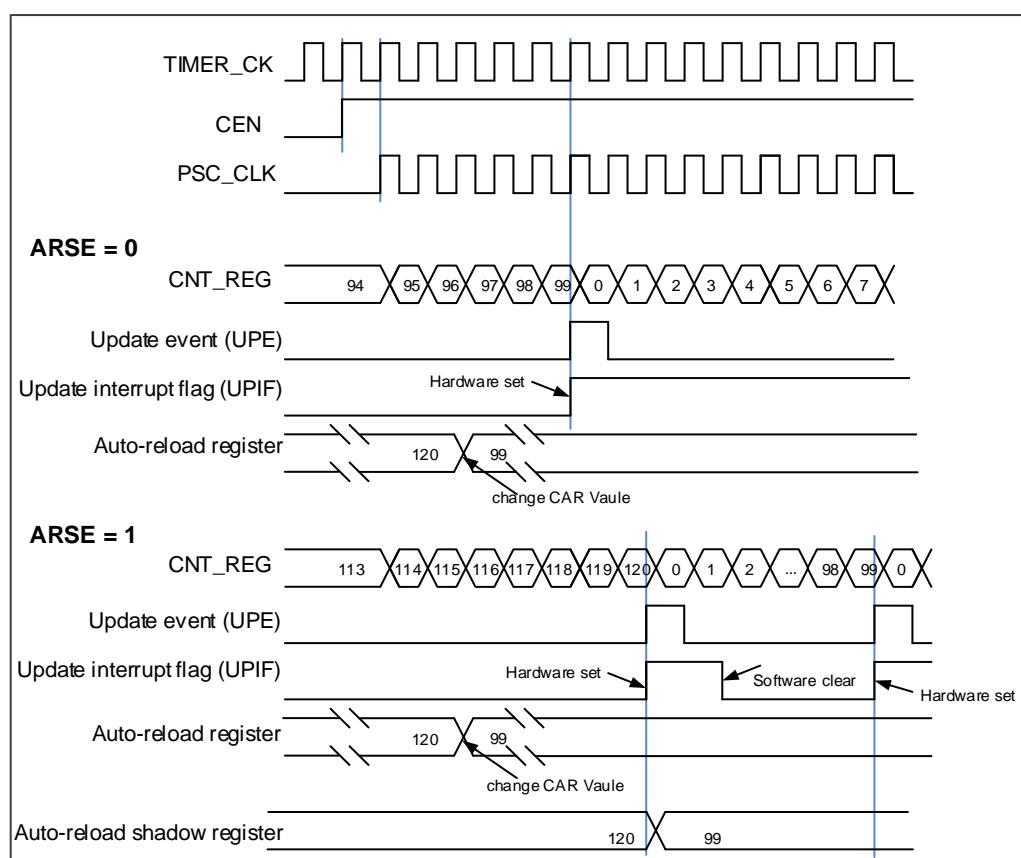
In this mode, the counter counts up continuously from 0 to the counter-reload value, which is defined in the **TIMERx\_CAR** register, in a count-up direction. Once the counter reaches the counter reload value, the counter will start counting up from 0 again. The update event is generated at each counter overflow. The counting direction bit **DIR** in the **TIMERx\_CTL1** register should be set to 0 for the up counting mode.

When the update event is set by the **UPG** bit in the **TIMERx\_SWEVG** register, the counter value will be initialized to 0 and generates an update event.

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

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

The following figures show some examples of the counter behavior for different clock prescaler factor when **TIMERx\_CAR=0x99**.

**Figure 16-60. Timing chart of up counting mode, PSC=0/2**

**Figure 16-61. Timing chart of up counting mode, change TIMERx\_CAR on the go**


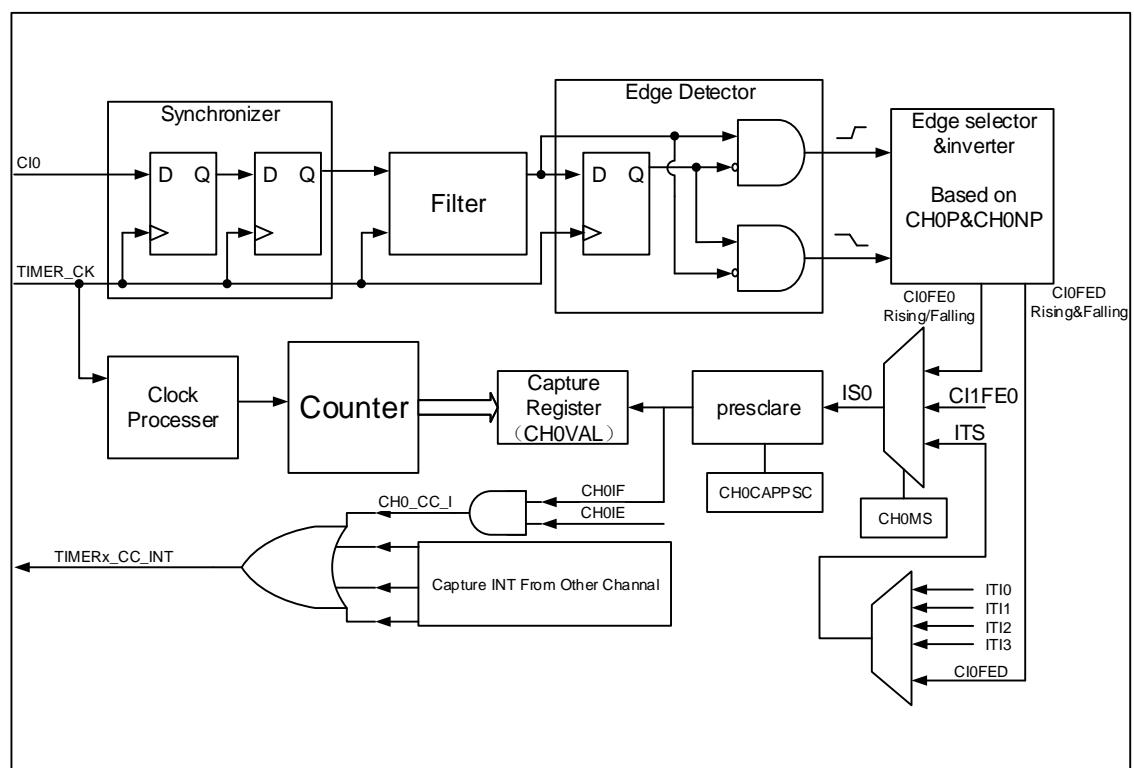
## Input capture and output compare channels

The general level2 timer has one independent channel which can be used as capture inputs or compare match outputs. Each channel is built around a channel capture compare register including an input stage, channel controller and an output stage.

### ■ Channel input capture function

Channel input capture function allows the channel to perform measurements such as pulse timing, frequency, period, duty cycle and so on. The input stage consists of a digital filter, a channel polarity selection, edge detection and a channel prescaler. When a selected edge occurs on the channel input, the current value of the counter is captured into the TIMERx\_CHxCV register, at the same time the CHxIF bit is set and the channel interrupt is generated if enabled by CHxIE = 1.

**Figure 16-62. Channel input capture principle**



First, the channel input signal (CIx) is synchronized to TIMER\_CK domain, and then sampled by a digital filter to generate a filtered input signal. Then through the edge detector, the rising and fall edge are detected. You can select one of them by CHxP. One more selector is for the other channel and trig, controlled by CHxMS. The IC\_prescaler make several the input event generate one effective capture event. On the capture event, TIMERx\_CHxCV will restore the value of counter.

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

**Step1:** Filter configuration. (CHxCAPFLT in TIMERx\_CHCTL0)

Based on the input signal and requested signal quality, configure compatible CHxCAPFLT.

**Step2:** Edge selection. (CHxP/CHxNP in TIMERx\_CHCTL2)

Rising or falling edge, choose one by CHxP/CHxNP.

**Step3:** Capture source selection. (CHxMS in TIMERx\_CHCTL0)

As soon as you select one input capture source by CHxMS, you have set the channel to input mode (CHxMS!=0x0) and TIMERx\_CHxCV cannot be written any more.

**Step4:** Interrupt enable. (CHxIE in TIMERx\_DMAINTEN)

Enable the related interrupt enable; you can get the interrupt.

**Step5:** Capture enables. (CHxEN in TIMERx\_CHCTL2)

**Result:** When you wanted input signal is got, TIMERx\_CHxCV will be set by Counter's value. And CHxIF is asserted. If the CHxIF is high, the CHxOF will be asserted also. The interrupt will be asserted based on your configuration of CHxIE in TIMERx\_DMAINTEN

**Direct generation:** If you want to generate a DMA request or Interrupt, you can set CHxG by software directly.

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

## ■ Channel output compare function

In channel Compare function, the TIMERx can generate timed pulses with programmable position, polarity, duration, and frequency. When the counter matches the value in the CHxVAL register of an output compare channel, the channel (n) output can be set, cleared, or toggled based on CHxCOMCTL. When the counter reaches the value in the CHxVAL register, the CHxIF bit is set and the channel (n) interrupt is generated if CHxIE = 1.

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

**Step1:** Clock configuration. Such as clock source, clock prescaler and so on.

**Step2:** Compare mode configuration.

- \* Set the shadow enable mode by CHxCOMSEN
- \* Set the output mode (Set/Clear/Toggle) by CHxCOMCTL.
- \* Select the active high polarity by CHxP/CHxNP
- \* Enable the output by CHxEN

**Step3:** Interrupt/DMA-request enables configuration by CHxIE

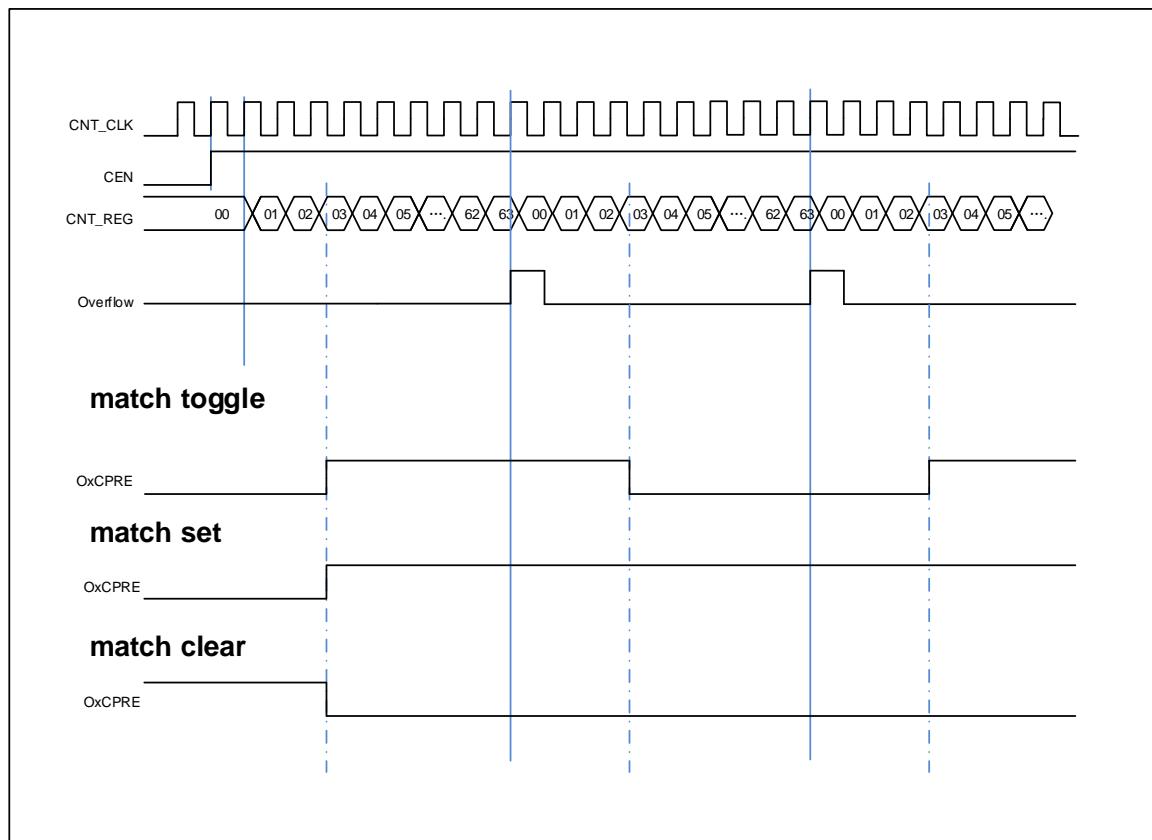
**Step4:** Compare output timing configuration by TIMERx\_CAR and TIMERx\_CHxCV.

About the CHxVAL, you can change it on the go to meet the waveform you expected.

**Step5:** Start the counter by CEN.

The timechart below show the three compare modes toggle/set/clear. CAR=0x63, CHxVAL=0x3

**Figure 16-63. Output-compare under three modes**



### 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.

## Timers interconnection

Refer to [Advanced timer \(TIMERx, x=0, 7\).](#)

### Timer debug mode

When the Cortex®-M4 halted, and the TIMERx\_HOLD configuration bit in DBG\_CTL0 register set to 1, the TIMERx counter stops.

### 16.4.5. TIMERx registers(x=9, 10, 12, 13)

TIMER9 base address: 0x4001 5000

TIMER10 base address: 0x4001 5400

TIMER12 base address: 0x4000 1C00

TIMER13 base address: 0x4000 2000

#### **Control register 0 (TIMERx\_CTL0)**

Address offset: 0x00

Reset value: 0x0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved				CKDIV[1:0]		ARSE		Reserved		UPS		UPDIS		CEN		
						rw		rw				rw		rw		rw

Bits	Fields	Descriptions
15:10	Reserved	Must be kept at reset value.
9:8	CKDIV[1:0]	Clock division The CKDIV bits can be configured by software to specify division factor between the CK_TIMER and the dead-time and digital filter sample clock (DTS). 00: $f_{DTS}=f_{CK\_TIMER}$ 01: $f_{DTS}= f_{CK\_TIMER} /2$ 10: $f_{DTS}= f_{CK\_TIMER} /4$ 11: Reserved
7	ARSE	Auto-reload shadow enable 0: The shadow register for TIMERx_CAR register is disabled 1: The shadow register for TIMERx_CAR register is enabled
6:3	Reserved	Must be kept at reset value.
2	UPS	Update source This bit is used to select the update event sources by software. 0: These events generate update interrupts or DMA requests: The UPG bit is set The counter generates an overflow or underflow event The restart mode generates an update event. 1: This event generates update interrupts or DMA requests: The counter generates an overflow or underflow event
1	UPDIS	Update disable.

This bit is used to enable or disable the update event generation.

0: Update event enable. When an update event occurs, the corresponding shadow registers are loaded with their preloaded values. These events generate update event:

- The UPG bit is set

- The counter generates an overflow or underflow event

- The restart mode generates an update event.

1: Update event disable.

**Note:** When this bit is set to 1, setting UPG bit or the restart mode does not generate an update event, but the counter and prescaler are initialized.

0	CEN	<p>Counter enable</p> <p>0: Counter disable</p> <p>1: Counter enable</p> <p>The CEN bit must be set by software when timer works in external clock, pause mode and encoder mode.</p>
---	-----	--

## Control register 1 (TIMERx\_CTL1)

Address offset: 0x04

Reset value: 0x0000

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

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

Bits	Fields	Descriptions
15:8	Reserved	Must be kept at reset value.
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: When a counter reset event occurs, a TRGO trigger signal is output. The counter reset source:</p> <ul style="list-style-type: none"> <li>Master timer generate a reset</li> <li>the UPG bit in the TIMERx_SWEVG register is set</li> </ul> <p>001: Enable. When a counter start event occurs, a TRGO trigger signal is output. The counter start source :</p> <ul style="list-style-type: none"> <li>CEN control bit is set</li> <li>The trigger input in pause mode is high</li> </ul> <p>010: When an update event occurs, a TRGO trigger signal is output. The update source depends on UPDIS bit and UPS bit.</p>

011: When a capture or compare pulse event occurs in channel0, a TRGO trigger signal is output.

100: When a compare event occurs, a TRGO trigger signal is output. The compare source is from O0CPRE.

101: Reserved

110: Reserved

111: Reserved

3            Reserved            Must be kept at reset value.

2:0          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														CH0IE	UPIE
rw														rw	rw

Bits	Fields	Descriptions
15:2	Reserved	Must be kept at reset value.
1	CH0IE	Channel 0 capture/compare interrupt enable 0: disabled 1: enabled
0	UPIE	Update interrupt enable 0: disabled 1: enabled

### **Interrupt flag register (TIMERx\_INTF)**

Address offset: 0x10

Reset value: 0x0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				CH0OF	Reserved.				CH0IF	UPIF				rc_w0	rc_w0
rc_w0														rc_w0	rc_w0

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

---

9	CH0OF	Channel 0 over capture flag  When channel 0 is configured in input mode, this flag is set by hardware when a capture event occurs while CH0IF flag has already been set. This flag is cleared by software.  0: No over capture interrupt occurred 1: Over capture interrupt occurred
8:2	Reserved	Must be kept at reset value.
1	CH0IF	Channel 0 's capture/compare interrupt flag  This flag is set by hardware and cleared by software. When channel 0 is in input mode, this flag is set when a capture event occurs. When channel 0 is in output mode, this flag is set when a compare event occurs.  If Channel0 is set to input mode, this bit will be reset by reading TIMERx_CH0CV. 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														CH0G	UPG
w														w	w

---

Bits	Fields	Descriptions
15:2	Reserved	Must be kept at reset value.
1	CH0G	Channel 0's capture or compare event generation  This bit is set by software in order to generate a capture or compare event in channel 0, it is automatically cleared by hardware. When this bit is set, the 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. 0: No generate a channel 1 capture or compare event 1: Generate a channel 1 capture or compare event
0	UPG	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.

- 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
Reserved.							Reserved	CH0COMCTL[2:0]		CH0COM SEN	CH0COM FEN	CH0MS[1:0]			
CH0CAPFLT[3:0]							CH0CAPPSC[1:0]								
							rw		rw		rw		rw		

#### **Output compare mode:**

Bits	Fields	Descriptions
15:7	Reserved	Must be kept at reset value.
6:4	CH0COMCTL[2:0]	<p>Channel 0 compare output control</p> <p>This bit-field specifies the compare output mode of the the output prepare signal O0CPRE. In addition, the high level of O0CPRE is the active level, and CH0_O and CH0_ON channels polarity 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 is equals to the output compare register TIMERx_CH0CV.</p> <p>010: Clear the channel output. O0CPRE signal is forced low when the counter is equals to the output compare register TIMERx_CH0CV.</p> <p>011: Toggle on match. O0CPRE toggles when the counter is equals to the output compare register TIMERx_CH0CV.</p> <p>100: Force low. O0CPRE is forced to low level.</p> <p>101: Force high. O0CPRE is forced to high level.</p> <p>110: PWM mode0. When counting up, O0CPRE is high when the counter is smaller than TIMERx_CH0CV, and low otherwise. When counting down, O0CPRE is low when the counter is larger than TIMERx_CH0CV, and high otherwise.</p> <p>111: PWM mode1. When counting up, O0CPRE is low when the counter is smaller than TIMERx_CH0CV, and high otherwise. When counting down, O0CPRE is high when the counter is larger than TIMERx_CH0CV, and low otherwise.</p> <p>If configured in PWM mode, the O0CPRE level changes only when the output compare mode is adjusted from “Timing” mode to “PWM” mode or the comparison result changes.</p> <p>This bit cannot be modified when PROT [1:0] bit-field in TIMERx_CCHP register is</p>

11 and CH0MS bit-field is 00(COMPARE MODE).

3	CH0COMSEN	<p>Channel 0 compare output shadow enable</p> <p>When this bit is set, the shadow register of TIMERx_CH0CV register, which updates at each update event, will be enabled.</p> <p>0: Channel 0 output compare shadow disable</p> <p>1: Channel 0 output compare shadow enable</p> <p>The PWM mode can be used without verifying the shadow register only in single pulse mode (when SPM=1)</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.</p> <p>1: Channel 0 output quickly compare enable.</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 programmed as output mode</p> <p>01: Channel 0 is programmed as input mode, IS0 is connected to CI0FE0</p> <p>10: Channel 0 is programmed as input mode, IS0 is connected to CI1FE0</p> <p>11: Channel 0 is programmed as input mode, IS0 is connected to ITS</p> <p><b>Note:</b> When CH0MS[1:0]=11, it is necessary to select an internal trigger input through TRGS bits in TIMERx_SMCFG register.</p>

#### **Input capture mode:**

Bits	Fields	Descriptions						
15:8	Reserved	Must be kept at reset value.						
7:4	CH0CAPFLT[3:0]	<p>Channel 0 input capture filter control</p> <p>The CI0 input signal can be filtered by digital filter and this bit-field configure the filtering capability.</p> <p>Basic principle of digital filter: continuously sample the CI0 input signal according to <math>f_{SAMP}</math> and record the number of times of the same level of the signal. After reaching the filtering capacity configured by this bit, it is considered to be an effective level.</p> <p>The filtering capability configuration is as follows:</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <th>CH0CAPFLT [3:0]</th> <th>Times</th> <th><math>f_{SAMP}</math></th> </tr> <tr> <td>4'b0000</td> <td colspan="2">Filter disabled.</td> </tr> </table>	CH0CAPFLT [3:0]	Times	$f_{SAMP}$	4'b0000	Filter disabled.	
CH0CAPFLT [3:0]	Times	$f_{SAMP}$						
4'b0000	Filter disabled.							

4'b0001	2	$f_{CK\_TIMER}$
4'b0010	4	
4'b0011	8	
4'b0100	6	$f_{DTS}/2$
4'b0101	8	
4'b0110	6	$f_{DTS}/4$
4'b0111	8	
4'b1000	6	$f_{DTS}/8$
4'b1001	8	
4'b1010	5	$f_{DTS}/16$
4'b1011	6	
4'b1100	8	
4'b1101	5	$f_{DTS}/32$
4'b1110	6	
4'b1111	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, input capture occurs on every channel input edge
	01:	The input capture occurs on every 2 channel input edges
	10:	The input capture occurs on every 4 channel input edges
	11:	The input capture occurs on every 8 channel input edges
1:0	CH0MS[1:0]	Channel 0 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..										CH0NP	Reserved	CH0P	CH0EN		
										rw		rw			

Bits	Fields	Descriptions
15:4	Reserved	Must be kept at reset value.
3	CH0NP	<p>Channel 0 complementary output polarity</p> <p>When channel 0 is configured in output mode, this bit specifies the complementary output signal polarity.</p> <p>0: Channel 0 complementary output high level is active level</p> <p>1: Channel 0 complementary output low level is active level</p>

		When channel 0 is configured in input mode, together with CH0P, this bit is used to define the polarity of CI0.
		This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 or 10.
2	Reserved	<p>Must be kept at reset value.</p> <p>Channel 0 capture/compare function polarity</p> <p>When channel 0 is configured in output mode, this bit specifies the output signal polarity.</p> <p>0: Channel 0 high level is active level</p> <p>1: Channel 0 low level is active level</p> <p>When channel 0 is configured in input mode, this bit specifies the CI0 signal polarity.</p> <p>[CH0NP, CH0P] will select the active trigger or capture polarity for CI0FE0 or CI1FE0.</p>
1	CH0P	<p>[CH0NP==0, CH0P==0]: CIxFE0's rising edge is the active signal for capture or trigger operation in slave mode. And CIxFE0 will not be inverted.</p> <p>[CH0NP==0, CH0P==1]: CIxFE0's falling edge is the active signal for capture or trigger operation in slave mode. And CIxFE0 will be inverted.</p> <p>[CH0NP==1, CH0P==0]: Reserved.</p> <p>[CH0NP==1, CH0P==1]: Reserved.</p> <p>This bit cannot be modified when PROT [1:0] bit-filed in TIMERx_CCHP register is 11 or 10.</p> <p>Channel 0 capture/compare function enable</p> <p>When channel 0 is configured in input mode, setting this bit enables CH0_O signal in active state. When channel 0 is configured in output mode, setting this bit enables the capture event in channel0.</p>
0	CH0EN	<p>0: Channel 0 disabled</p> <p>1: Channel 0 enabled</p>

## Counter register (TIMERx\_CNT)

Address offset: 0x24

Reset value: 0x0000

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

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]	<p>Prescaler value of the counter clock</p> <p>The TIMER_CK clock is divided by (PSC+1) to generate the counter clock. The value of this bit-field will be loaded to the corresponding shadow register at every update event.</p>

### **Counter auto reload register (TIMERx\_CAR)**

Address offset: 0x2C

Reset value: 0x0000

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

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><b>Note:</b> 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

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

### **Configuration register (TIMERx\_CFG )**

Address offset: 0xFC

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														CHVSEL	Reserved

rw

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
15:2	Reserved	Must be kept at reset value.
1	CHVSEL	<p>Write CHxVAL register selection</p> <p>This bit-field set and reset by software.</p> <p>1: If write the CHxVAL register, the write value is same as the CHxVAL value, the write access ignored</p> <p>0: No effect</p>
0	Reserved	Must be kept at reset value.

## 16.5. Basic timer (TIMERx, x=5, 6)

### 16.5.1. 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.

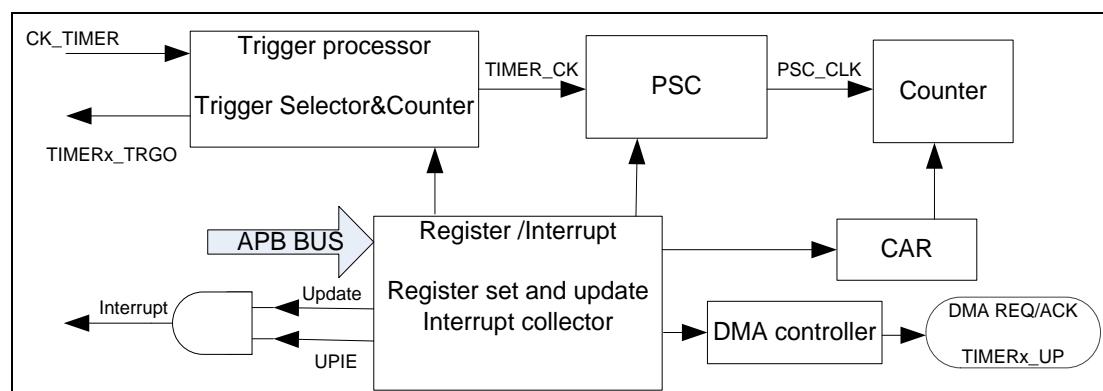
### **16.5.2. Characteristics**

- Counter width: 16 bits.
  - Source of count clock is internal clock only.
  - Multiple counter modes: count up.
  - Programmable prescaler: 16 bits. Factor can be changed on the go.
  - Auto-reload function.
  - Interrupt output or DMA request on update event.

### 16.5.3. Block diagram

[\*\*Figure 16-64. Basic timer block diagram\*\*](#) provides details on the internal configuration of the basic timer.

**Figure 16-64.** Basic timer block diagram

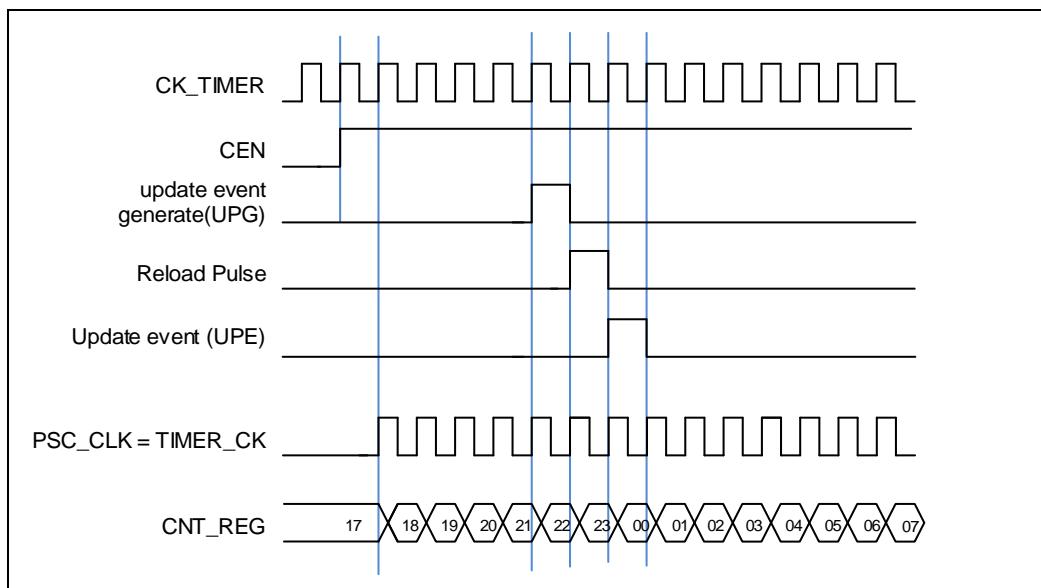


#### 16.5.4. Function overview

## Clock source configuration

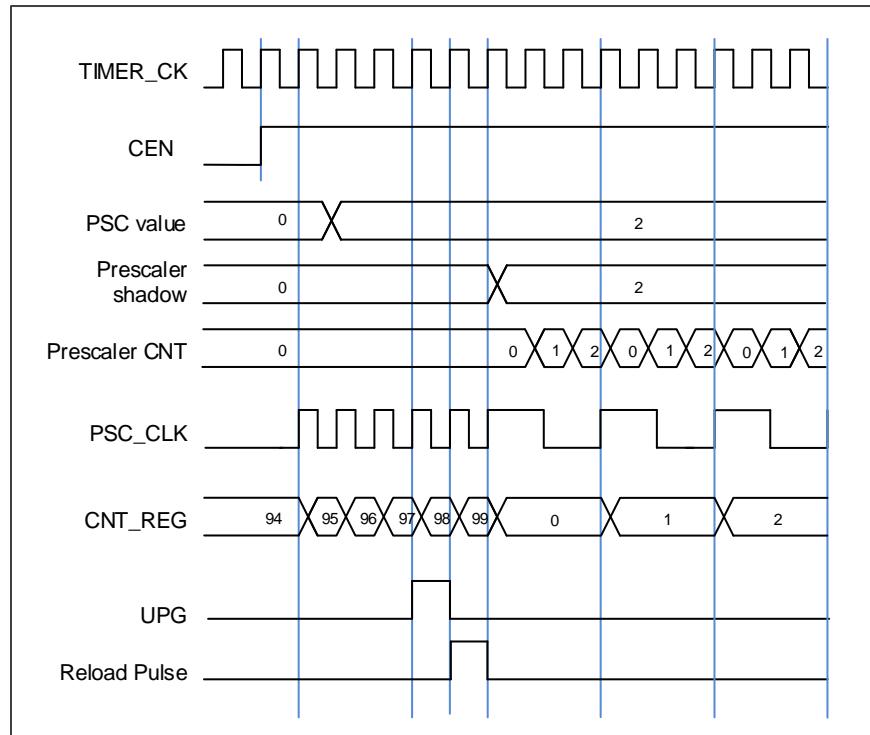
The basic TIMER can only be clocked by the internal timer clock CK\_TIMER, which is from the source named CK\_TIMER in RCU

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 16-65. Timing chart of internal clock divided by 1**


### Clock prescaler

The counter clock (PSC\_CLK) is obtained by the TIMER\_CK through the prescaler, and the prescale factor can be configured from 1 to 65536 through the prescaler register (TIMERx\_PSC). The new written prescaler value will not take effect until the next update event.

**Figure 16-66. Timing chart of PSC value change from 0 to 2**


## Counter up counting

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 will start counting up from 0 again. 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 shadow registers (counter auto reload register, prescaler register) are updated.

The following figures show some examples of the counter behavior for different clock prescaler factor when TIMERx\_CAR=0x99.

**Figure 16-67. Timing chart of up counting mode, PSC=0/2**

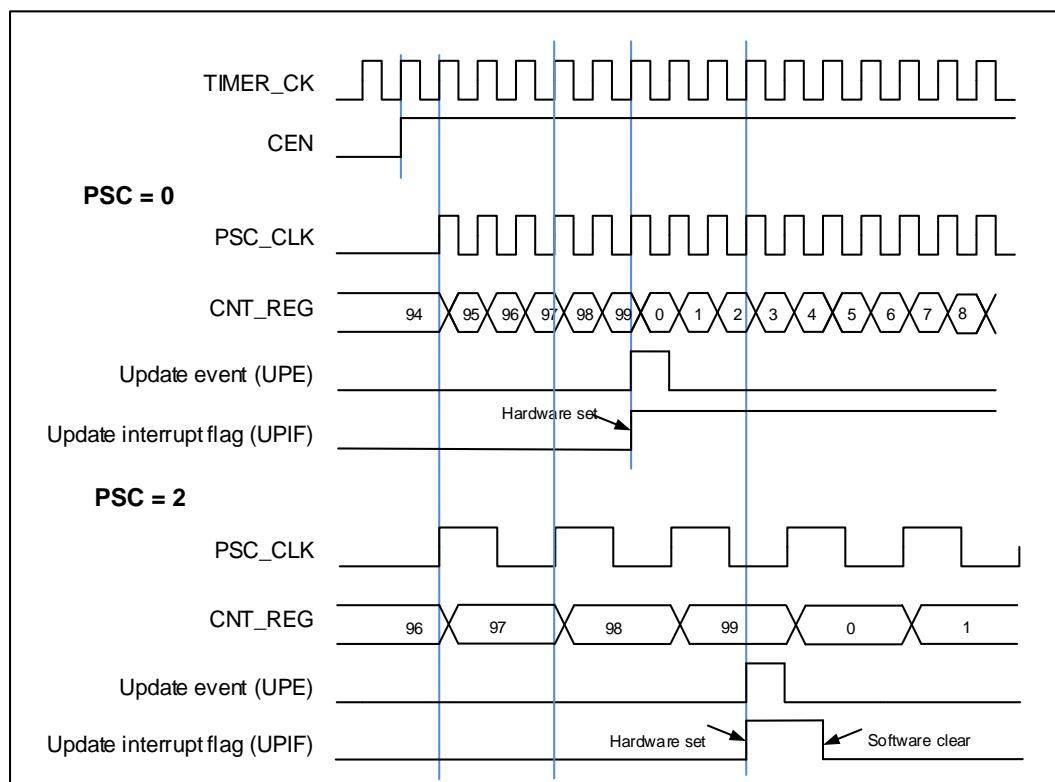
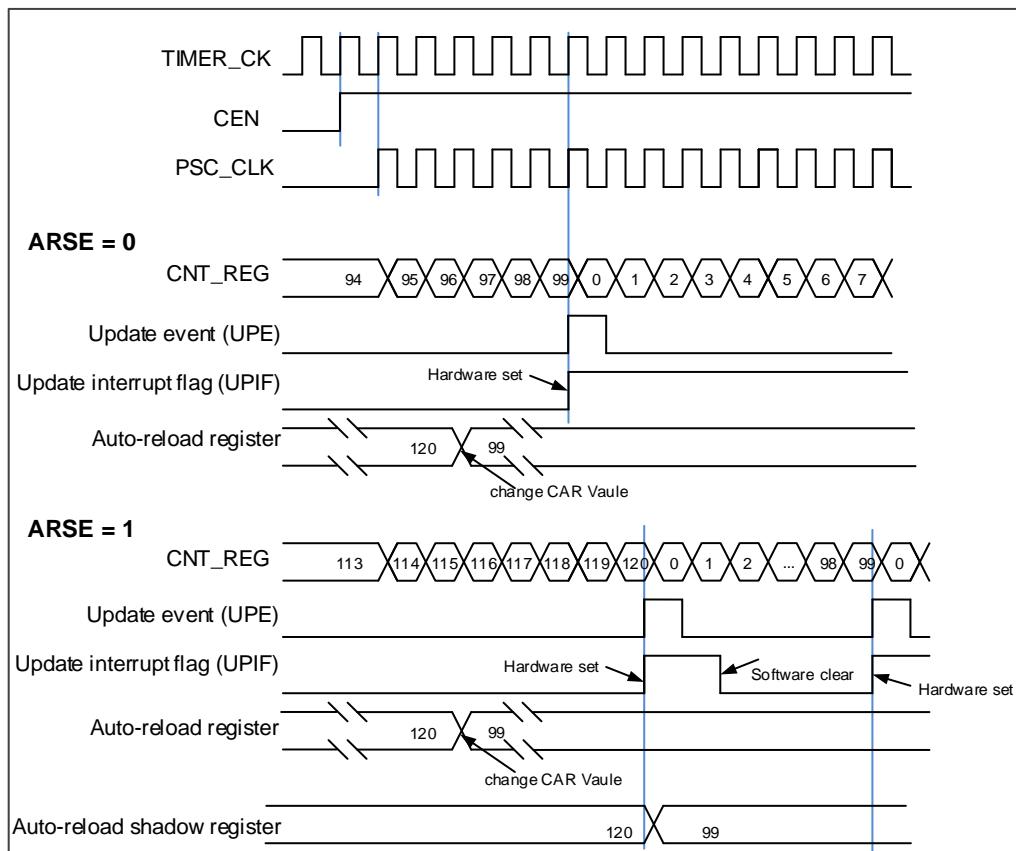


Figure 16-68. Timing chart of up counting mode, change TIMERx\_CAR ongoing



### Single pulse mode

Single pulse mode is opposite to the repetitive mode, which can be enabled by setting SPM in TIMERx\_CTL0. When you set SPM, the counter will be clear and stop when the next update event.

Once the timer is set to operate in the single pulse mode, it is necessary to set the timer enable bit CEN in the TIMERx\_CTL0 register to 1 to enable the counter, then the CEN bit keeps at a high state until the update event occurs or the CEN bit is written to 0 by software. If the CEN bit is cleared to 0 using software, the counter will be stopped and its value held.

### Timer debug mode

When the Cortex®-M4 halted, and the TIMERx\_HOLD configuration bit in DBG\_CTL0 register set to 1, the TIMERx counter stops.

### **16.5.5. TIMERx registers(x=5, 6)**

TIMER5 base address: 0x4000 1000

TIMER6 base address: 0x4000 1400

## 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	

Bits	Fields	Descriptions
15:8	Reserved	Must be kept at reset value.
7	ARSE	<p>Auto-reload shadow enable</p> <p>0: The shadow register for TIMERx_CAR register is disabled</p> <p>1: The shadow register for TIMERx_CAR register is enabled</p>
6:4	Reserved	Must be kept at reset value.
3	SPM	<p>Single pulse mode.</p> <p>0: Single pulse mode disable. The counter continues after update event.</p> <p>1: Single pulse mode enable. The counter counts until the next update event occurs.</p>
2	UPS	<p>Update source</p> <p>This bit is used to select the update event sources by software.</p> <p>0: These events generate update interrupts or DMA requests:</p> <ul style="list-style-type: none"> <li>The UPG bit is set</li> <li>The counter generates an overflow or underflow event</li> <li>The restart mode generates an update event.</li> </ul> <p>1: This event generates update interrupts or DMA requests:</p> <p>The counter generates an overflow or underflow event</p>
1	UPDIS	<p>Update disable.</p> <p>This bit is used to enable or disable the update event generation.</p> <p>0: Update event enable. When an update event occurs, the corresponding shadow registers are loaded with their preloaded values. These events generate update event:</p> <ul style="list-style-type: none"> <li>The UPG bit is set</li> <li>The counter generates an overflow or underflow event</li> <li>The restart mode generates an update event.</li> </ul>

1: Update event disable.

**Note:** When this bit is set to 1, setting UPG bit or the restart mode does not generate an update event, but the counter and prescaler are initialized.

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.
---	-----	---

### 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: When a counter reset event occurs, a TRGO trigger signal is output. The counter reset source:</p> <ul style="list-style-type: none"> <li>Master timer generate a reset</li> <li>the UPG bit in the TIMERx_SWEVG register is set</li> </ul> <p>001: Enable. When a counter start event occurs, a TRGO trigger signal is output. The counter start source :</p> <ul style="list-style-type: none"> <li>CEN control bit is set</li> <li>The trigger input in pause mode is high</li> </ul> <p>010: When an update event occurs, a TRGO trigger signal is output. The update source depends on UPDIS bit and UPS bit.</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															

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
15:1	Reserved	Must be kept at reset value.
0	UPG	<p>This bit can be set by software, and cleared by hardware automatically. When this bit is set, the counter is cleared. The prescaler counter is cleared at the same time.</p> <p>0: No generate an update event 1: Generate an update event</p>

### Counter register (TIMERx\_CNT)

Address offset: 0x24

Reset value: 0x0000

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

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]	<p>Prescaler value of the counter clock</p> <p>The TIMER_CK 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)

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><b>Note:</b> 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>

## 17. Universal synchronous/asynchronous receiver/transmitter (USART)

### 17.1. Overview

The Universal Synchronous/Asynchronous Receiver/Transmitter (USART) provides a flexible serial data exchange interface. Data frames can be transferred in full duplex or half duplex mode, synchronously or asynchronously through this interface. A programmable baud rate generator divides the UCLK (PCLK1 or PCLK2) to produce a dedicated baud rate clock for the USART transmitter and receiver.

Besides the standard asynchronous receiver and transmitter mode, the USART implements several other types of serial data exchange modes, such as IrDA (infrared data association) SIR mode, smartcard mode, LIN (local interconnection network) mode, 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 polarity of the data bits and the TX/RX pins can be configured independently and flexibly.

The USART supports DMA function for high-speed data communication, except UART4.

### 17.2. Characteristics

- NRZ standard format.
- Asynchronous, full duplex 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 7.5 MBit/s (PCLK2 120M 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)

- Block mode (T=1)
- Direct and inverse convention
- Multiprocessor communication
  - Enter into mute mode if address match does not occur.
  - Wake up from mute mode by idle frame or address match detection.
- Various status flags:
  - Flags for transfer detection: receive buffer not empty (RBNE), transmit buffer empty (TBE), transfer complete (TC), and busy (BSY).
  - 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).
  - Flags for smartcard block mode: end of block (EBF) and receiver timeout (RTF).
  - 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.

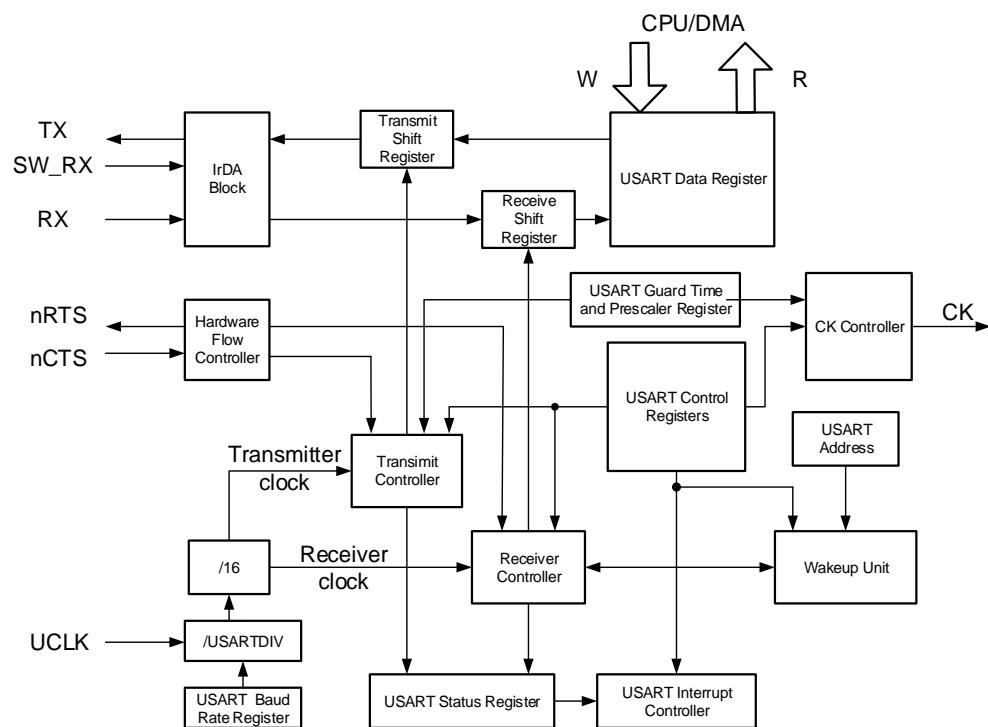
- Smartcard mode
- Synchronous mode
- Hardware flow control protocol (CTS/RTS)
- Configurable data polarity

### 17.3. Function overview

The interface is externally connected to another device by the main pins listed in [\*\*Table 17-1. Description of USART important pins\*\*](#).

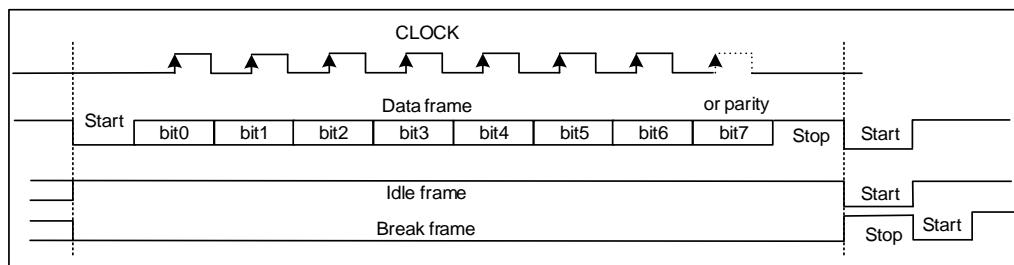
**Table 17-1. Description of USART important pins**

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 17-1. USART module block diagram**


### 17.3.1. USART frame format

The USART frame starts with a start bit and ends up with a number of stop bits. The length of the data frame is configured by the WL bit in the USART\_CTL0 register. The last data bit can be used as parity check bit by setting the PCEN bit 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 17-2. 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 17-2. Configuration of stop bits**

STB[1:0]	stop bit length (bit)	usage description
00	1	default value
01	0.5	Smartcard mode for receiving

<b>STB[1:0]</b>	<b>stop bit length (bit)</b>	<b>usage description</b>
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.

The break frame structure is a 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 UCLK and the configuration of the baud rate generator.

### 17.3.2. Baud rate generation

The baud-rate divider is a 16-bit number which consists of a 12-bit integer and a 4-bit fractional part. The number formed by these two values is used by the baud rate generator to determine the bit period. Having a fractional baud-rate divider allows the USART to generate all the standard baud rates.

The baud-rate divider (USARTDIV) has the following relationship with the peripheral clock:

$$\text{USARTDIV} = \frac{\text{UCLK}}{16 \times \text{Baud Rate}} \quad (16-1)$$

For example, when oversampled by 16:

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.

### 17.3.3. USART transmitter

If the transmit enable bit (TEN) in USART\_CTL0 register is set, when the transmit data buffer is not empty, the transmitter shifts out the transmit data frame through the TX pin. The polarity of the TX pin can be configured by the TINV bit in the USART\_CTL3 register. Clock pulses can be output through the CK pin.

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 in the USART\_STAT0 register is asserted. The TBE bit is cleared by writing to the

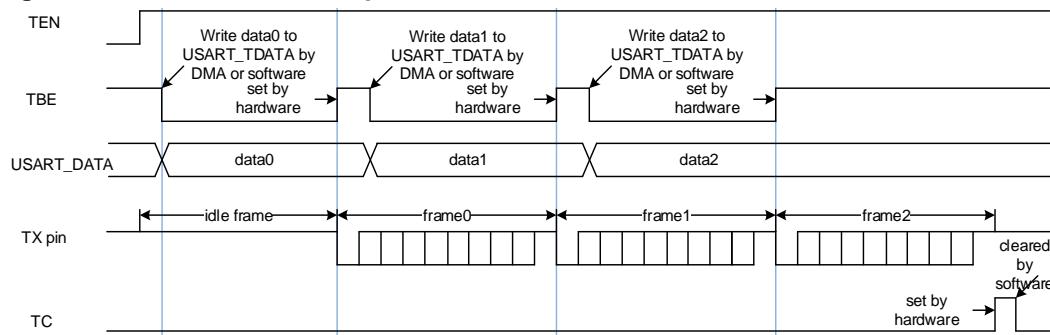
USART\_DATA register and it is set by hardware after the data is put into the transmit shift register. If a data is written to the USART\_DATA register while a transmission is ongoing, it will be firstly stored in the transmit buffer, and transferred to the transmit shift register after the current transmission 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\_STAT0 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 17-3. USART transmit procedure](#). The software operating process is as follows:

1. Set the UEN bit in USART\_CTL0 to enable the USART.
2. Write the WL bit in USART\_CTL0 to set the data bits length.
3. Set the STB[1:0] bits in USART\_CTL1 to configure the number of stop bits.
4. Enable DMA (DENT bit) in USART\_CTL2 if multibuffer communication is selected.
5. Set the baud rate in USART\_BAUD.
6. Set the TEN bit in USART\_CTL0.
7. Wait for the TBE to be asserted.
8. Write the data to the USART\_DATA register.
9. Repeat step 7-8 for each data, if DMA is not enabled.
10. Wait until TC=1 to finish.

**Figure 17-3. 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\_STAT0 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.

#### 17.3.4. USART receiver

After power on, the USART receiver can be enabled by the following procedure:

1. Set the UEN bit in USART\_CTL0 to enable the USART.
2. Write the WL bit in USART\_CTL0 to set the data bits length.
3. Set the STB[1:0] bits in USART\_CTL1.

4. Enable DMA (DENR bit) in USART\_CTL2 if multibuffer communication is selected.
5. Set the baud rate in USART\_BAUD.
6. Set the REN bit in USART\_CTL0.

After being enabled, the receiver receives a bit stream after a valid start pulse has been detected. Detection on noisy error, parity error, frame error and overrun error is performed during the reception of a frame.

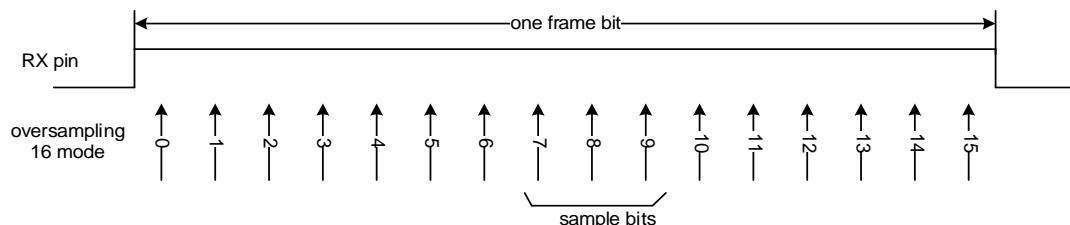
When a frame is received, the RBNE bit in USART\_STAT0 is asserted, an interrupt is generated if the corresponding interrupt enable bit (RBNEIE) is set in the USART\_CTL0 register. The status bits of the reception are stored in the USART\_STAT0 register.

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 is generated, if the receive DMA is enabled and the ERRIE bit in USART\_CTL2 register is set.

**Figure 17-4. 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\_STAT0 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\_STAT0 register will be set. An interrupt will be 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\_STAT0 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.

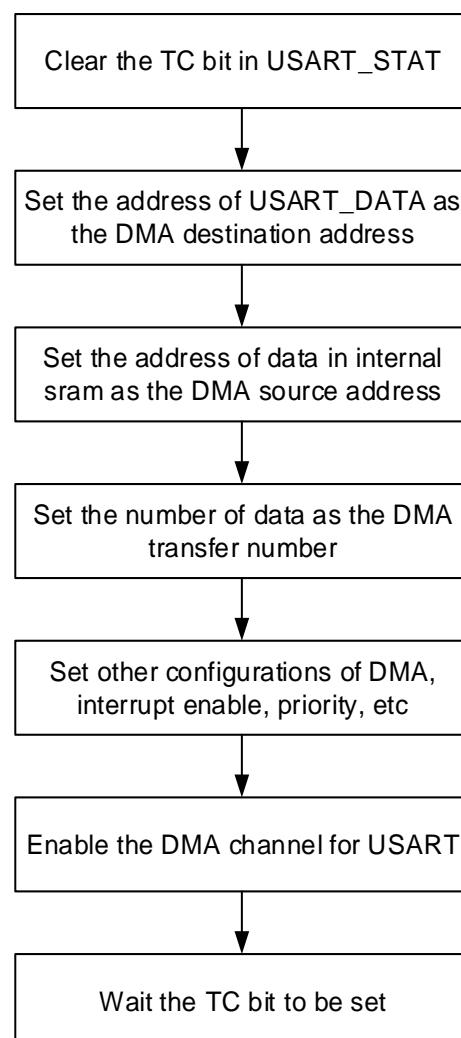
If a noise error (NERR), parity error (PERR), frame error (FERR) or overrun error (ORERR) is generated during a receiving process, then NERR, PERR, FERR or ORERR will be set at same time with RBNE. If DMA is disabled, the software needs to check whether the RBNE interrupt is caused by noise error, parity error, framing error or overflow error when the RBNE interrupt occurs.

### 17.3.5. Use DMA for data buffer access

To reduce the burden of the processor, DMA can be used to access the transmitting and receiving data buffer. The DENT bit in USART\_CTL2 is used to enable the DMA transmission, and the DENR bit in USART\_CTL2 is used to enable the DMA reception.

When DMA is used for USART transmission, DMA transfers data from internal SRAM to the transmit data buffer of the USART. The configuration steps are shown in [Figure 17-5. Configuration step when using DMA for USART transmission](#).

**Figure 17-5. Configuration step when using DMA for USART transmission**

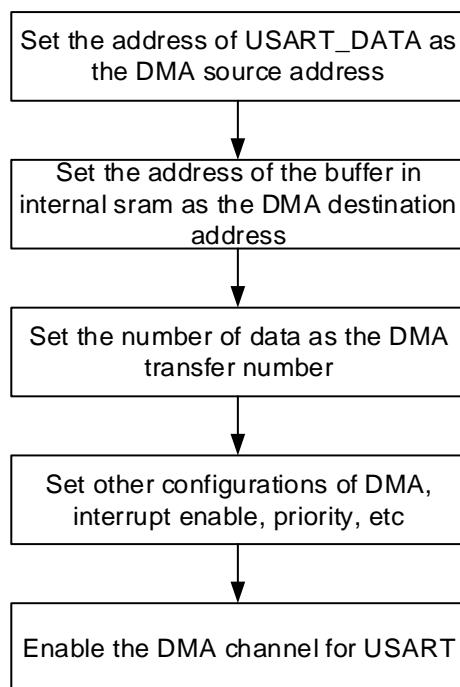


After all of the data frames are transmitted, the TC bit in USART\_STAT0 is set. An interrupt

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 17-6. 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\_STAT0.

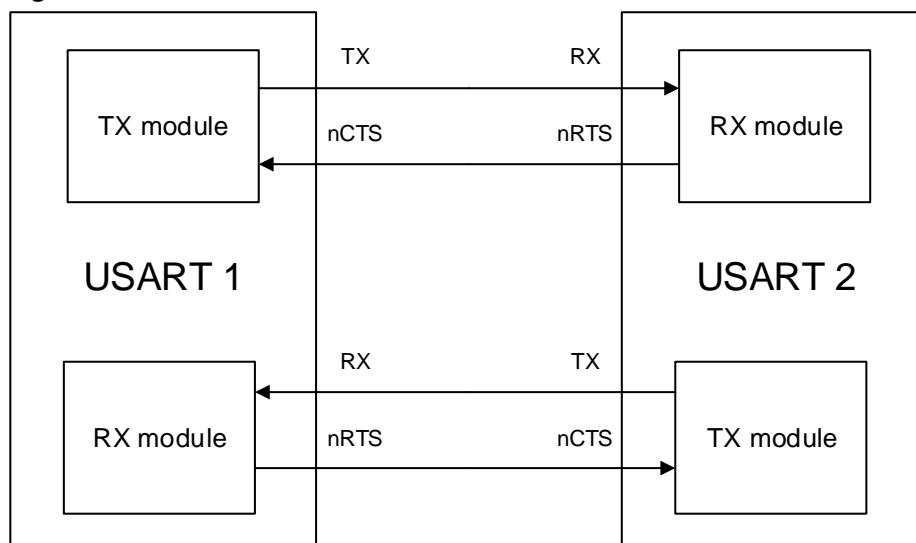
**Figure 17-6. Configuration steps when using DMA for USART reception**



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.

### 17.3.6. Hardware flow control

The hardware flow control function is realized by the nCTS and nRTS pins. The RTS flow control is enabled by writing '1' to the RTSEN bit in USART\_CTL2 and the CTS flow control is enabled by writing '1' to the CTSEN bit in USART\_CTL2.

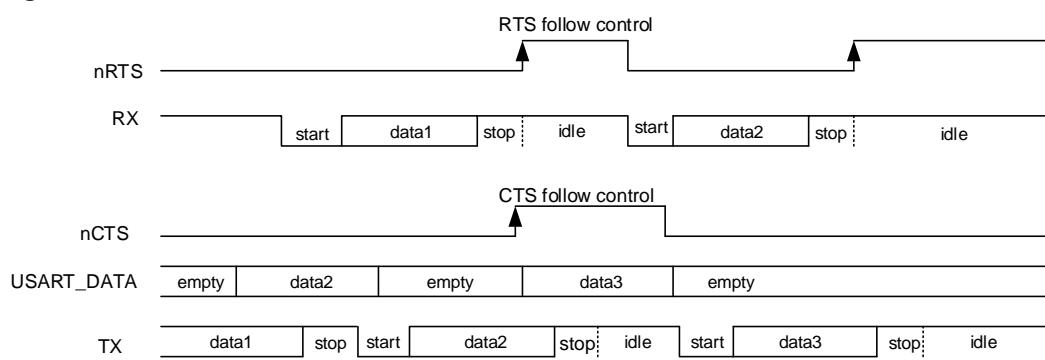
**Figure 17-7. Hardware flow control between two USARTs**


### RTS flow control

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\_STAT0 is '0' and the nCTS signal is low, the transmitter transmits the data frame. When the nCTS signal goes high during a transmission, the transmitter stops after the current transmission is accomplished.

**Figure 17-8. Hardware flow control**


If the CTS flow control is enabled, the CTSF bit in USART\_STAT0 is set when the nCTS pin toggles. An interrupt is generated if the CTSIE bit in USART\_CTL2 is set.

### 17.3.7. Multi-processor communication

In multiprocessor communication, several USARTs are connected as a network. It will be a

big burden for a device to monitor all of the messages on the RX pin. To reduce the burden of a device, software can put an USART module into a mute mode by 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\_STAT0 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\_STAT0 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 flag.

### 17.3.8. 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.

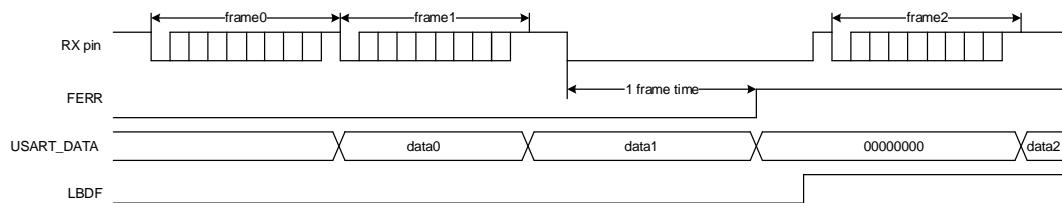
When transmitting a normal data frame, the transmission procedure is the same as the normal USART mode. When the SBKCMD bit in USART\_CTL0 is set, the USART transmits 13 '0' bits continuously, followed by 1 stop bit.

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 bit in USART\_STAT0 is set. An interrupt occurs if the LBDIE bit in USART\_CTL1 is set.

As shown in [\*\*Figure 17-9. Break frame occurs during idle state\*\*](#), if a break frame occurs during the idle state on the RX pin, the USART receiver will receive an all '0' frame, with an

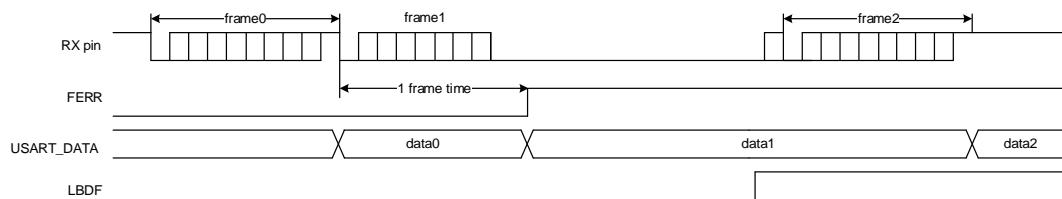
asserted FERR status.

**Figure 17-9. Break frame occurs during idle state**



As shown in [Figure 17-10. Break frame occurs during a frame](#), if a break frame occurs during a frame on the RX pin, the FERR status will be asserted for the current frame.

**Figure 17-10. Break frame occurs during a frame**



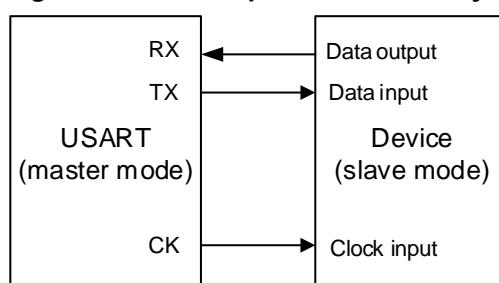
### 17.3.9. Synchronous mode

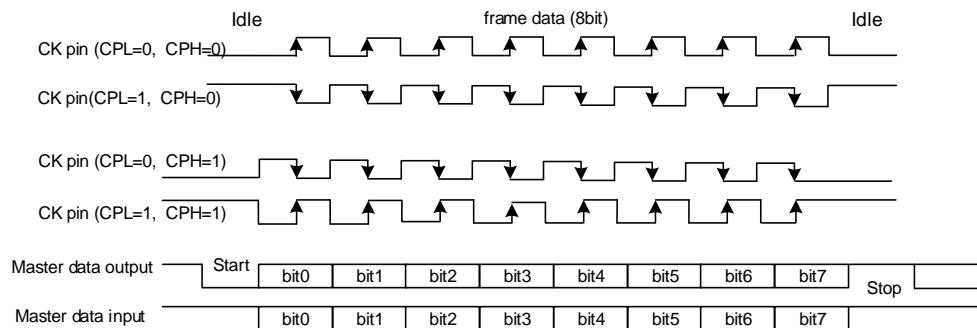
The USART can be used for full-duplex synchronous serial communications only in master mode, by setting the CKEN bit in USART\_CTL1. The LMEN bit in USART\_CTL1 and SCEN, HDEN, IREN bits in USART\_CTL2 should be cleared in synchronous mode. The CK pin is the clock output of the synchronous USART transmitter, 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 17-11. Example of USART in synchronous mode**

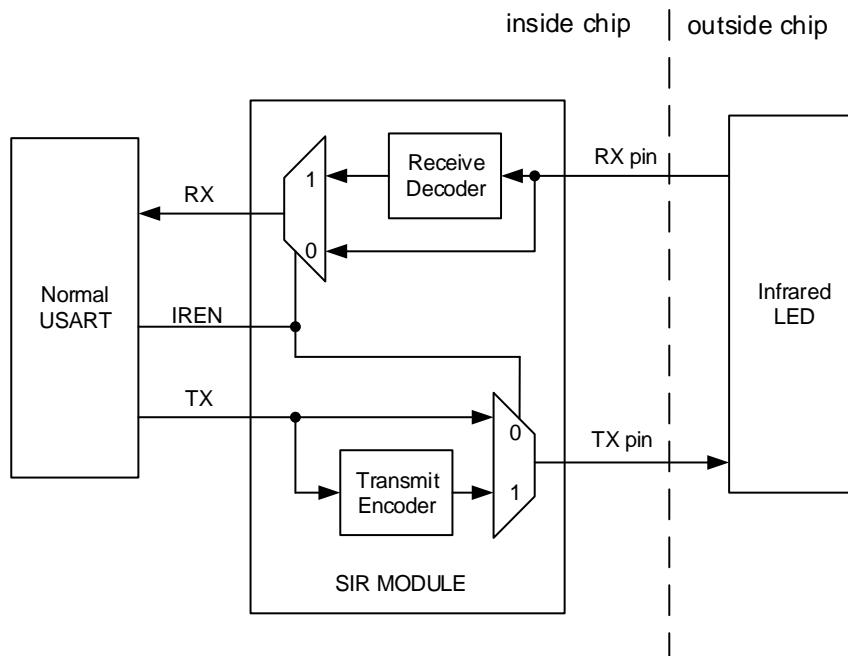


**Figure 17-12. 8-bit format USART synchronous waveform (CLEN=1)**


### 17.3.10. IrDA SIR ENDEC mode

The IrDA mode is enabled by setting the IREN bit in USART\_CTL2. The LMEN, STB[1:0], CKEN bits in USART\_CTL1 and HDEN, SCEN bits in USART\_CTL2 should be 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.

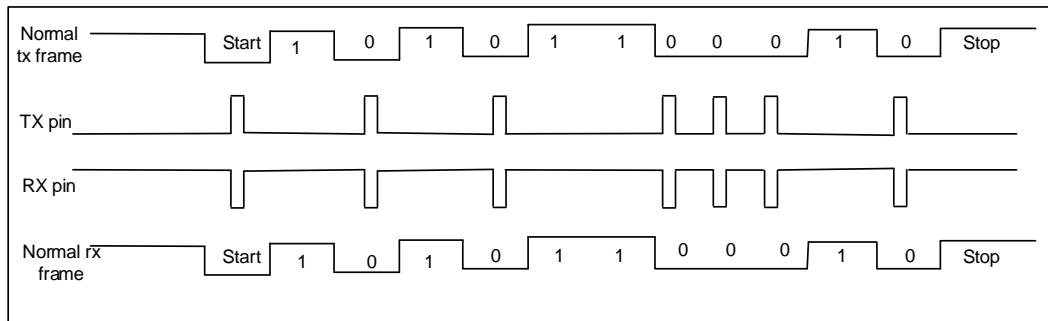
**Figure 17-13. IrDA SIR ENDEC module**


In IrDA mode, the polarity of the TX and RX pins is different. The TX pin is usually at low state, while the RX pin is usually at high state. The IrDA pins keep stable to represent the logic '1', while an infrared light pulse on the IrDA pins (a Return to Zero signal) represents the logic '0'. The pulse width should be 3/16 of a bit period. The IrDA could not detect any pulse if the pulse

width is less than 1 PSC clock. While it can detect a pulse by chance if the pulse width is greater than 1 but smaller than 2 times PSC clock.

Because the IrDA is a half-duplex protocol, the transmission and the reception should not be carried out at the same time in the IrDA SIR ENDEC block.

**Figure 17-14. IrDA data modulation**



The SIR 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.

### 17.3.11. Half-duplex communication mode

The half-duplex communication mode is enabled by setting the HDEN bit in USART\_CTL2. The LMEN, CKEN bits in USART\_CTL1 and SCEN, IREN bits in USART\_CTL2 should be 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.

### 17.3.12. Smartcard (ISO7816-3) mode

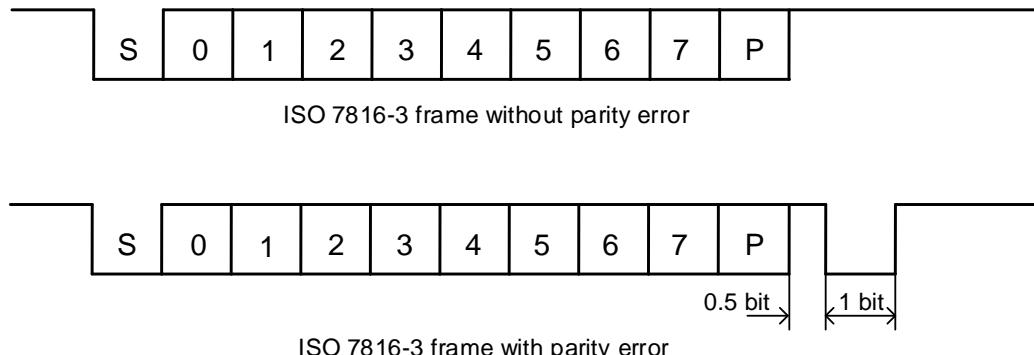
The smartcard mode is an asynchronous mode, which is designed to support the ISO7816-3 protocol. Both the character ( $T=0$ ) mode and the block ( $T=1$ ) mode are supported. The smartcard mode is enabled by setting the SCEN bit in USART\_CTL2. The LMEN bit in USART\_CTL1 and HDEN, IREN bits in USART\_CTL2 should be cleared in smartcard mode.

A clock is provided to the external smartcard through the CK pin after the CKEN bit is set. The clock is divided from the PCLK. The division 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 mode, 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 17-15. ISO7816-3 frame format**



### Character (T=0) mode

Comparing 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 for SCRTNUM times. An interframe gap of 2.5 bits time will be inserted before the start of a resented frame. At the end of the last repeated character the TC bit is set immediately without guard time. The USART will stop transmitting and assert the framing error status if it still receives the NACK signal after the programmed number of retries. 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 smartcard. Then a frame error occurs in smartcard side. The RBNE/receive DMA request is not activated if the received character is erroneous. According to the protocol, the smartcard can resend the data. The USART stops transmitting the NACK and the error is regarded as a parity error if the received character is still erroneous after the maximum number of retries which is specified in the SCRTNUM bit field. The NACK signal is enabled by setting the NKEN bit in USART\_CTL2.

The idle frame and break frame are not supported in the smartcard mode.

### Block (T=1) mode

In block (T=1) mode, the NKEN bit in the USART\_CTL2 register should be cleared to deactivate the NACK transmission.

When requesting a read from the smartcard, the RT[23:0] bits in USART\_RT register should be programmed with the BWT (block wait time) - 11 value and RBNEIE must be set. This timeout period is expressed in baud time units. The RTF bit in USART\_STAT1 will be asserted, if no answer is received from the card before the expiration of this period. An interrupt is generated if the RTIE bit in USART\_CTL3 is set. The USART generates a RBNE interrupt if the first character is received before the expiration of the RT[23:0] period. If DMA is used to read from the smartcard in block mode, the DMA must be enabled only after the first character is received.

After the first character is received, the RT[23:0] bits should be configured to the CWT (character wait time) - 11 to enable the automatic check of the maximum interframe gap between two consecutive characters. The RTF bit in USART\_STAT1 will be asserted, if the smartcard stops sending characters in the RT[23:0] period.

The USART uses a block length counter, which is reset when the USART is transmitting (TBE=0), to count the number of received characters. The length of the block, which must be programmed to the BL[7:0] bits in the USART\_RT register, is received from the smartcard in the third byte of the block (prologue field). The block length counter counts up from 0 to the maximum value of BL[7:0]+4. The end of the block status (EBF bit in USART\_STAT1) is set after the block length counter reaches the maximum value. An interrupt is generated if the EBIE bit in USART\_CTL3 is set. The RTF bit may be set in case that an error in the block length.

If DMA is used for reception, this register field must be programmed to the minimum value (0x0) before the start of the block. With this value, the end of the block interrupt occurs after the 4th received character. The block length value can be read from the receive buffer at the third byte.

If DMA is not used for reception, the BL[7:0] bits should be firstly configured with the maximum value 0xFF to avoid generating an EBF status. The real block length value can be reconfigured to the BL[7:0] bits after the third byte is received.

### Direct and inverse convention

The smartcard protocol defines two conventions: direct and inverse.

When the directed convention is selected, the LSB of the data frame is transferred first, high state on the TX pin represents logic '1', the parity check mode is even. In this case the MSBF and DINV bits in USART\_CTL3 should be cleared.

When the inverse convention is selected, the MSB of the data frame is transferred first, high state on the TX pin represents logic '0', the parity check mode is even. In this case the MSBF and DINV bits in USART\_CTL3 should be set.

### 17.3.13. USART interrupts

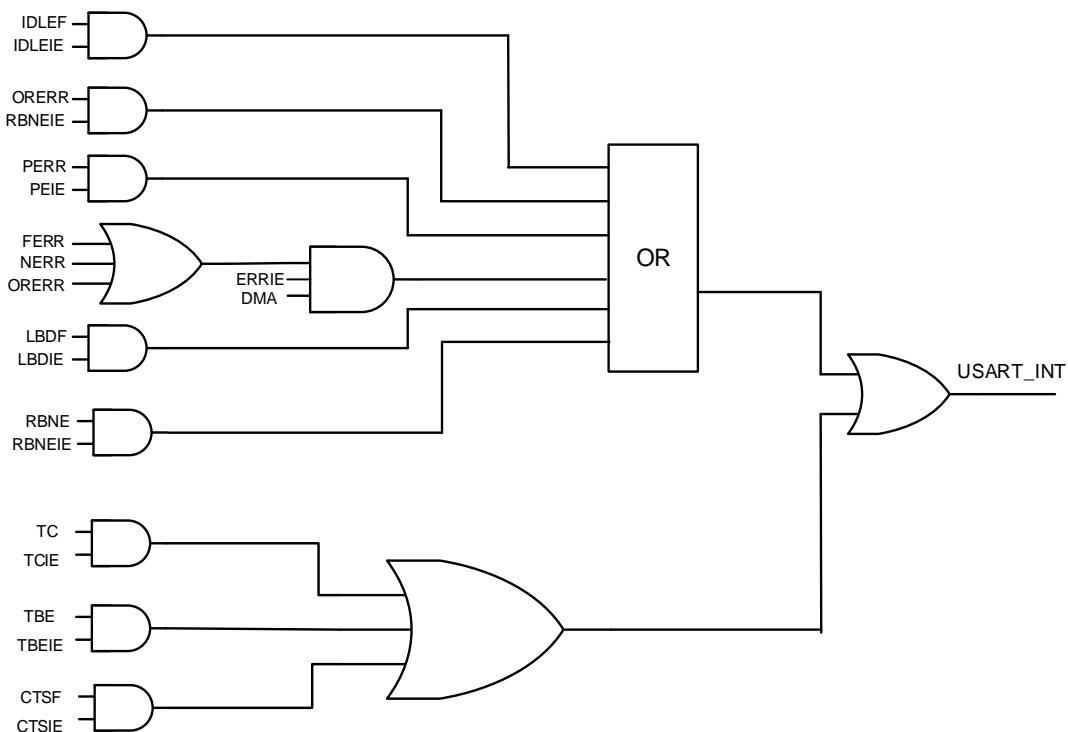
The USART interrupt events and flags are listed in [Table 17-3. USART interrupt requests](#).

Table 17-3. 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
Receiver timeout	RTF	USART_CTL3	RTIE
End of block	EBF	USART_CTL3	EBIE
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 17-16. USART interrupt mapping diagram



## 17.4. 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

### 17.4.1. Status register 0 (USART\_STAT0)

Address offset: 0x00

Reset value: 0x0000 00C0

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					CTSF	LBDF	TBE	TC	RBNE	IDLEF	ORERR	NERR	FERR	PERR	
rc_w0					rc_w0	r	rc_w0	rc_w0	r	r	r	r	r	r	

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 not available for UART3/4.</p>
8	LBDF	<p>LIN break detected flag.</p> <p>LMEN bit in USART_CTL1 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>

		1: Transmit data buffer is empty.
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> <p>Software can clear this bit by writing 0 to it.</p> <p>0: Transmission of current data is not complete.</p> <p>1: Transmission of current data is complete.</p>
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.</p> <p>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_STAT0 and USART_DATA registers one by one.</p> <p>0: The USART module does not detect an IDLE frame.</p> <p>1: The USART module has detected an IDLE frame.</p>
3	ORERR	<p>Overrun error</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 the ERRIE bit in USART_CTL2 is set.</p> <p>Software can clear this bit by reading the USART_STAT0 and USART_DATA registers one by one.</p> <p>0: The USART does not detect a overrun error.</p> <p>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_STAT0 and USART_DATA registers one by one.</p> <p>0: The USART does not detect a noise error.</p> <p>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_STAT0 and USART_DATA</p>

registers one by one.

0: The USART does not detect a framing error.

1: The USART has detected a framing error.

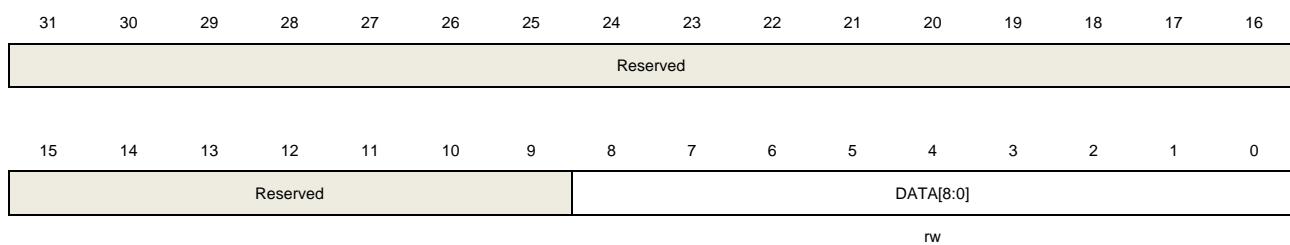
0	PERR	Parity error flag
		This bit is set when the parity bit of a receive frame does not match the expected parity value. An interrupt occurs if the PERRIE bit in USART_CTL0 is set. Software can clear this bit in the sequence: read the USART_STAT0 register, and then read or write the USART_DATA register.
		0: The USART does not detect a parity error.
		1: The USART has detected a parity error.

#### 17.4.2. Data register (USART\_DATA)

Offset: 0x04

Reset value: Undefined

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



Bits	Fields	Descriptions
31:9	Reserved	Must be kept the reset value.
8:0	DATA[8:0]	<p>Transmit or read data value.</p> <p>Software can write these bits to update the transmit data or read these bits to get the receive data.</p> <p>If the parity check function is enabled, when transmit 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>

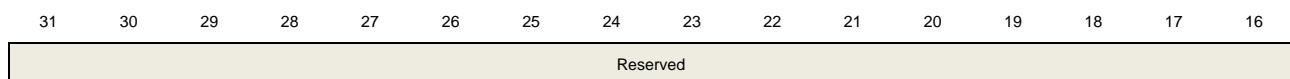
#### 17.4.3. Baud rate register (USART\_BAUD)

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).



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

Bits	Fields	Descriptions
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.

#### 17.4.4. 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	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
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_STAT0 is set. 0: Parity error interrupt is disabled. 1: Parity error interrupt is enabled.
7	TBEIE	Transmitter buffer empty interrupt enable.  If this bit is set, an interrupt occurs when the TBE bit in USART_STAT0 is set. 0: Transmitter buffer empty interrupt is disabled. 1: Transmitter buffer empty interrupt is enabled.
6	TCIE	Transmission complete interrupt enable.  If this bit is set, an interrupt occurs when the TC bit in USART_STAT0 is set. 0: Transmission complete interrupt is disabled. 1: Transmission complete interrupt is enabled.
5	RBNEIE	Read data buffer not empty interrupt and overrun error interrupt enable.  If this bit is set, an interrupt occurs when the RBNE bit or the ORERR bit in USART_STAT0 is set. 0: Read data register not empty interrupt and overrun error interrupt disabled. 1: Read data register not empty interrupt and overrun error interrupt enabled.
4	IDLEIE	IDLE line detected interrupt enable.  If this bit is set, an interrupt occurs when the IDLEF bit in USART_STAT0 is set. 0: IDLE line detected interrupt disabled. 1: IDLE line detected interrupt enabled.
3	TEN	Transmitter enable 0: Transmitter is disabled. 1: Transmitter is enabled.
2	REN	Receiver enable 0: Receiver is disabled. 1: Receiver is enabled.
1	RWU	Receiver wakeup from mute mode..  Software can set this bit to make the USART work in mute mode and reset this bit to wake up the USART.  In wake up by idle frame mode (WM=0), this bit can be reset by hardware when an idle frame has been detected. In wake up by address match mode (WM=1), this bit can be reset by hardware when receiving an address match frame or set by hardware when receiving an address mismatch frame. 0: Receiver in active mode. 1: Receiver in mute mode.
0	SBKCMD	Send break command  Software can set this to send a break frame.  Hardware resets this bit automatically when the break frame has been transmitted. 0: Do not transmit a break frame.

1: Transmit a break frame.

### 17.4.5. 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                    rw                    rw                    rw                    rw                    rw                    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.
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 bit 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 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 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 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 is reserved for UART3/4.

7	Reserved	Must be kept the reset value.
6	LBDIE	<p>LIN break detected interrupt enable</p> <p>If this bit is set, an interrupt occurs when the LBDF bit in USART_STAT0 is set.</p> <p>0: LIN break detected interrupt is disabled</p> <p>1: LIN break detected interrupt is enabled</p>
5	LBLEN	<p>LIN break frame length</p> <p>This bit specifies the length of a LIN break frame.</p> <p>0: 10 bit</p> <p>1: 11 bit</p>
4	Reserved	Must be kept the reset value.
3:0	ADDR[3:0]	<p>Address of the USART</p> <p>In wake up by address match mode (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.</p>

#### 17.4.6. 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	rw	rw

Bits	Fields	Descriptions
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_STAT0 is set.</p> <p>0: CTS interrupt is disabled.</p> <p>1: CTS interrupt is enabled.</p> <p>This bit is reserved for UART3/4.</p>
9	CTSEN	CTS enable

		This bit enables the CTS hardware flow control function. 0: CTS hardware flow control disabled. 1: CTS hardware flow control enabled. This bit is reserved for UART3/4.
8	RTSEN	RTS enable  This bit enables the RTS hardware flow control function. 0: RTS hardware flow control disabled. 1: RTS hardware flow control enabled. This bit is reserved for UART3/4.
7	DENT	DMA request enable for transmission. 0: DMA request is disabled for transmission. 1: DMA request is enabled for transmission.
6	DENR	DMA request enable for reception. 0: DMA request is disabled for reception. 1: DMA request is enabled for reception.
5	SCEN	Smartcard mode enable  This bit enables the smartcard work mode. 0: Smartcard Mode disabled. 1: Smartcard Mode enabled. This bit is reserved for UART3/4.
4	NKEN	NACK enable in smartcard mode  This bit enables the NACK transmission when parity error occurs in smartcard mode. 0: Disable NACK transmission. 1: Enable NACK transmission. This bit is reserved for UART3/4.
3	HDEN	Half-duplex enable  This bit enables the half-duplex USART mode. 0: Half duplex mode is disabled. 1: Half duplex mode is enabled.
2	IRLP	IrDA low-power  This bit selects low-power mode of IrDA mode. 0: Normal mode 1: Low-power mode
1	IREN	IrDA mode enable  This bit enables the IrDA mode of USART. 0: IrDA disabled 1: IrDA enabled
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\_STAT0 is set.

- 0: Error interrupt disabled.
- 1: Error interrupt enabled.

## 17.4.7. Guard time and prescaler register (USART\_GP)

Address offset: 0x18

Reset value: 0x0000 0000

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

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
GUAT[7:0]								PSC[7:0]							
rw								rw							

Bits	Fields	Descriptions
31:16	Reserved	Must be kept the reset value.
15:8	GUAT[7:0]	<p>Guard time value in Smartcard mode</p> <p>TC flag assertion time is delayed by GUAT[7:0] baud clock cycles.</p> <p>These bits are not available for UART3/4.</p>
7:0	PSC[7:0]	<p>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.</p> <p>00000000: Reserved - never program this value.</p> <p>00000001: divides by 1</p> <p>00000010: divides by 2</p> <p>...</p> <p>11111111: divides by 255</p> <p>When the USART works in IrDA normal mode, these bits must be set to 00000001.</p> <p>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.</p> <p>00000: Reserved - never program this value.</p> <p>00001: divides by 2</p> <p>00010: divides by 4</p> <p>...</p> <p>11111: divides by 62</p> <p>The PSC [7:5] bits are reserved in smartcard mode.</p>

### 17.4.8. Control register 3 (USART\_CTL3)

Address offset: 0x80

Reset value: 0x0000 0000

This register is not available for UART3/4.

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	MSBF	DINV	TINV	RINV	Reserved	EBIE	RTIE	SCRTNUM[2:0]	RTEN						

rw      rw      rw      rw      rw      rw      rw      rw      rw      rw

Bits	Fields	Descriptions
31:12	Reserved	Must be kept the reset value.
11	MSBF	<p>Most significant bit first</p> <p>This bit specifies the sequence of the data bits in transmission and reception.</p> <p>0: data is transmitted/received with the LSB first.</p> <p>1: data is transmitted/received with the MSB first.</p> <p>This bit field cannot be written when the USART is enabled (UEN=1).</p>
10	DINV	<p>Data bit level inversion.</p> <p>This bit specifies the polarity of the data bits in transmission and reception.</p> <p>0: Data bit signal values are not inverted.</p> <p>1: Data bit signal values are inverted.</p> <p>This bit field cannot be written when the USART is enabled (UEN=1).</p>
9	TINV	<p>TX pin level inversion</p> <p>This bit specifies the polarity of the TX pin.</p> <p>0: TX pin signal values are not inverted.</p> <p>1: TX pin signal values are inverted.</p> <p>This bit field cannot be written when the USART is enabled (UEN=1).</p>
8	RINV	<p>RX pin level inversion</p> <p>This bit specifies the polarity of the RX pin.</p> <p>0: RX pin signal values are not inverted.</p> <p>1: RX pin signal values are inverted.</p> <p>This bit field cannot be written when the USART is enabled (UEN=1).</p>
7:6	Reserved	Must be kept the reset value.
5	EBIE	<p>Interrupt enable bit of end of block event.</p> <p>If this bit is set, an interrupt occurs when the EBF bit in USART_STAT1 is set.</p> <p>0: End of block interrupt is enabled.</p> <p>1: End of block interrupt is disabled.</p>

---

4	RTIE	Interrupt enable bit of receive timeout event. If this bit is set, an interrupt occurs when the RTF bit in USART_STAT1 is set. 0: Receive timeout interrupt is enabled. 1: Receive timeout interrupt is disabled.
3:1	SCRTNUM[2:0]	Smartcard auto-retry number. In Smartcard mode, these bits specify the number of retries in transmission and reception. In transmission mode, a frame can be retransmitted by SCRTNUM times. If the frame is NACKed by (SCRTNUM+1) times, the FERR is set. In reception mode, a frame reception can be tried by (SCRTNUM+1) times. If the parity bit mismatch event occurs (SCRTNUM+1) times for a frame, the RBNE and PERR bits are set. When these bits are configured as 0x0, there will be no automatic retransmission in transmit mode.
0	RTEN	Receiver timeout enable. This bit enables the receive timeout counter of the USART. 0: Receiver timeout function disabled. 1: Receiver timeout function enabled.

### 17.4.9. Receiver timeout register (USART\_RT)

Address offset: 0x84

Reset value: 0x0000 0000

This register is not available for UART3/4.

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
BL[7:0]								RT[23:16]							
rw								rw							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RT[15:0]								rw							

Bits	Fields	Descriptions
31:24	BL[7:0]	<p>Block Length</p> <p>These bits specify the block length in Smartcard T=1 Reception. Its value equals to the number of information characters + the length of the Epilogue Field (1-LEC/2-CRC) - 1.</p> <p>This value, which must be programmed only once per received block, can be programmed after the start of the block reception (using the data from the LEN character in the Prologue Field). The block length counter is reset when TBE=0 in Smartcard mode.</p> <p>In other modes, when REN=0 (receiver disabled), or when the EBF bit of</p>

USART\_STAT1 is written to 0, the Block length counter is reset.

23:0	RT[23:0]	<p>Receiver timeout threshold.</p> <p>These bits are used to specify receiver timeout value in terms of number of baud clocks.</p> <p>If Smartcard mode is not enabled, the RTF bit of USART_STAT1 is set if no new start bit is detected longer than RT bits time after the last received character.</p> <p>If Smartcard mode is enabled, the CWT and BWT are implemented by this value.</p> <p>In this case, the timeout measurement is started from the start bit of the last received character.</p> <p>These bits can be written on the fly. The RTF flag will be set if the new value is lower than or equal to the internal timeout counter. These bits must only be programmed once per received character.</p>
------	----------	---

#### 17.4.10. Status register 1 (USART\_STAT1)

Address offset: 0x88

Reset value: 0x0000 0000

This register is not available for UART3/4.

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															BSY
															r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		EBF	RTF	Reserved											
		w0	w0												

Bits	Fields	Descriptions
31:17	Reserved	Must be kept the reset value.
16	BSY	<p>Busy flag</p> <p>This bit is set when the USART is receiving a data frame.</p> <p>0: USART reception path is idle.</p> <p>1: USART reception path is working.</p>
15:13	Reserved	Must be kept the reset value.
12	EBF	<p>End of block flag</p> <p>This bit is set when the number of received bytes (from the start of the block, including the prologue) is equal or greater than BLEN + 4. An interrupt occurs if the EBIE bit in USART_CTL3 is set.</p> <p>Software can clear this bit by writing 0 to it.</p> <p>0: End of Block event does not occur.</p> <p>1: End of Block event has occurred.</p>

11	RTF	Receiver timeout flag.  This bit is set when the RX pin is in idle state for longer than RT bits time. An interrupt occurs if the RTIE bit in USART_CTL3 is set. Software can clear this bit by writing 0 to it. 0: Receiver timeout event does not occur. 1: Receiver timeout event has occurred.
10:0	Reserved	Must be kept the reset value.

## 18. Inter-integrated circuit interface (I2C)

### 18.1. Overview

The I2C (inter-integrated circuit) module provides an I2C interface which is an industry standard two-line serial interface for MCU to communicate with external I2C interface. I2C bus uses two serial lines: a serial data line, SDA, and a serial clock line, SCL.

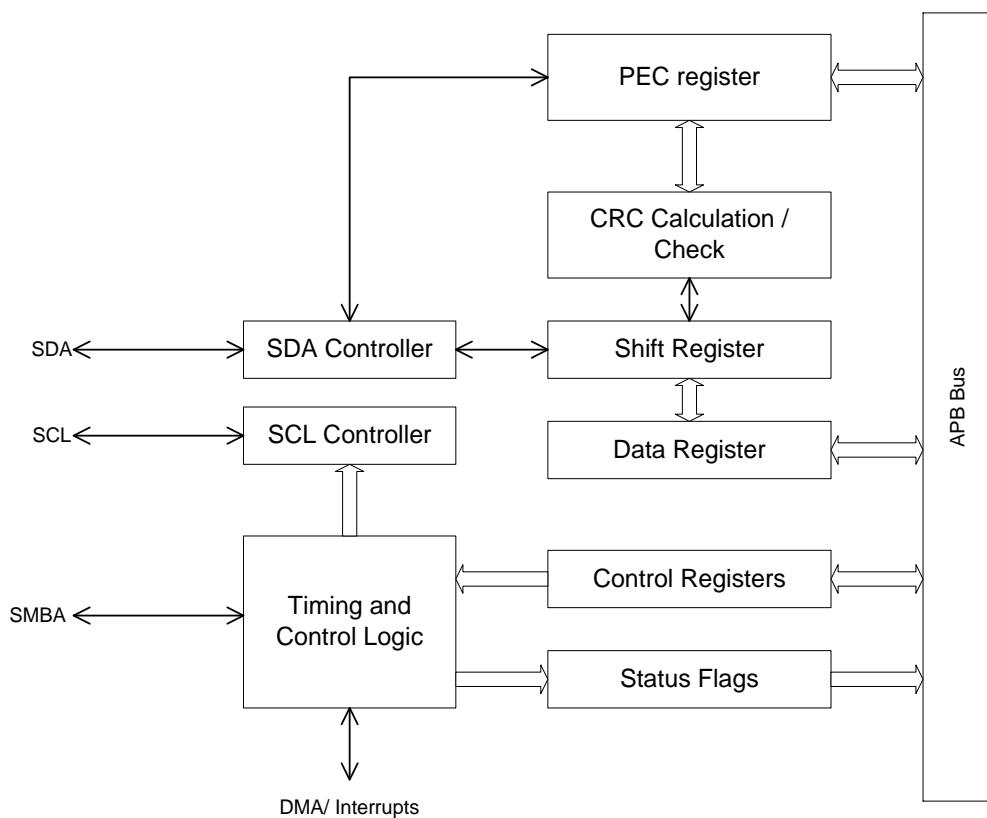
The I2C interface implements standard I2C protocol with standard-mode, fast-mode and fast-mode-plus as well as CRC calculation and checking, SMBus (system management bus) and PMBus (power management bus). It also supports multi-master I2C bus. The I2C interface provides DMA mode for users to reduce CPU overload.

### 18.2. Characteristics

- Parallel-bus to I2C-bus protocol conversion 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), fast-mode (up to 400 kHz) and fast-mode-plus (up to 1MHz).
- 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.

### 18.3. Function overview

[\*\*Figure 18-1. I2C module block diagram\*\*](#) below provides details of the internal configuration of the I2C interface.

**Figure 18-1. I2C module block diagram**

**Table 18-1. Definition of I2C-bus terminology (refer to the I2C specification of Philips semiconductors)**

Term	Description
Transmitter	The device which sends data to the bus
Receiver	The device which receives data from the bus
Master	The device which initiates a transfer, generates clock signals and terminates a transfer
Slave	The device addressed by a master
Multi-master	More than one master can attempt to control the bus at the same time without corrupting the message
Synchronization	Procedure to synchronize the clock signals of two or more devices
Arbitration	Procedure to ensure that, if more than one master tries to control the bus simultaneously, only one is allowed to do so and the winning master's message is not corrupted

### 18.3.1. SDA and SCL lines

The I2C module has two external lines, the serial data SDA and serial clock SCL lines. The two wires carry information between the devices connected to the bus.

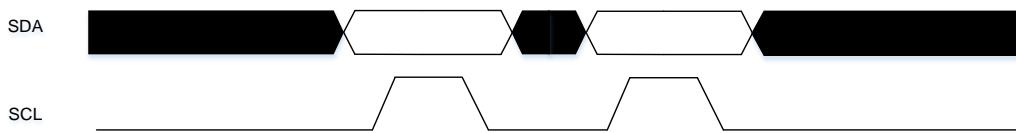
Both SDA and SCL are bidirectional lines, connected to a positive supply voltage via current-source or pull-up resistor. When the bus is free, both lines are HIGH. The output stages of

devices connected to the bus must have an open-drain or open-collect to perform the wired-AND function. Data on the I2C-bus can be transferred at rates of up to 100 Kbit/s in the standard-mode, up to 400 Kbit/s in the fast-mode and up to 1Mbit/s in the fast-mode-plus if the FMPEN bit in I2C\_FMPCFG is set. 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>.

### 18.3.2. Data validation

The data on the SDA line must be stable during the HIGH period of the clock. The HIGH or LOW state of the SDA line can only change when the clock signal on the SCL line is LOW (see [Figure 18-2. Data validation](#)). One clock pulse is generated for each data bit transferred.

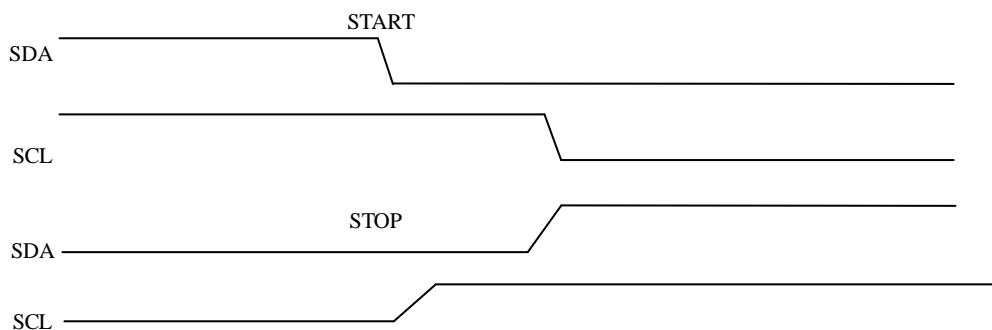
**Figure 18-2. Data validation**



### 18.3.3. START and STOP signal

All transmissions begin with a START and are terminated by a STOP (see [Figure 18-3. START and STOP condition](#)). A HIGH to LOW transition on the SDA line while SCL is HIGH defines a START signal. A LOW to HIGH transition on the SDA line while SCL is HIGH defines a STOP signal.

**Figure 18-3. START and STOP condition**



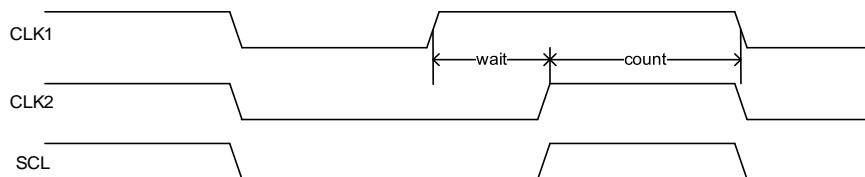
### 18.3.4. Clock synchronization

Two masters can begin transmitting on a free bus at the same time and there must be a method for deciding which master takes control of the bus and completes 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 their LOW period and, once a master clock has gone LOW, it holds the SCL line in that state until the clock HIGH state is reached (see [Figure 18-4. Clock synchronization](#)). However, if another clock is still within its LOW period, the LOW to HIGH transition of this clock may not change the state of the SCL line. The SCL line is therefore held LOW by the master with the longest LOW period. Masters with shorter LOW period enter a HIGH wait-state during this time.

**Figure 18-4. Clock synchronization**



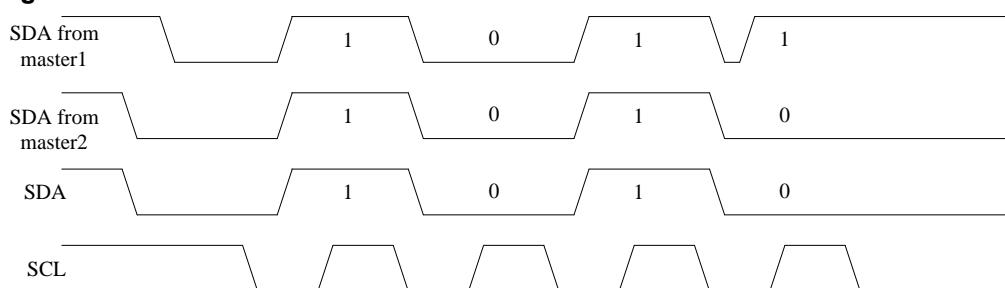
### 18.3.5. Arbitration

Arbitration, like synchronization, is part of the protocol where more than one master is used in the system. Slaves are not involved in the arbitration procedure.

A master may start a transfer only if the bus is free. Two masters may generate a START signal within the minimum hold time of the START signal which results in a valid START signal 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 whether the SDA level matches what it has been sent. This process may take many bits. Two masters can even complete an entire transmission 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 transmission.

**Figure 18-5. SDA line arbitration**



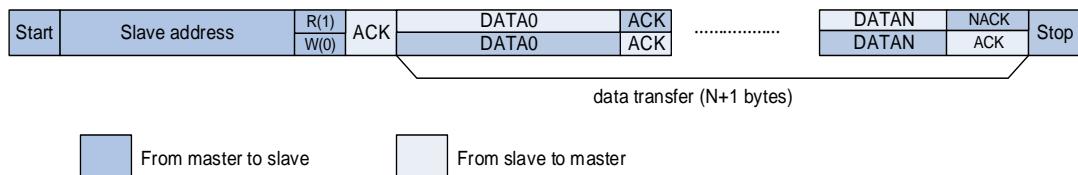
### 18.3.6. I2C communication flow

Each I2C device is recognized by a unique address (whether it is a microcontroller, LCD driver, memory or keyboard interface) and can be operated as either a transmitter or receiver, depending on the function of the device.

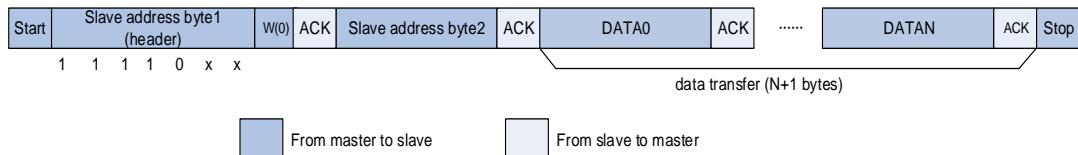
An I2C slave will continue to detect addresses after a START signal on I2C bus and compare the detected address with its slave address which is programmed by software. Once the two addresses match with each other, the I2C slave will send an ACK to the I2C bus and respond to the following command on I2C bus: transmitting or receiving the desired data. Additionally, if General Call is enabled by software, the I2C slave always responds to a General Call Address (0x00). The I2C block supports both 7-bit and 10-bit address modes.

An I2C master always initiates or ends a transfer using START or STOP signal and it's also responsible for SCL clock generation.

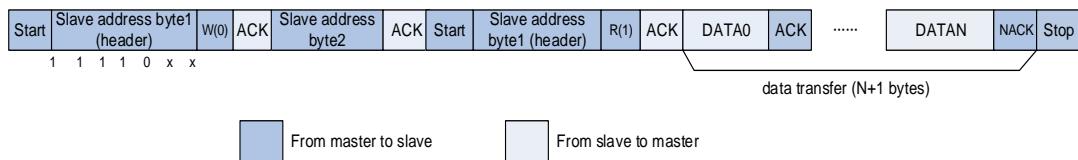
**Figure 18-6. I2C communication flow with 7-bit address**



**Figure 18-7. I2C communication flow with 10-bit address (Master Transmit)**



**Figure 18-8. I2C communication flow with 10-bit address (Master Receive)**



### 18.3.7. Programming model

An I2C device such as LCD driver may only be a receiver, whereas a memory can both receive and transmit data. In addition to transmitters and receivers, devices can also be considered as masters or slaves when performing data transfers. A master is the device which initiates a data transfer on the bus and generates the clock signals to permit that transfer. At that time, any device addressed is considered 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.

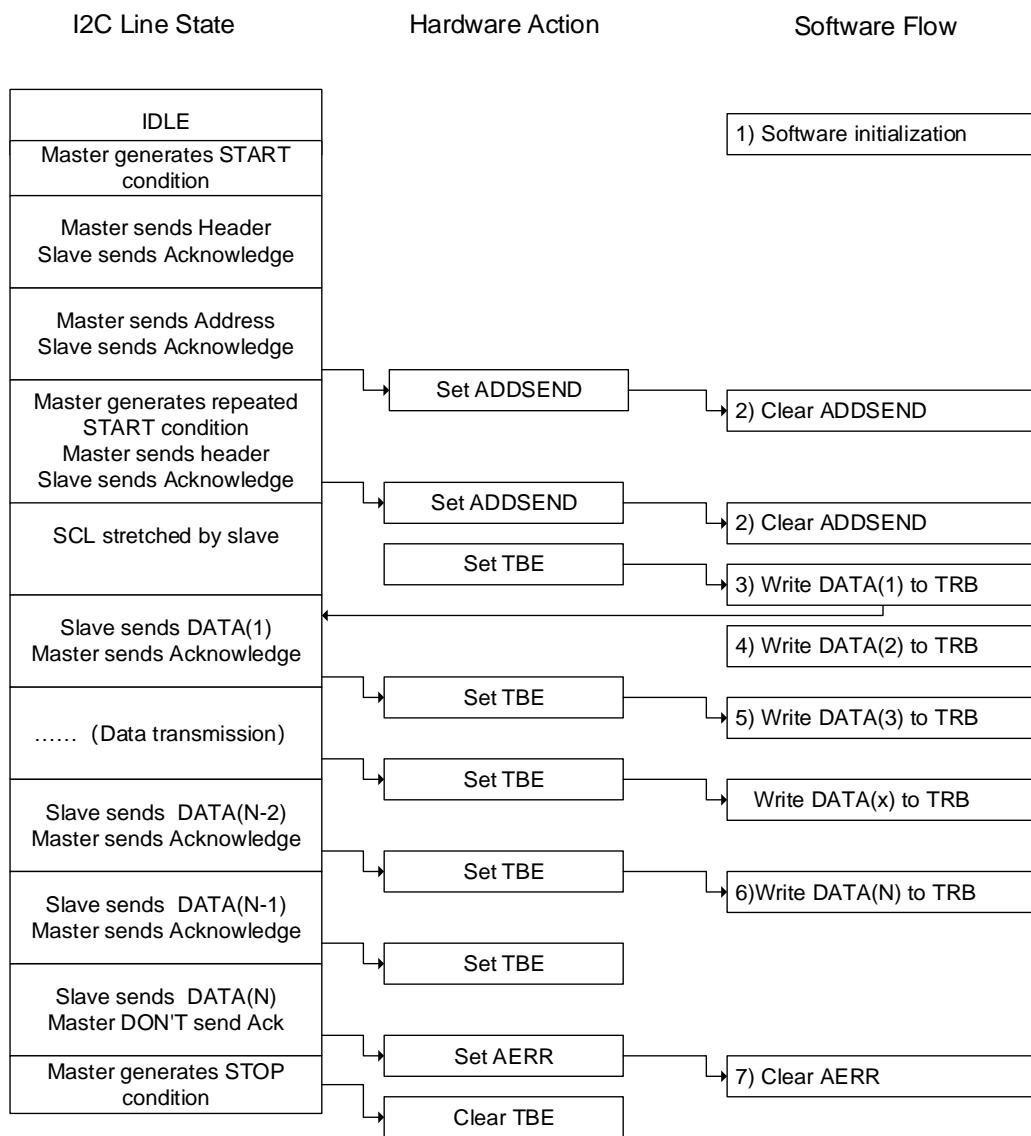
I2C block supports all of the four I2C modes. After system reset, it works in slave mode. After sending a START signal on I2C bus, it changes into master mode. The I2C changes back to slave mode after sending a STOP signal on I2C bus.

### Programming model in slave transmitting mode

As is shown in [Figure 18-9. Programming model for slave transmitting mode \(10-bit address mode\)](#), the following software procedure should be followed if users wish to transmit data in slave transmitter mode:

1. First of all, 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 signal followed by address on I2C bus.
2. After receiving a START signal 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 signal followed by a header to the I2C bus. The slave sets ADDSEND bit again after it detects the repeated START signal and the following header. The ADDSEND bit must be cleared by software 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 byte written 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 transmission of the first byte, 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. After the transmission of the first byte, the TBE bit will be set, the software can write the third byte to the I2C\_DATA register and TBE is cleared. After this, any time TBE is set, software can write a byte to I2C\_DATA as long as there is still data to be transmitted.
6. During the transmission of the second last byte, software writes 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 signal.

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 signal on I2C bus and sets AERR (Acknowledge Error) bit to notify software that the transmission completes. Software clears AERR bit by writing 0 to it.

**Figure 18-9. Programming model for slave transmitting mode (10-bit address mode)**


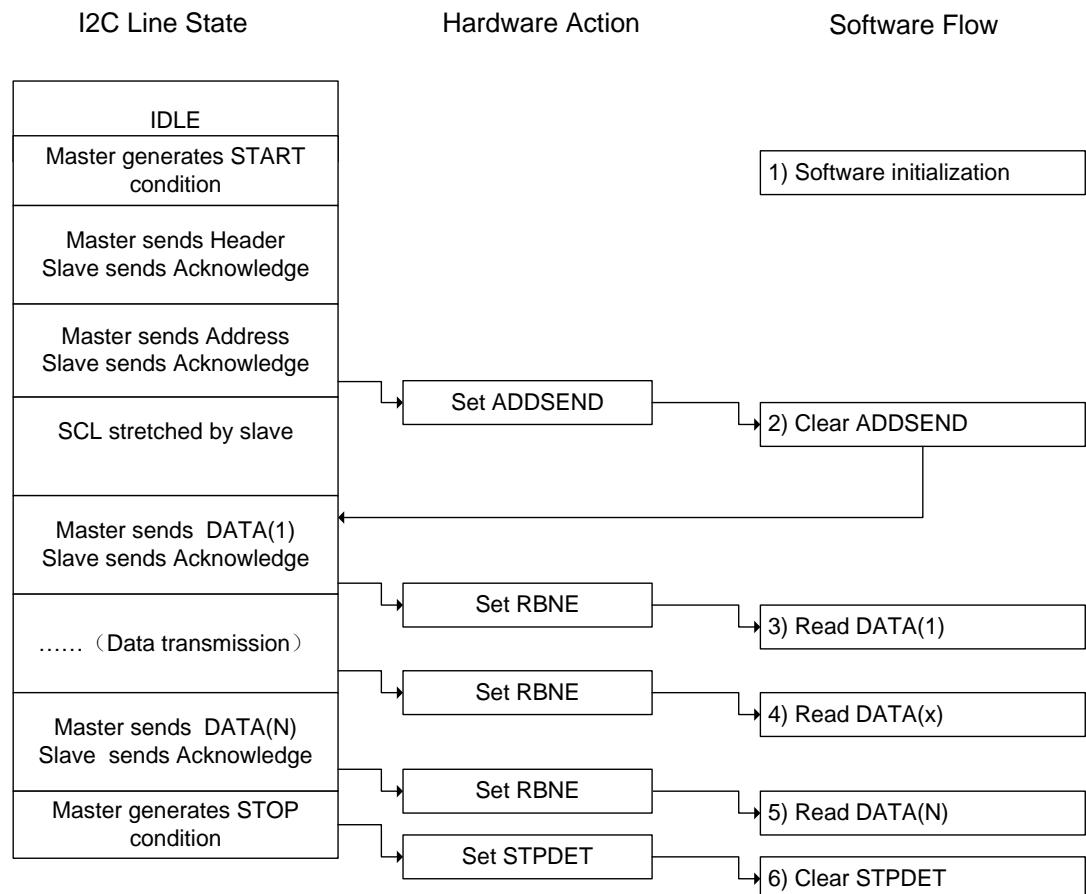
### Programming model in slave receiving mode

As is shown in [\*\*Figure 18-10. Programming model for slave receiving \(10-bit address mode\)\*\*](#), the following software procedure should be followed if users wish to receive data in slave receiver mode:

1. First of all, 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 signal followed by address on I2C bus.
2. After receiving a START signal followed by a matched 7-bit or 10-bit address, the I2C hardware sets the ADDSEND bit in I2C status register 0, 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 on 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 the last byte is received, RBNE is set. Software reads the last byte.
6. STPDET bit is set when I2C detects a STOP signal on I2C bus and software reads I2C\_STAT0 and then writes I2C\_CTL0 to clear the STPDET bit.

**Figure 18-10. Programming model for slave receiving (10-bit address mode)**



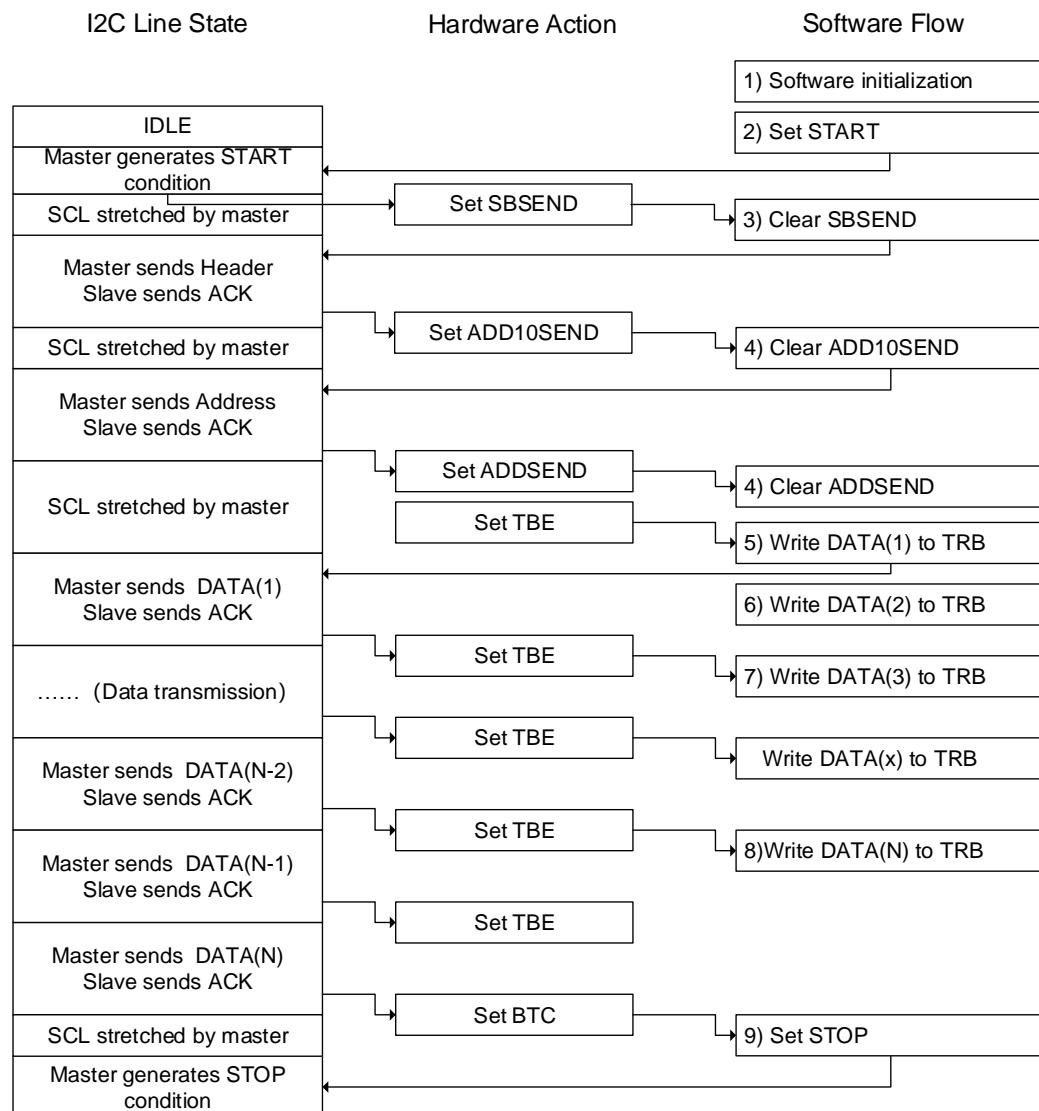
### Programming model in master transmitting mode

As it shows in [\*\*Figure 18-11. Programming model for master transmitting mode \(10-bit address mode\)\*\*](#), the following software procedure should be followed if users wish to make transaction in master transmitter mode:

1. First of all, 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 signal followed by address on I2C bus.
2. Software sets START bit requesting I2C to generate a START signal on I2C bus.
3. After sending a START signal, the I2C hardware sets the SBSEND bit in I2C\_STAT0 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 which has been sent is header of a 10-bit address, the hardware sets ADD10SEND bit after sending the 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 has been 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 writes the first byte data to I2C\_DATA register, but the TBE will not be cleared because the byte written in I2C\_DATA is moved to internal shift register immediately. The I2C begins to transmit data to I2C bus as soon as the shift register is not empty.
6. During the transmission of the first byte, 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 is still data to be transmitted.
8. During the transmission of the second last byte, software writes the last data to I2C\_DATA to clear the TBE flag and doesn't care TBE anymore. So TBE will be asserted after the transmission of the byte and not be cleared until a STOP signal.
9. After sending the last byte, I2C master sets BTC bit because both the shift register and I2C\_DATA are empty. Software should set the STOP bit to generate a STOP signal, then the I2C clears both TBE and BTC flags.

**Figure 18-11. Programming model for master transmitting mode (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 a 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 applications: Solution A and B. Solution A requires the software's quick response to I2C events, while Solution B doesn't.

#### **Solution A**

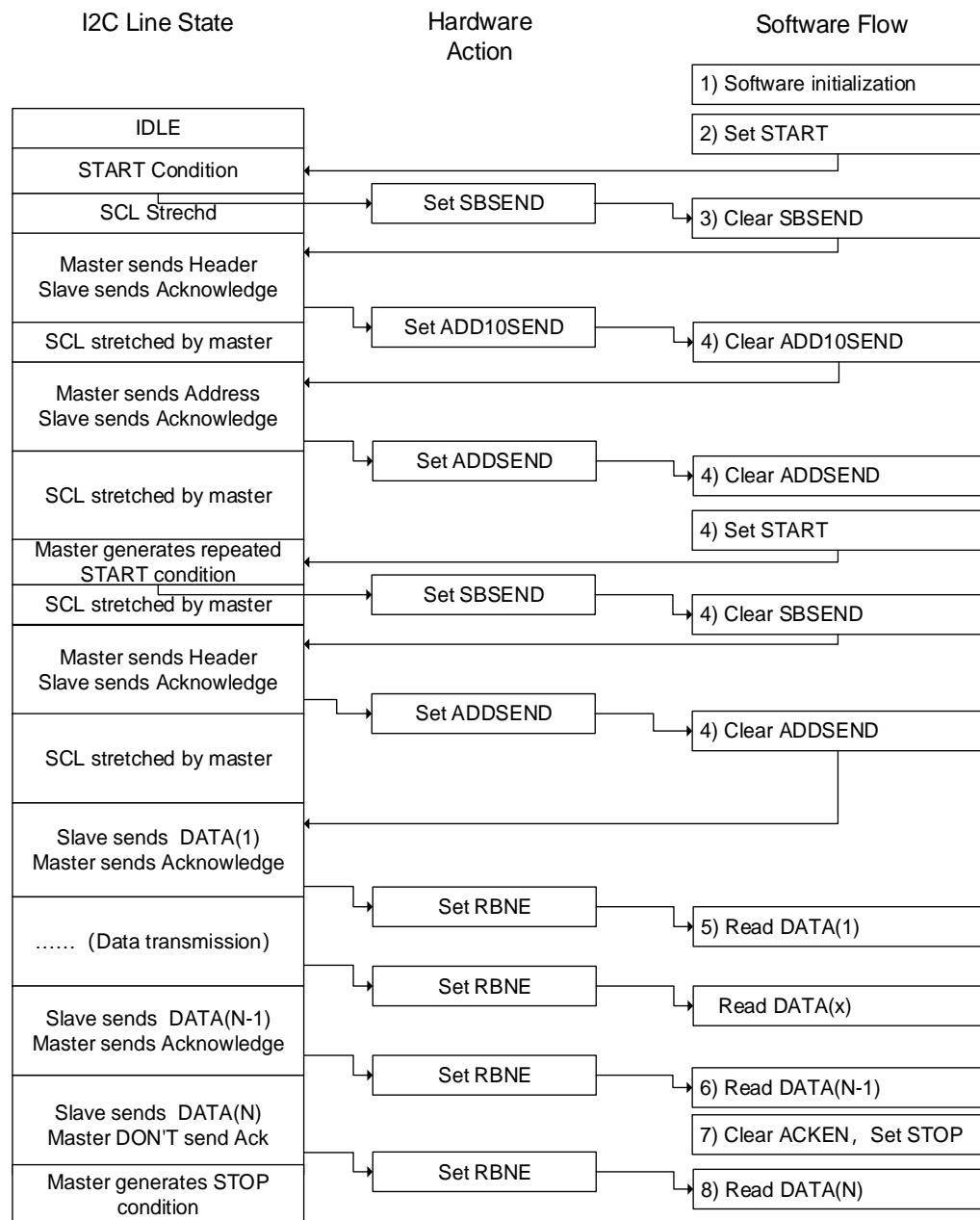
1. First of all, enable I2C peripheral clock as well as configure clock related registers in I2C\_CTL1 to make sure correct I2C timing. After enabled and configured, I2C operates in its default slave state and waits for START signal followed by address on I2C bus.
2. Software sets START bit requesting I2C to generate a START signal on I2C bus.
3. After sending a START signal, the I2C hardware sets the SBSEND bit in I2C\_STAT0

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 which has been sent is header of a 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 has been 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 signal 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 (N-1) 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 will be sent for the last byte.
8. After the last byte is received, RBNE is set. Software reads the last byte. Since ACKEN has been cleared in the previous step, I2C doesn't send ACK for the last byte and it generates a STOP signal after the transmission of the last byte.

The above steps require byte number  $N > 1$ . If  $N = 1$ , Step 7 should be performed after Step 4 and completed before the end of the single byte's receiving.

**Figure 18-12. Programming model for master receiving using Solution A (10-bit**

**address mode)**

**Solution B**

1. First of all, 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 signal followed by address on I2C bus.
2. Software sets START bit requesting I2C to generate a START signal on I2C bus.
3. After sending a START signal, the I2C hardware sets the SBSEND bit in I2C\_STAT0 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 which has been 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 has been 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 signal on I2C bus and SBSEND is set after the repeated START is sent out. Software should clear the SBSEND bit by reading I2C\_STAT0 and writing header to I2C\_DATA. Then the header is sent out to I2C bus, and ADDSEND is set again. Software should again clear ADDSEND by reading I2C\_STAT0 and then I2C\_STAT1.
5. As soon as the first byte is received, RBNE is set by hardware. Software now can read the first byte from I2C\_DATA and RBNE is cleared as well.
6. Any time RBNE is set, software can read a byte from I2C\_DATA until the master receives N-3 bytes.

As shown in [Figure 18-13. Programming model for master receiving mode 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. Master doesn't send an ACK for the last byte because ACKEN is already cleared.
8. After the last byte is received, both BTC and RBNE are set again, and SCL is stretched low. Software sets STOP bit and master sends out a STOP signal 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.

The above steps require that byte number N>2. N=1 and N=2 are similar:

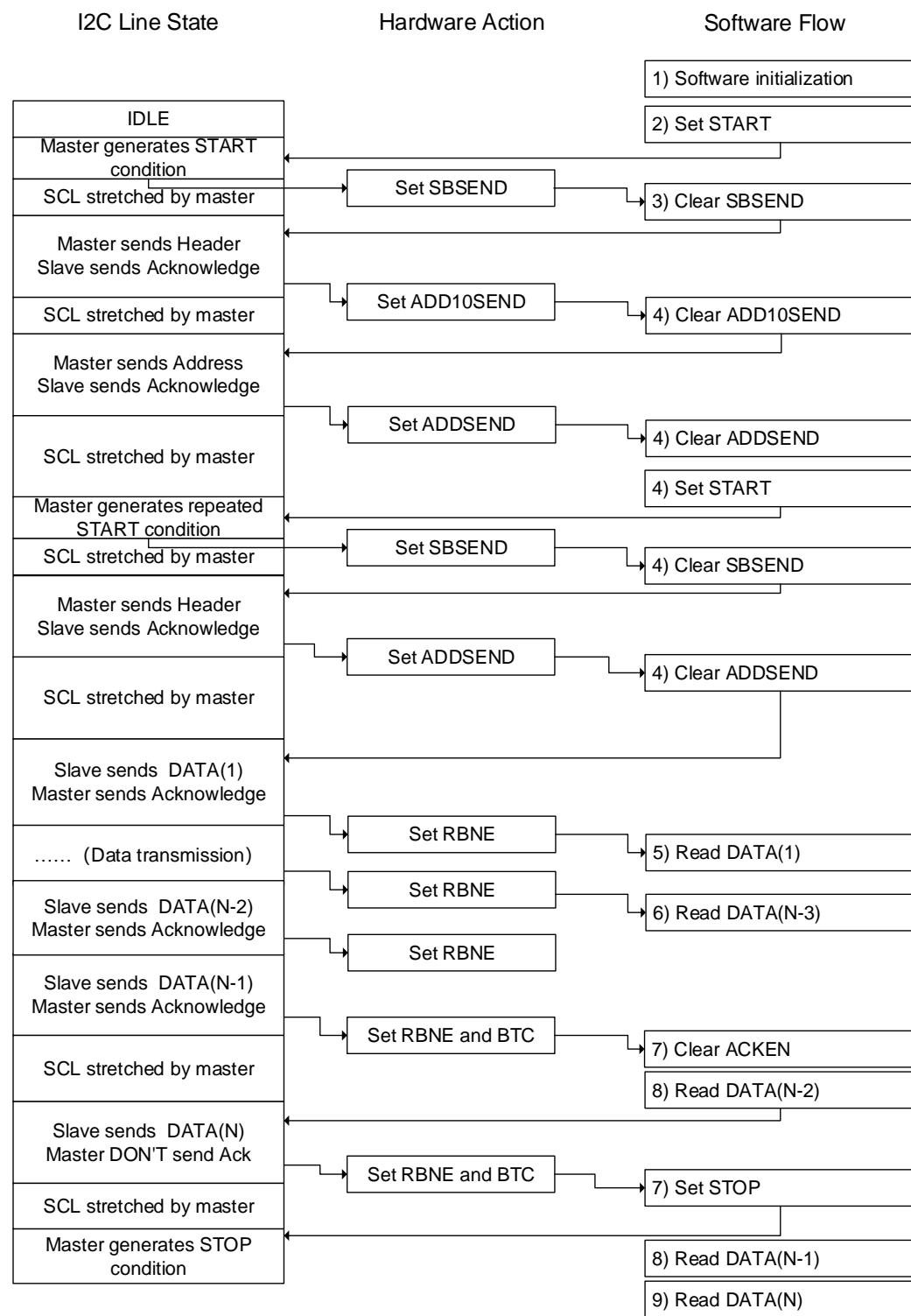
### **N=1**

In Step4, software should reset ACKEN 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 setting 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 read I2C\_DATA twice.

**Figure 18-13. Programming model for master receiving mode using solution B (10-bit**

**address mode)**


### 18.3.8. SCL line stretching

The SCL line stretching function is designed to avoid overflow error in reception and underflow error in transmission. As is shown in Programming Model, when the TBE and BTC bits are set in transmitting mode, the transmitter stretches the SCL line low until the transfer buffer

register is filled with the next data to be transmitted. When the RBNE and BTC bits are set in receiving mode, 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.

### 18.3.9. Use DMA for data transfer

As is shown in Programming Model, each time TBE or RBNE is asserted, software should write or read a byte, this may cause CPU to be high overloaded. The DMA controller can be used to process TBE and RBNE flags: each time TBE or RBNE is asserted, DMA controller does a read or write operation automatically. It reduces the load on the CPU. See the DMA section for details on how to configure DMA.

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 the I2C transfer. When the configured number of bytes have been transferred, the DMA controller generates End of Transfer (EOT) interrupt. DMA will send an End of Transmission (EOT) signal to the I2C interface and generates a DMA full transfer finish interrupt.

When a master receives two or more bytes, the DMALST bit in the I2C\_CTL1 register should be set. The I2C master will send NACK after the last byte. The STOP bit can be set by software to generate a STOP signal in the ISR of the DMA full transfer finish 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 signal after clearing the ADDSEND status, or in the ISR of the DMA full transfer finish interrupt.

### 18.3.10. Packet error checking

There is a CRC-8 calculator in I2C block to perform PEC (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 are set.

### 18.3.11. SMBus support

The System Management Bus (abbreviated to SMBus or SMB) is a single-ended simple two-wire bus for the purpose of lightweight communication. Most commonly it is found in computer motherboards for communication with power source for ON/OFF instructions. It is derived from I2C for communication with low-bandwidth devices on a motherboard, especially power related chips such as a laptop's rechargeable battery subsystem (see Smart Battery Data).

#### SMBus protocol

Each message transmission 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 is realized based on I2C hardware and it uses I2C hardware addressing, but it adds the 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 allows 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 this protocol there is a very useful distinction between a system host and all the other devices in the system, that is the host provides address assignment function.

#### Time-out feature

SMBus has a time-out feature which resets devices if a communication takes too long. This explains the minimum clock frequency is 10 kHz to prevent locking up the bus. I2C can be a 'DC' bus, which means that a slave device stretches the master clock when performing some routines while the master is accessing it. This will notify the master that the slave is busy but does not want to lose the communication. The slave device will continue the communication after its task is completed. There is no limit in the I2C bus protocol of 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 solve the problem. 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 byte is appended at the end of each transaction. The byte is a CRC-8 checksum of 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 which is based on the I2C multi-master mode but it can pass more data.

### 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, respond 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 values.
2. In order to support address resolution protocol (ARP) (ARPEN=1), the software should respond 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 respond to SMBALT flag and implement the related function.

#### 18.3.12. Status, errors and interrupts

There are several status and error flags in I2C, and interrupts may be asserted from these flags by setting some register bits (refer to [Register definition](#) for detail).

**Table18-2. Event status flags**

Event Flag Name	Description
SBSEND	START signal sent (master)
ADDSEND	Address sent or received
ADD10SEND	Header of 10-bit address sent
STPDET	STOP signal detected
BTC	Byte transmission completed
TBE	I2C_DATA is empty when transmitting
RBNE	I2C_DATA is not empty when receiving

**Table18-3. I2C error flags**

Error Name	Description
BERR	Bus error
LOSTARB	Arbitration lost
OUERR	Over-run or under-run when SCL stretch is disabled.

Error Name	Description
AERR	No acknowledge received
PECERR	CRC value doesn't match
SMBTO	Bus timeout in SMBus mode
SMBALT	SMBus Alert

## 18.4. Register definition

I2C0 base address: 0x4000 5400

I2C1 base address: 0x4000 5800

### 18.4.1. Control register 0 (I2C\_CTL0)

Address offset: 0x00

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
SRESET	Reserved	SALT	PECTRA NS	POAP	ACKEN	STOP	START	SS	GCEN	PECEN	ARPEN	SMBSEL	Reserved	SMBEN	I2CEN

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15	SRESET	Software resets I2C, software should wait until the I2C lines are released to reset the I2C  0: I2C is not reset 1: I2C is reset
14	Reserved	Must be kept at 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 sets and clears this bit while hardware clears this bit when PEC is transferred or START/STOP condition is detected I2CEN=0  0: Don't transfer PEC value 1: Transfer PEC value
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	ACK enable  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 is 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 a START condition is detected or I2CEN=0. 0: START will not be sent 1: START will be sent
7	DISSTRC	SCL stretching  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 respond to a General Call 1: Slave will respond to a General Call
5	PECEN	PEC calculation enable 0: PEC calculation disable 1: PEC calculation enable
4	ARPEN	ARP protocol enable 0: ARP is disabled 1: ARP is enabled
3	SMBSEL	SMBus type selection 0: Device 1: Host
2	Reserved	Must be kept at 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
---	-------	--

### 18.4.2. Control register 1 (I2C\_CTL1)

Address offset: 0x04

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	DMALST	DMAON	BUFIE	EVIE	ERRIE	Reserved						I2CCLK[5:0]				rw

Bits	Fields	Descriptions
31:13	Reserved	Must be kept at reset value.
12	DMALST	DMA last transfer configure 0: Next DMA EOT is not the last transfer 1: Next DMA EOT is the last transfer
11	DMAON	DMA is mode switched on 0: DMA mode is switched off 1: DMA mode is switched on
10	BUFIE	0: Buffer interrupt is disabled 1: Buffer interrupt is enabled, which means that interrupt will be generated when TBE = 1 or RBNE = 1 if EVIE=1.
9	EVIE	Event interrupt enable 0: Event interrupt is disabled 1: Event interrupt is enabled, which 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 is disabled 1: Error interrupt is enabled, which means that interrupt will be generated when BERR, LOSTARB, AERR, OUERR, PECERR, SMBTO or SMBALT flag is asserted.

7	Reserved	Must be kept at reset value.
6:0	I2CCLK[6:0]	<p>I2C peripheral clock frequency</p> <p>I2CCLK[6:0]should be the frequency of input APB1 clock in MHz which is at least 2.</p> <p>0d – 1d: Not allowed</p> <p>2d – 60d: 2 MHz~60MHz</p> <p>61d – 127d: Not allowed due to the limitation of APB1 clock</p> <p><b>Note:</b></p> <p>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. In I2C fast mode plus, the frequencies of APB1 must be equal or greater than 24MHz.</p>

#### 18.4.3. Slave address register 0 (I2C\_SADDR0)

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
ADDFOR MAT	Reserved				ADDRESS[9:8]	ADDRESS[7:1]						ADDRES S0			

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:16	Reserved	Must be kept at reset value.
15	ADDFORMAT	Address format for the I2C slave 0: 7-bit address 1: 10-bit address
14:10	Reserved	Must be kept at reset value.
9:8	ADDRESS[9:8]	Highest two bits of a 10-bit address
7:1	ADDRESS[7:1]	7-bit address or bits 7:1 of a 10-bit address
0	ADDRESS0	Bit 0 of a 10-bit address

#### 18.4.4. Slave address register 1 (I<sub>2</sub>C\_SADDR1)

Address offset: 0x0C

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								ADDRESS2[7:1]				DUADEN			

rw rw

Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value.
7:1	ADDRESS2[7:1]	The second I2C address for the slave in Dual-Address mode
0	DUADEN	Dual-Address mode enable 0: Dual-Address mode is disabled 1: Dual-Address mode is enabled

#### 18.4.5. Transfer buffer register (I2C\_DATA)

Address offset: 0x10

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								TRB[7:0]							

rw

Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value.
7:0	TRB[7:0]	Transmission or reception data buffer

#### 18.4.6. Transfer status register 0 (I2C\_STAT0)

Address offset: 0x14

Reset value: 0x0000 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
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	r	r	r	r	r	r	r	r	r

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15	SMBALT	<p>SMBus Alert status</p> <p>This bit is set by hardware and cleared by writing 0.</p> <p>0: SMBA pin not pulled down (device mode) or no Alert detected (host mode)</p> <p>1: SMBA pin pulled down and Alert address received (device mode) or Alert detected (host mode)</p>
14	SMBTO	<p>Timeout signal in SMBus mode</p> <p>This bit is set by hardware and cleared by writing 0.</p> <p>0: No timeout error</p> <p>1: Timeout event occurs (SCL is low for 25 ms)</p>
13	Reserved	Must keep at reset value.
12	PECERR	<p>PEC error when receiving data</p> <p>This bit is set by hardware and cleared by writing 0.</p> <p>0: Received PEC matches calculated PEC</p> <p>1: Received PEC doesn't match the calculated PEC, I2C will send NACK careless of ACKEN bit.</p>
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.</p> <p>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</p> <p>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</p> <p>1: Arbitration lost occurs and the I2C block changes back to slave mode.</p>
8	BERR	<p>Bus error</p> <p>A bus error occurs when an unexpected START or STOP signal on I2C bus.</p> <p>This bit is set by hardware and cleared by writing 0.</p>

		0: No bus error 1: A bus error detected
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 will be 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 at reset value.
4	STPDET	<p>STOP signal is 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 signal not detected in slave mode 1: STOP signal detected in slave mode</p>
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 is sent in master mode 1: Header of 10-bit address is sent in master mode</p>
2	BTC	<p>Byte transmission is 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 and cleared by 3 ways as follow:</p> <ol style="list-style-type: none"> <li>1. Software clearing: reading I2C_STAT0 followed by reading or writing I2C_DATA</li> <li>2. Hardware clearing: sending the STOP signal or START signal</li> <li>3. Bit 0 (I2CEN bit) of the I2C_CTL0 is reset.</li> </ol> <p>0: BTC not asserted 1: BTC asserted</p>
1	ADDSEND	<p>Address is sent and ACK is received in master mode or address is received and matches with its own address in slave mode.</p> <p>This bit is set by hardware and cleared by reading I2C_STAT0 and reading I2C_STAT1.</p>

0: In slave mode, no address is received or the received address does not match with its own address. In master mode, no address is sent or address has been sent but not received the ACK from slave.

1: In slave mode, address is received and matches with its own address. In master mode, address has been sent and receives the ACK from slave.

0	SBSEND	START signal is sent out in master mode  This bit is set by hardware and cleared by reading I2C_STAT0 and writing I2C_DATA.  0: No START signal sent 1: START signal sent
---	--------	--

#### 18.4.7. Transfer status register 1 (I2C\_STAT1)

Address offset: 0x18

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
					PECV[7:0]			DUMODF	HSTSMB	DEFSMB	RXGC	Reserved	TR	I2CBSY	MASTER
					r			r	r	r	r		r	r	r

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:8	PECV[7:0]	Packet Error Checking value that calculated by hardware when PEC is enabled.
7	DUMODF	Dual flag in slave mode indicates which address matches with the address in Dual-Address mode  This bit is cleared by hardware after a STOP or a START condition or I2CEN=0 0: The address matches with SADDR0 address 1: The address matches with SADDR1 address
6	HSTSMB	SMBus host Header detected in slave mode  This bit is cleared by hardware after a STOP or a START signal or I2CEN=0 0: No SMBus host Header is detected 1: SMBus host Header is detected
5	DEFSMB	Default address of SMBus device  This bit is cleared by hardware after a STOP or a START signal 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 (0x00) received.

		This bit is cleared by hardware after a STOP or a START signal or I2CEN=0.
0	No general call address (0x00) received	
1	General call address (0x00) received	
3	Reserved	Must be kept at reset value.
2	TR	Transmitter or receiver  This bit indicates whether the I2C is a transmitter or a receiver. It is cleared by hardware after a STOP or a START signal or I2CEN=0 or LOSTARB=1. 0: Receiver 1: Transmitter
1	I2CBSY	Busy flag  This bit is cleared by hardware after a STOP signal 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 signal generates. This bit is cleared by hardware after a STOP signal or I2CEN=0 or LOSTARB=1. 0: Slave mode 1: Master mode

#### 18.4.8. Clock configure register (I2C\_CKCFG)

Address offset: 0x1C

Reset value: 0x0000 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
FAST	DTCY	Reserved										CLKC[11:0]			

rw      rw      rw

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15	FAST	I2C speed selection in master mode 0: Standard speed 1: Fast speed
14	DTCY	Duty cycle in fast mode 0: $T_{low}/T_{high}=2$ 1: $T_{low}/T_{high}=16/9$

13:12	Reserved	Must be kept the reset value
11:0	CLKC[11:0]	<p>I2C clock control in master mode</p> <p>In standard speed mode: <math>T_{high} = T_{low} = CLKC \cdot T_{PCLK1}</math></p> <p>In fast speed mode or fast mode plus, if DTCY=0:</p> $T_{high} = CLKC \cdot T_{PCLK1}$ , $T_{low} = 2 \cdot CLKC \cdot T_{PCLK1}$ <p>In fast speed mode or fast mode plus, if DTCY=1:</p> $T_{high} = 9 \cdot CLKC \cdot T_{PCLK1}$ , $T_{low} = 16 \cdot CLKC \cdot T_{PCLK1}$

**Note:** If DTCY is 0, when PCLK1 is an integral multiple of 3, the baud rate will be more accurate. If DTCY is 1, when PCLK1 is an integral multiple of 25, the baud rate will be more accurate.

#### 18.4.9. Rise time register (I2C\_RT)

Address offset: 0x20

Reset value: 0x0000 0002

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								RISETIME[6:0]							
rw															

Bits	Fields	Descriptions
31:7	Reserved	Must be kept at reset value.
6:0	RISETIME[6:0]	Maximum rise time in master mode The RISETIME value should be the maximum SCL rise time incremented by 1.

#### 18.4.10. Fast-mode-plus configure register (I2C\_FMPCFG)

Address offset: 0x90

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								FMPEN							
rw															

Bits	Fields	Descriptions
31:1	Reserved	Must be kept at reset value.
0	FMPEN	<p>Fast mode plus enable</p> <p>The I2C device supports up to 1MHz when this bit is set.</p> <p>0: Fast mode plus disabled</p> <p>1: Fast mode plus enabled</p>

## 19. Serial peripheral interface/Inter-IC sound (SPI/I2S)

### 19.1. Overview

The SPI/I2S module can communicate with external devices using the SPI protocol or the I2S audio protocol.

The Serial Peripheral Interface (SPI) provides a SPI protocol of data transmission and reception function in master or slave mode. Both full-duplex and simplex communication modes are supported, with hardware CRC calculation and checking. Quad-SPI master mode is also supported in SPI0.

The inter-IC sound (I2S) supports four audio standards: I2S Phillips standard, MSB justified standard, LSB justified standard, and PCM standard. I2S works at either master or slave mode for transmission and reception.

### 19.2. Characteristics

#### 19.2.1. SPI characteristics

- Master or slave operation with full-duplex or simplex mode.
- Separate transmit and receive buffer, 16 bits wide.
- Data frame size can be 8 or 16 bits.
- Bit order can be LSB first or MSB first.
- Software and hardware NSS management.
- Hardware CRC calculation, transmission and checking.
- Transmission and reception using DMA.
- SPI TI mode supported.
- SPI NSS pulse mode supported.
- Quad-SPI configuration available in master mode (only in SPI0).

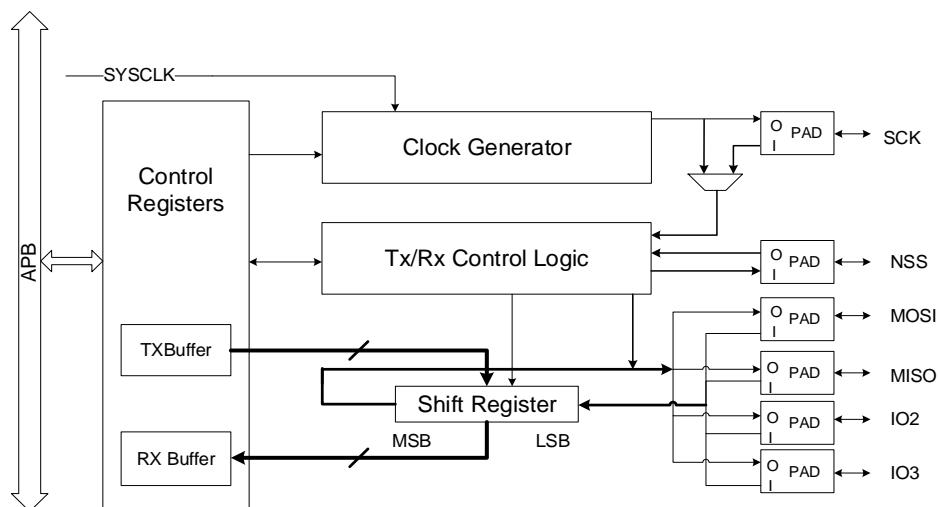
#### 19.2.2. I2S characteristics

- Master or slave operation with transmission or reception mode.
- Four I2S standards supported: Phillips, MSB justified, LSB justified and PCM standard.
- Data length can be 16 bits, 24 bits or 32 bits.
- Channel length can be 16 bits or 32 bits.
- Transmission and reception using a 16 bits wide buffer.
- Audio sample frequency can be 8 kHz to 192 kHz using I2S clock divider.
- Programmable idle state clock polarity.
- Master clock (MCK) can be output.
- Transmission and reception using DMA.

## 19.3. SPI function overview

### 19.3.1. SPI block diagram

Figure 19-1. Block diagram of SPI



### 19.3.2. SPI signal description

#### Normal configuration (Not Quad-SPI Mode)

Table 19-1. SPI signal description

Pin Name	Direction	Description
SCK	I / O	Master: SPI Clock Output Slave: SPI Clock Input
MISO	I / O	Master: Data reception line Slave: Data transmission line Master with Bidirectional mode: Not used Slave with Bidirectional mode: Data transmission and reception Line.
MOSI	I / O	Master: Data transmission line Slave: Data reception line Master with Bidirectional mode: Data transmission and reception Line. Slave with Bidirectional mode: Not used
NSS	I / O	Software NSS Mode: Not Used Master in hardware NSS mode: when NSSDRV=1, it is NSS output, suitable for single master application; when NSSDRV=0, it is NSS input, suitable for multi-master

application.

Slave in Hardware NSS Mode: NSS input, as a chip select signal for slave.

### Quad-SPI configuration

SPI is in single wire mode by default and enters into Quad-SPI mode after QMOD bit in SPI\_QCTL register is set (only available in SPI0). Quad-SPI mode can only work at master mode.

Software is able to drive IO2 and IO3 pins high in normal Non-Quad-SPI mode by using IO23\_DRV bit in SPI\_QCTL register.

The SPI is connected to external devices through 6 pins in Quad-SPI mode:

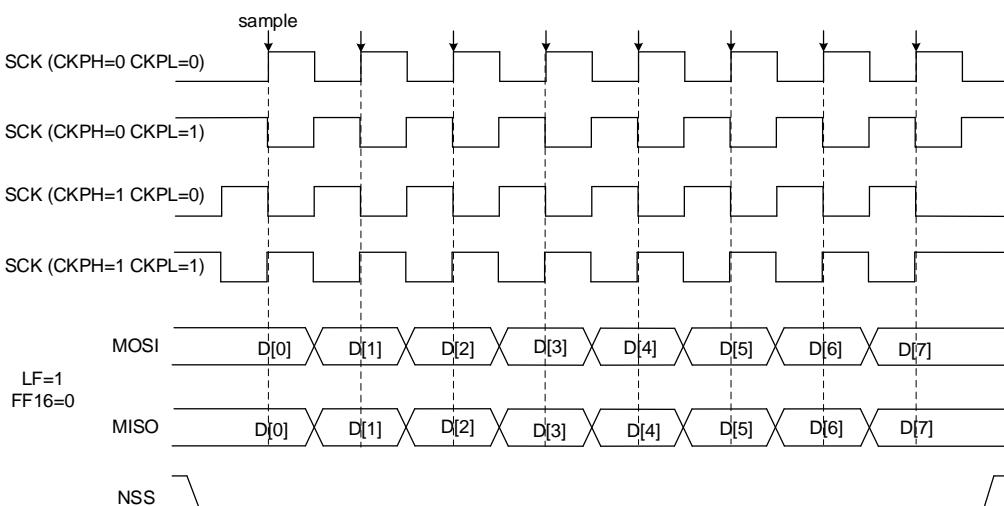
**Table 19-2. Quad-SPI signal description**

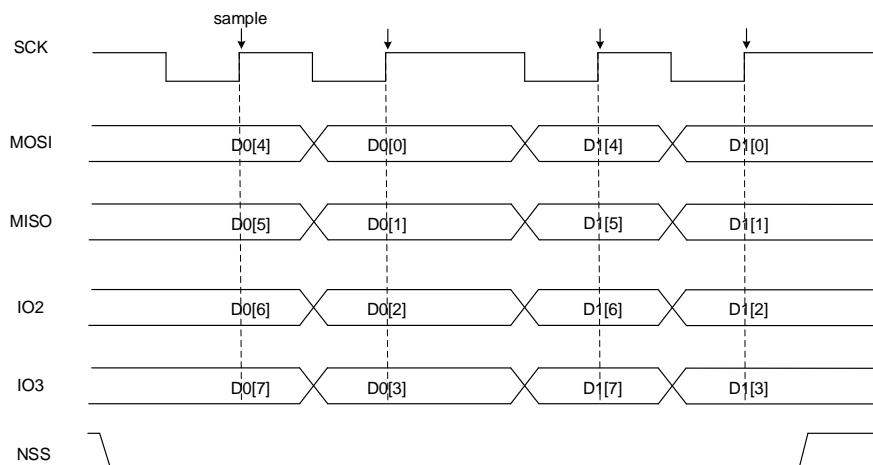
Pin Name	Direction	Description
SCK	O	SPI Clock Output
MOSI	I / O	Transmission or Reception Data 0 line
MISO	I / O	Transmission or Reception Data 1 line
IO2	I / O	Transmission or Reception Data 2 line
IO3	I / O	Transmission or Reception Data 3 line
NSS	O	NSS output

### 19.3.3. 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 19-2. SPI timing diagram in normal mode**



**Figure 19-3. SPI timing diagram in Quad-SPI mode (CKPL=1, CKPH=1, LF=0)**


In normal mode, the length of data is configured by the FF16 bit in the SPI\_CTL0 register. Data length is 16 bits if FF16=1, otherwise is 8 bits. The data frame length is fixed to 8 bits in Quad-SPI mode.

Data order is configured by LF bit in SPI\_CTL0 register, and SPI will first send the LSB if LF=1, or the MSB if LF=0. The data order is fixed to MSB first in TI mode.

#### 19.3.4. 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 transmits/receives data only when NSS level is low. In software NSS mode, NSS pin is not used.

**Table 19-3. NSS function in slave mode**

Mode	Register configuration	Description
Slave hardware NSS mode	MSTMOD = 0 SWNSSEN = 0	SPI slave gets NSS level from NSS pin.
Slave software NSS mode	MSTMOD = 0 SWNSSEN = 1	SPI slave NSS level is determined by the SWNSS bit. SWNSS = 0: NSS level is low SWNSS = 1: NSS level is high

##### 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 to slave mode and triggers a master fault flag CONFERR.

If the application wants to use NSS line to control the SPI slave, NSS should be configured to hardware output mode (SWNSSEN=0, NSSDRV=1). NSS stays high after SPI is enabled and goes low when transmission or reception process begins.

The application may also use a general purpose IO as NSS pin to realize more flexible NSS.

**Table 19-4. NSS function in master mode**

Mode	Register configuration	Description
Master hardware NSS output mode	MSTMOD = 1 SWNSSEN = 0 NSSDRV=1	Applicable to single-master mode. The master uses the NSS pin to control the SPI slave device. At this time, the NSS is configured as the hardware output mode. NSS goes low after enabling SPI.
Master hardware NSS input mode	MSTMOD = 1 SWNSSEN = 0 NSSDRV=0	Applicable to multi-master mode. At this time, NSS is configured as hardware input mode. Once the NSS pin is pulled low, SPI will automatically enter slave mode, and a master configuration error will occur and the CONFERR bit will be set to 1.
Master software NSS mode	MSTMOD = 1 SWNSSEN = 1 SWNSS = 0 NSSDRV: Don't care	Applicable to multi-master mode. Once SWNSS = 0, SPI will automatically enter slave mode, and a master configuration error will occur and the CONFERR bit will be 1.
	MSTMOD = 1 SWNSSEN = 1 SWNSS = 1 NSSDRV: Don't care	The slave can use hardware or software NSS mode.

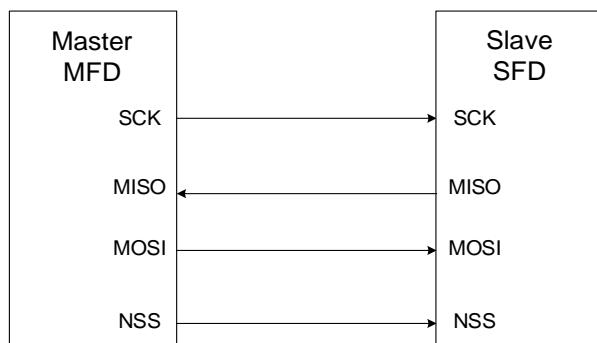
### 19.3.5. SPI operation modes

**Table 19-5. SPI operation modes**

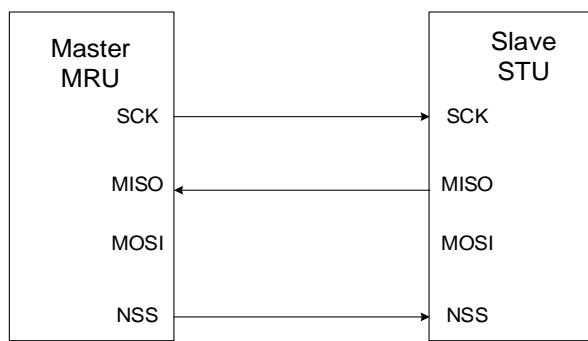
Mode	Description	Register Configuration	Data Pin Usage
MFD	Master Full-Duplex	MSTMOD = 1 RO = 0 BDEN = 0 BDOEN: Don't care	MOSI: Transmission MISO: Reception
MTU	Master Transmission with unidirectional connection	MSTMOD = 1 RO = 0	MOSI: Transmission MISO: Not used

<b>Mode</b>	<b>Description</b>	<b>Register Configuration</b>	<b>Data Pin Usage</b>
		BDEN = 0 BDOEN: Don't care	
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 bidirectional connection	MSTMOD = 0 RO = 0 BDEN = 1 BDOEN = 1	MOSI: Not used MISO: Transmission
SRB	Slave Reception with bidirectional connection	MSTMOD = 0 RO = 0 BDEN = 1 BDOEN = 0	MOSI: Not used MISO: Reception

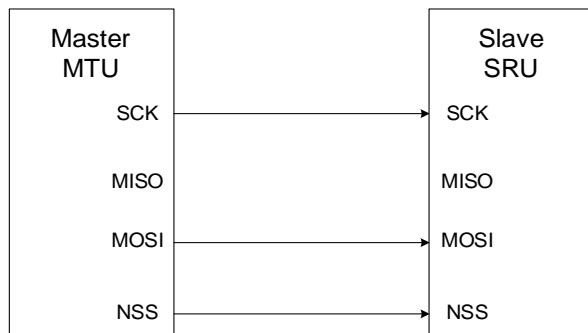
**Figure 19-4. A typical Full-duplex connection**



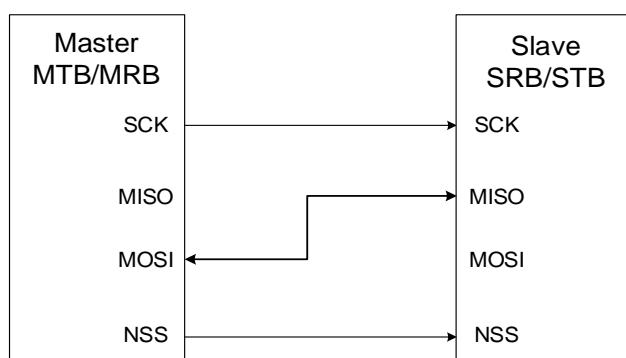
**Figure 19-5. A typical simplex connection (Master: Receive, Slave: Transmit)**



**Figure 19-6. A typical simplex connection (Master: Transmit only, Slave: Receive)**



**Figure 19-7. 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. If NSSP mode is used, set NSSP bit in SPI\_CTL1 register, otherwise, ignore this step.
8. Configure MSTMOD, RO, BDEN and BDOEN depending on the operation modes described above.
9. If Quad-SPI mode is used, set the QMOD bit in SPI\_QCTL register. Ignore this step if Quad-SPI mode is not used.
10. Enable the SPI (set the SPIEN bit).

## SPI basic transmission and reception sequence

### Transmission sequence

After the initialization sequence, the SPI is enabled and stays at idle state. In master mode, the transmission starts when the application writes a data into the transmit buffer. In slave mode, the transmission starts when SCK clock signal begins to toggle at SCK pin and NSS level is low, so application should ensure that data is already written into transmit buffer before the transmission starts in slave mode.

When SPI begins to send a data frame, it first loads this data frame from the data buffer to the shift register and then begins to transmit the loaded data frame, TBE (transmit buffer empty) flag is set after the first bit of this frame is transmitted. After TBE flag is set, which means the transmit buffer is empty, the application should write SPI\_DATA register again if it has more data to transmit.

In master mode, software should write the next data into SPI\_DATA register before the transmission of current data frame is completed if it desires to generate continuous transmission.

### Reception sequence

The incoming data will be moved from shift register to the receive buffer after the last valid sample clock and also, RBNE (receive buffer not empty) will be set. The application should read SPI\_DATA register to get the received data and this will clear the RBNE flag automatically. In MRU and MRB modes, hardware continuously sends clock signal to receive the next data

frame, while in full-duplex master mode (MFD), hardware only receives the next data frame when the transmit buffer is not empty.

### SPI operation sequence in different modes (Not Quad-SPI, TI mode or NSSP mode)

In full-duplex mode, either MFD or SFD, application should monitor the RBNE and TBE flags and follow the sequences described above.

The transmission mode (MTU, MTB, STU or STB) is similar to full-duplex mode, except that application should ignore the RBNE and OVRE flags and only perform transmission sequence described above.

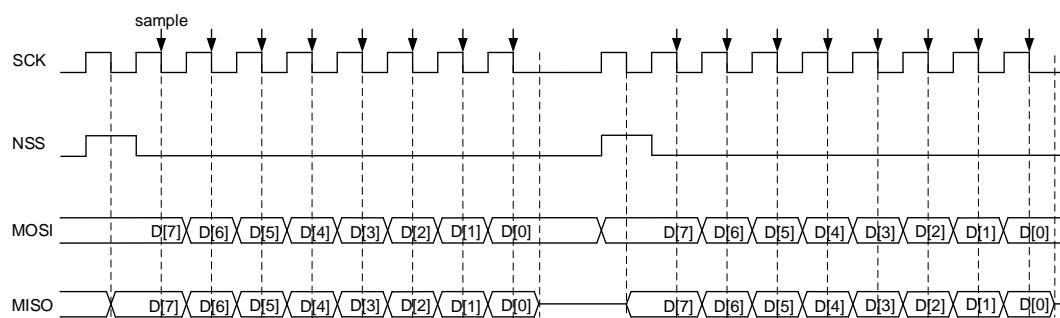
In master reception mode (MRU or MRB), the behavior is different from full-duplex mode or transmission mode. In MRU or MRB mode, the SPI continuously generates SCK just after SPI is enabled, until the SPI is disabled. So the application should ignore the TBE flag and read out reception buffer in time after the RBNE flag is set, otherwise a data overrun fault will occur.

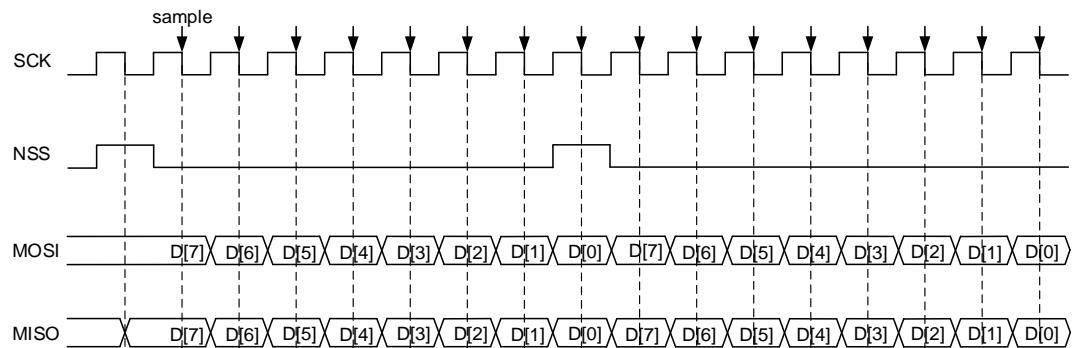
The slave reception mode (SRU or SRB) is similar to full-duplex mode, except that application should ignore the TBE flag and only perform reception sequence described above.

### SPI TI mode

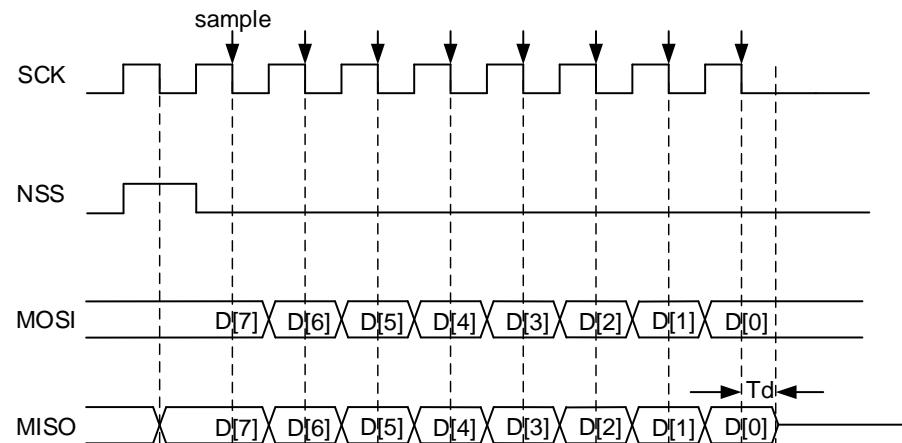
SPI TI mode takes NSS as a special frame header flag signal and its operation sequence is similar to normal mode described above. The modes described above (MFD, MTU, MRU, MTB, MRB, SFD, STU, SRU, STB and SRB) are still supported in TI mode. While, in TI mode the CKPL and CKPH bits in SPI\_CTL0 registers take no effect and the SCK sample edge is falling edge.

**Figure 19-8. Timing diagram of TI master mode with discontinuous transfer**



**Figure 19-9. 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 19-10. 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.

$$T_d = \frac{T_{\text{bit}}}{2} + 5 * T_{\text{pclk}} \quad (21-1)$$

For example, if PSC[2:0] = 010,  $T_d$  is 9\*Tpclk.

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**

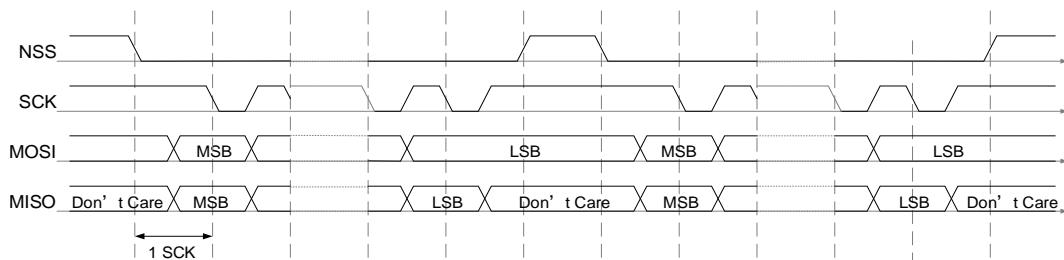
This function is controlled by NSSP bit in SPI\_CTL1 register, for this function to fully take place, several additional conditions must be met, users must also set the device into master

mode, and frame format should follow the normal SPI protocol, and set the data capture edge to first clock transition.

In summary: NSSP = 1; MSTMOD = 1; CKPH = 0;

When active, a pulse duration of least 1 SCK clock period is inserted between successive data frames depending on internal data transmit buffer status, multiple SCK clock cycle interval is 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 19-11. Timing diagram of NSS pulse with continuous transmit**



### Quad-SPI mode operation sequence

The Quad-SPI mode is designed to control quad SPI flash.

In order to enter Quad-SPI mode, the software should first verify that the TBE bit is set and TRANS bit is cleared, then set QMOD bit in SPI\_QCTL register. In Quad-SPI mode, BDEN, BDOEN, CRCEN, CRCNT, FF16, RO and LF in SPI\_CTL0 register should be kept cleared and MSTMOD should be set to ensure that SPI is in master mode. SPIEN, PSC, CKPL and CKPH should be configured as desired.

There are 2 operation modes in Quad-SPI mode: quad write and quad read, decided by QRD bit in SPI\_QCTL register.

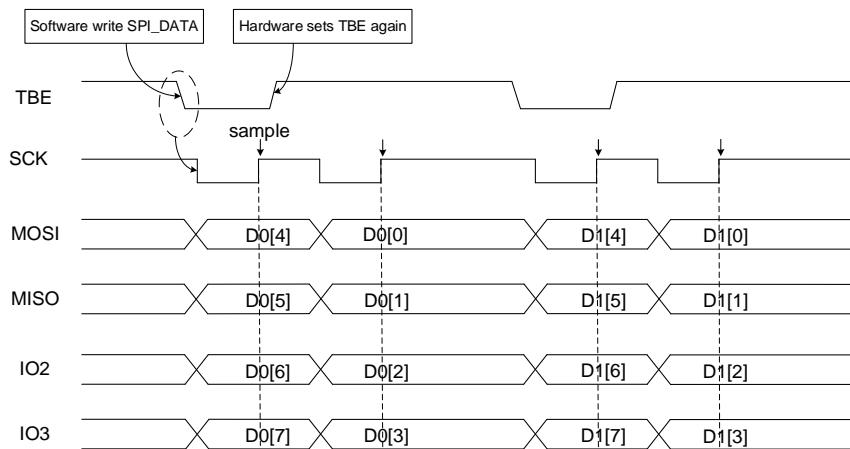
#### Quad write operation

SPI works in quad write mode when QMOD is set and QRD is cleared in SPI\_QCTL register. In this mode, MOSI, MISO, IO2 and IO3 are all used as output pins. SPI begins to generate clock on SCK line and transmit data on MOSI, MISO, IO2 and IO3 as soon as data is written into SPI\_DATA (TBE is cleared) and SPIEN is set. Once SPI starts transmission, it always checks TBE status at the end of a frame and stops when condition is not met.

The operation flow for transmitting in quad mode:

1. Configure clock prescaler, clock polarity, phase, etc. in SPI\_CTL0 and SPI\_CTL1 based on your application requirements.
2. Set QMOD bit in SPI\_QCTL register and then enable SPI by setting SPIEN in SPI\_CTL0.
3. Write the byte to SPI\_DATA register and the TBE will be cleared.
4. Wait until TBE is set by hardware again before writing the next byte.

Figure 19-12. Timing diagram of quad write operation in Quad-SPI mode



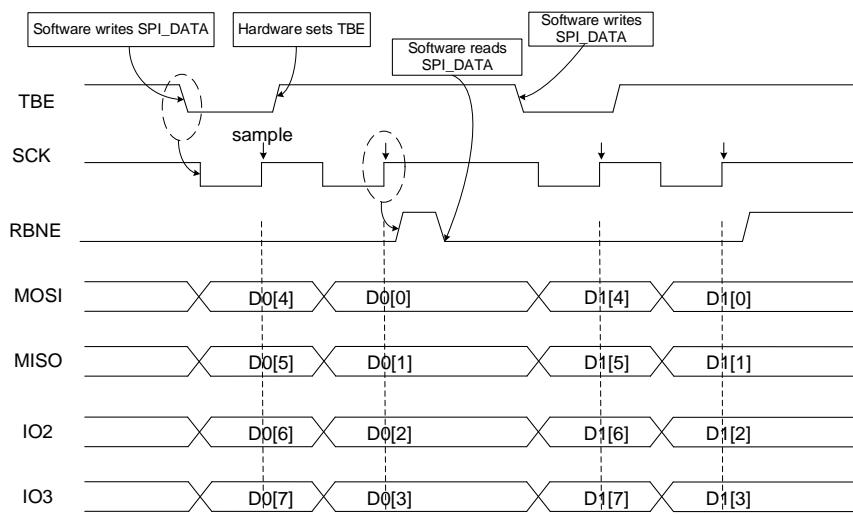
### Quad read operation

SPI works in quad read mode when QMOD and QRD are both set in SPI\_QCTL register. In this mode, MOSI, MISO, IO2 and IO3 are all used as input pins. SPI begins to generate clock on SCK line as soon as a data is written into SPI\_DATA (TBE is cleared) and SPIEN is set. Writing data into SPI\_DATA is only to generate SCK clocks, so the written data can be any value. Once SPI starts transmission, it always checks SPIEN and TBE status at the end of a frame and stops when condition is not met. So, software should always write dummy data into SPI\_DATA to make SPI generate SCK.

The operation flow for receiving in quad mode:

1. Configure clock prescaler, clock polarity, phase, etc. in SPI\_CTL0 and SPI\_CTL1 register based on your application requirements.
2. Set QMOD and QRD bits in SPI\_QCTL register and then enable SPI by setting SPIEN in SPI\_CTL0 register.
3. Write an arbitrary byte (for example, 0xFF) to SPI\_DATA register.
4. Wait until the RBNE flag is set and read SPI\_DATA to get the received byte.
5. Write an arbitrary byte (for example, 0xFF) to SPI\_DATA to receive the next byte.

**Figure 19-13. Timing diagram of quad read operation in Quad-SPI mode**



### SPI disabling sequence

Different sequences are used to disable the SPI in different operation modes:

#### MFD SFD

Wait for the last RBNE flag and then receive the last data. Confirm that TBE=1 and TRANS=0. At last, disable the SPI by clearing SPIEN bit.

#### MTU MTB STU STB

Write the last data into SPI\_DATA and wait until the TBE flag is set and then wait until the TRANS flag is cleared. Disable the SPI by clearing SPIEN bit.

#### MRU MRB

After getting the second last RBNE flag, read out this data and delay for a SCK clock time and then, disable the SPI by clearing SPIEN bit. Wait until the last RBNE flag is set and read out the last data.

#### SRU SRB

Application can disable the SPI when it doesn't want to receive data, and then wait until the TRANS=0 to ensure the on-going transfer completes.

#### TI mode

The disabling sequence of TI mode is the same as the sequences described above.

#### NSS pulse mode

The disabling sequence of NSS pulse mode is the same as the sequences described above.

### Quad-SPI mode

Before leaving quad wire mode or disabling SPI, software should first check that, TBE bit is set and TRANS bit is cleared, then the QMOD bit in SPI\_QCTL register and SPIEN bit in SPI\_CTL0 register are cleared.

### 19.3.6. DMA function

The DMA function frees the application from data writing and reading process during transfer, thus improve the system efficiency.

DMA function in SPI is enabled by setting DMATEN and DMAREN bits in SPI\_CTL1 register. To use DMA function, application should first correctly configure DMA modules, then configure SPI module according to the initialization sequence, at last enable SPI.

After being enabled, If DMATEN is set, SPI will generate a DMA request each time TBE=1, then DMA will acknowledge to this request and write data into the SPI\_DATA register automatically. If DMAREN is set, SPI will generate a DMA request each time RBNE=1, then DMA will acknowledge to this request and read data from the SPI\_DATA register automatically.

### 19.3.7. CRC function

There are two CRC calculators in SPI: one for transmission and the other for reception. The CRC calculation uses the polynomial in SPI\_CRCPOLY register.

Application can switch on the CRC function by setting CRCEN bit in SPI\_CTL0 register. The CRC calculators continuously calculate CRC for each bit transmitted and received on lines, and the calculated CRC values can be read from SPI\_TCRC and SPI\_RCRC register.

To transmit the calculated CRC value, application should set the CRCNT bit in SPI\_CTL0 register after the last data is written to the transmit buffer. In full-duplex mode (MFD or SFD) the SPI treats the incoming data as a CRC value when it transmits a CRC and will check the received CRC value. In reception mode (MRB, MRU, SRU and SRB), the application should set the CRCNT bit after the second-last data frame is received. When CRC checking fails, the CRCERR flag will be set.

If DMA function is enabled, application doesn't need to operate CRCNT bit and hardware will automatically process the CRC transmitting and checking.

### 19.3.8. SPI interrupts

#### Status flags

##### ■ Transmit buffer empty flag (TBE)

This bit is set when the transmit buffer is empty, the software can write the next data to the transmit buffer by writing the SPI\_DATA register.

##### ■ Receive buffer not empty flag (RBNE)

This bit is set when receive buffer is not empty, which means that one data is received and stored in the receive buffer, and software can read the data by reading the SPI\_DATA register.

## ■ SPI Transmitting On-Going flag (TRANS)

TRANS is a status flag to indicate whether the transfer is on-going or not. It is set and cleared by internal hardware and not controlled by software. This flag doesn't generate any interrupt.

## Error conditions

### ■ Configuration Fault Error (CONFERR)

CONFERR is an error flag in master mode. In NSS hardware mode and the NSSDRV is not enabled, the CONFERR is set when the NSS pin is pulled low. In NSS software mode, the CONFERR is set when the SWNSS bit is 0. When the CONFERR is set, the SPIEN bit and the MSTMOD bit are cleared by hardware, the SPI is disabled and the device is forced into slave mode.

The SPIEN and MSTMOD bit are write protection until the CONFERR is cleared. The CONFERR bit of the slave cannot be set. In a multi-master configuration, the device can be in slave mode with CONFERR bit set, which means there might have been a multi-master conflict for system control.

### ■ Rx Overrun Error (RXORERR)

The RXORERR bit is set if a data is received when the RBNE is set. That means, the last data has not been read out and the newly incoming data is received. The receive buffer contents won't be covered with the newly incoming data, so the newly incoming data is lost.

### ■ 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 value received in the SPI\_RCRC register will be compared with the CRC value received immediately after the last frame of data. The CRCERR will set when they are different.

**Table 19-6. SPI interrupt requests**

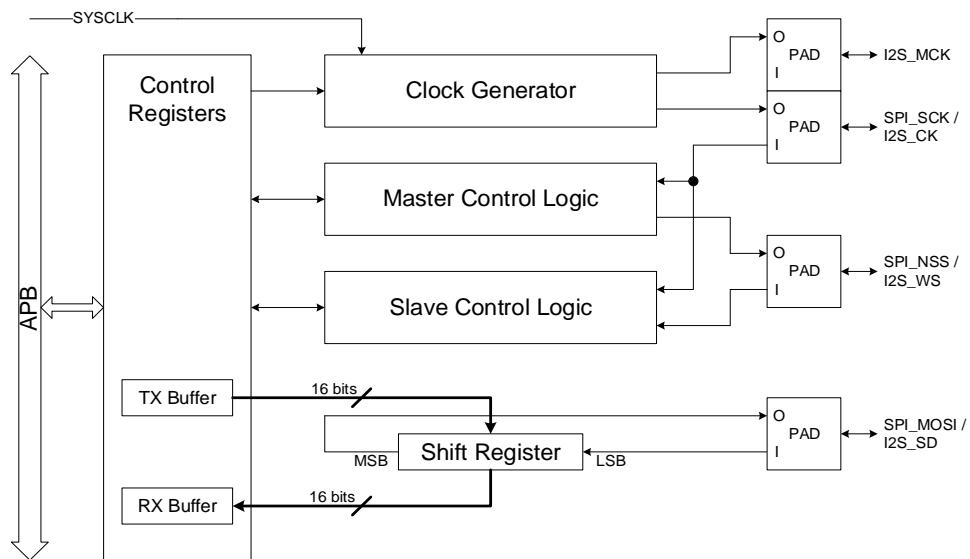
Flag	Description	Clear Method	Interrupt Enable bit
TBE	Transmit buffer empty	Write SPI_DATA register.	TBEIE
RBNE	Receive buffer not empty	Read SPI_DATA register	RBNEIE
CONFERR	Configuration Fault Error	Read or write SPI_STAT register, then write SPI_CTL0 register.	ERRIE
RXORERR	Rx Overrun Error	Read SPI_DATA register, then read SPI_STAT register.	

Flag	Description	Clear Method	Interrupt Enable bit
CRCERR	CRC error	Write 0 to CRCERR bit	
FERR	TI Mode Format Error	Write 0 to FERR bit	

## 19.4. I2S function overview

### 19.4.1. I2S block diagram

Figure 19-14. Block diagram of I2S



There are five sub modules to support I2S function, including control registers, clock generator, master control logic, slave control logic and shift register. All the user configuration registers are implemented in the control registers module, including the TX buffer and RX buffer. The clock generator is used to produce I2S communication clock in master mode. The master control logic is implemented to generate the I2S\_WS signal and control the communication in master mode. The slave control logic is implemented to control the communication in slave mode according to the received I2SCK and I2S\_WS. The shift register handles the serial data transmission and reception on I2S\_SD.

### 19.4.2. 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 equal to 256 x Fs, and Fs is the audio sampling frequency.

### 19.4.3. I2S audio standards

The I2S audio standard is selected by the I2SSTD bits in the SPI\_I2SCTL register. Four audio standards are supported, including I2S Phillips standard, MSB justified standard, LSB justified standard, and PCM standard. All standards except PCM handle audio data time-multiplexed on two channels (the left channel and the right channel). For these standards, the I2S\_WS signal indicates the channel side. For PCM standard, the I2S\_WS signal indicates frame synchronization information.

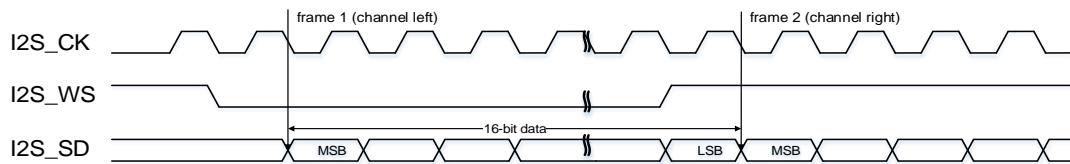
The data length and the channel length are configured by the DTLEN bits and CHLEN bit in the SPI\_I2SCTL register. Since the channel length must be greater than or equal to the data length, four packet types are available. They are 16-bit data packed in 16-bit frame, 16-bit data packed in 32-bit frame, 24-bit data packed in 32-bit frame, and 32-bit data packed in 32-bit frame. The data buffer for transmission and reception is 16-bit wide. In the case that the data length is 24 bits or 32 bits, two write or read operations to or from the SPI\_DATA register are needed to complete a frame. In the case that the data length is 16 bits, only one write or read operation to or from the SPI\_DATA register is needed to complete a frame. When using 16-bit data packed in 32-bit frame, 16-bit 0 is inserted by hardware automatically to extend the data to 32-bit format.

For all standards and packet types, the most significant bit (MSB) is always sent first. For all standards based on two channels time-multiplexed, the channel left is always sent first followed by the channel right.

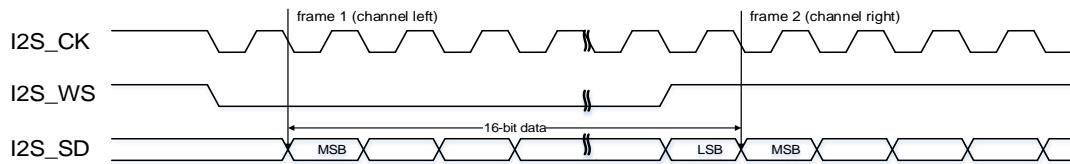
#### I2S Phillips standard

For I2S Phillips standard, I2S\_WS and I2S\_SD are updated on the falling edge of I2S\_CK. The timing diagrams for each configuration are shown below.

**Figure 19-15. I2S Phillips standard timing diagram (DTLEN=00, CHLEN=0, CKPL=0)**

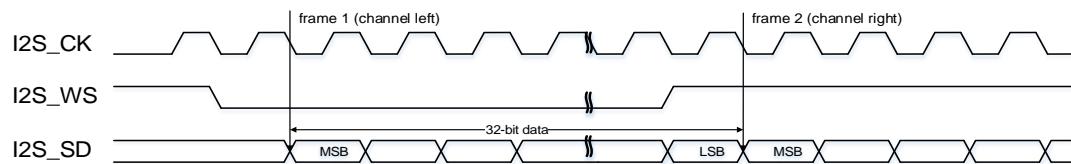


**Figure 19-16. I2S Phillips standard timing diagram (DTLEN=00, CHLEN=0, CKPL=1)**

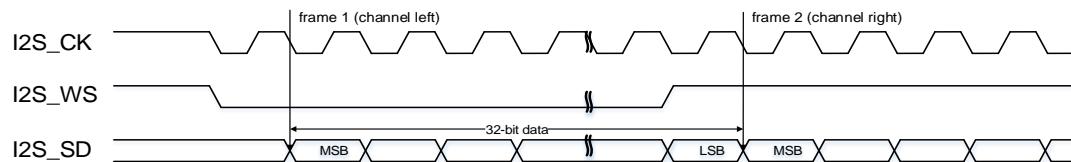


When the packet type is 16-bit data packed in 16-bit frame, only one write or read operation to or from the SPI\_DATA register is needed to complete a frame.

**Figure 19-17. I2S Phillips standard timing diagram (DTLEN=10, CHLEN=1, CKPL=0)**

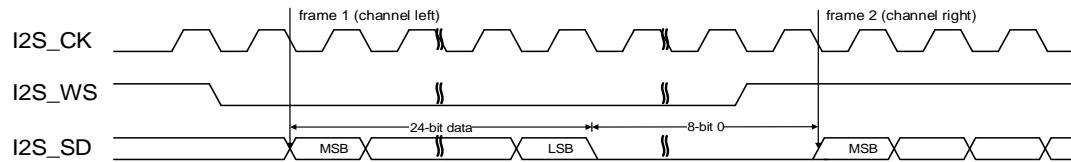


**Figure 19-18. I2S Phillips standard timing diagram (DTLEN=10, CHLEN=1, CKPL=1)**

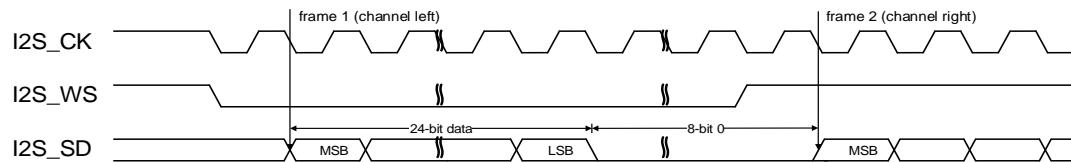


When the packet type is 32-bit data packed in 32-bit frame, two write or read operations to or from the SPI\_DATA register are needed to complete a frame. In transmission mode, if a 32-bit data is going to be sent, the first data written to the SPI\_DATA register should be the higher 16 bits, and the second one should be the lower 16 bits. In reception mode, if a 32-bit data is received, the first data read from the SPI\_DATA register should be higher 16 bits, and the second one should be the lower 16 bits.

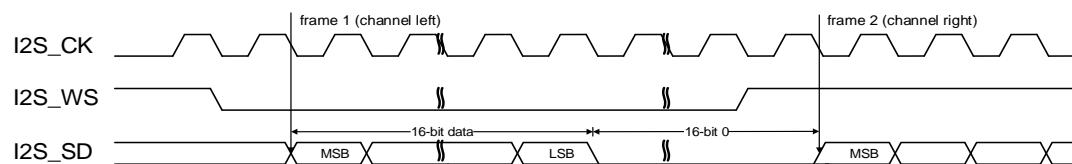
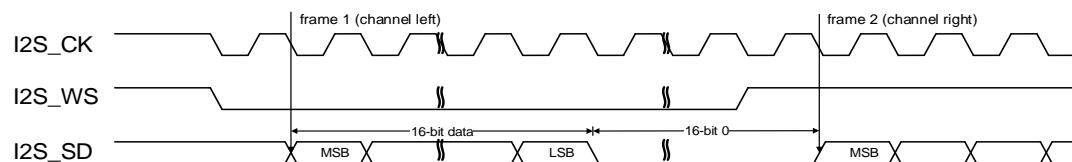
**Figure 19-19. I2S Phillips standard timing diagram (DTLEN=01, CHLEN=1, CKPL=0)**



**Figure 19-20. I2S Phillips standard timing diagram (DTLEN=01, CHLEN=1, CKPL=1)**



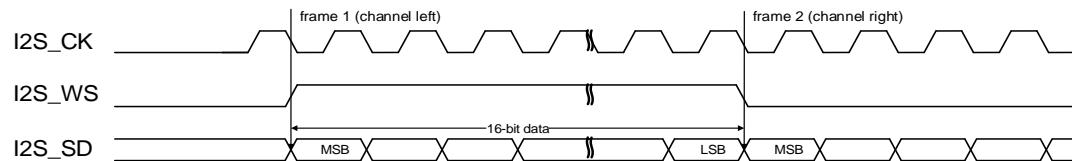
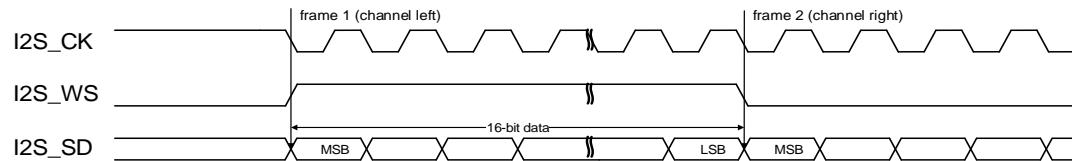
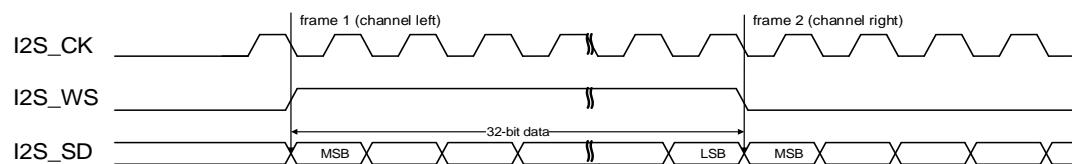
When the packet type is 24-bit data packed in 32-bit frame, two write or read operations to or from the SPI\_DATA register are needed to complete a frame. In transmission mode, if a 24-bit data D[23:0] is going to be sent, the first data written to the SPI\_DATA register should be the higher 16 bits: D[23:8], and the second one should be a 16-bit data. The higher 8 bits of this 16-bit data should be D[7:0] and the lower 8 bits can be any value. In reception mode, if a 24-bit data D[23:0] is received, the first data read from the SPI\_DATA register is D[23:8], and the second one is a 16-bit data. The higher 8 bits of this 16-bit data are D[7:0] and the lower 8 bits are zeros.

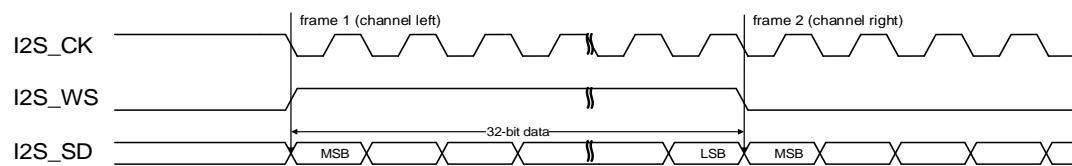
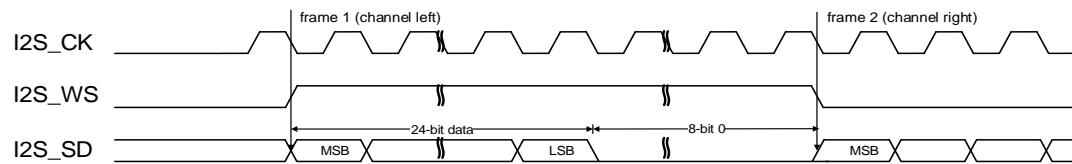
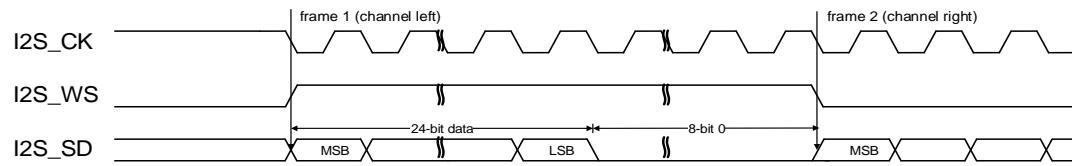
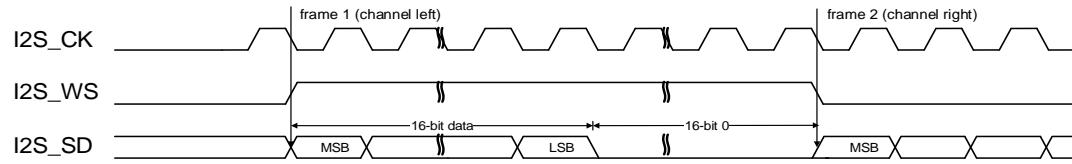
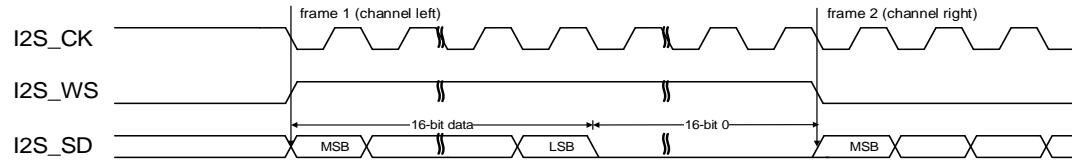
**Figure 19-21. I2S Phillips standard timing diagram (DTLEN=00, CHLEN=1, CKPL=0)**

**Figure 19-22. I2S Phillips standard timing diagram (DTLEN=00, CHLEN=1, CKPL=1)**


When the packet type is 16-bit data packed in 32-bit frame, only one write or read operation to or from the SPI\_DATA register is needed to complete a frame. The 16 remaining bits are forced by hardware to 0x0000 to extend the data to 32-bit format.

### **MSB justified standard**

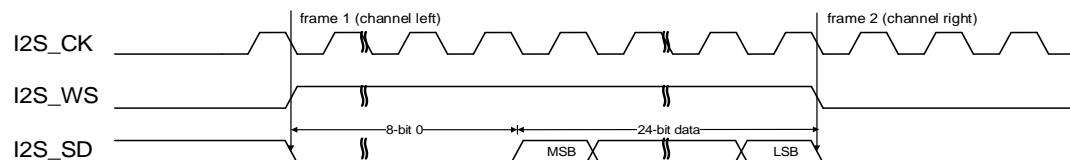
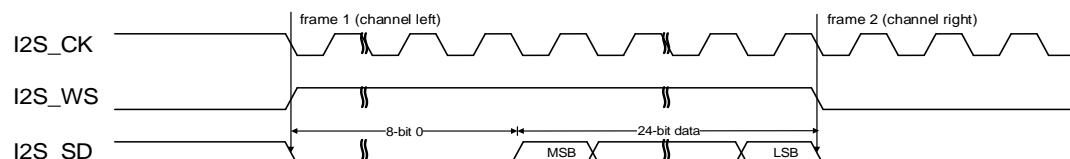
For MSB justified standard, I2S\_WS and I2S\_SD are updated on the falling edge of I2S\_CK. The SPI\_DATA register is handled in the exactly same way as that for I2S Phillips standard. The timing diagrams for each configuration are shown below.

**Figure 19-23. MSB justified standard timing diagram (DTLEN=00, CHLEN=0, CKPL=0)**

**Figure 19-24. MSB justified standard timing diagram (DTLEN=00, CHLEN=0, CKPL=1)**

**Figure 19-25. MSB justified standard timing diagram (DTLEN=10, CHLEN=1, CKPL=0)**


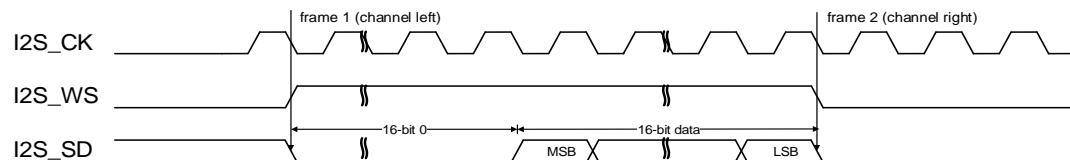
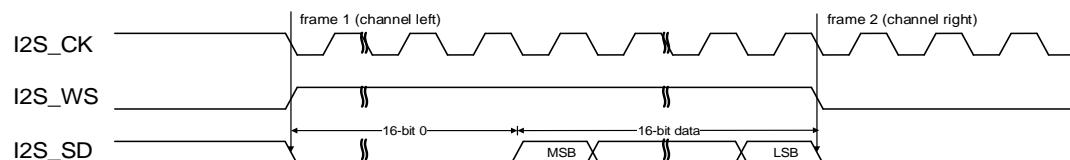
**Figure 19-26. MSB justified standard timing diagram (DTLEN=10, CHLEN=1, CKPL=1)**

**Figure 19-27. MSB justified standard timing diagram (DTLEN=01, CHLEN=1, CKPL=0)**

**Figure 19-28. MSB justified standard timing diagram (DTLEN=01, CHLEN=1, CKPL=1)**

**Figure 19-29. MSB justified standard timing diagram (DTLEN=00, CHLEN=1, CKPL=0)**

**Figure 19-30. MSB justified standard timing diagram (DTLEN=00, CHLEN=1, CKPL=1)**


### **LSB justified standard**

For LSB justified standard, I2S\_WS and I2S\_SD are updated on the falling edge of I2S\_CK. In the case that the channel length is equal to the data length, LSB justified standard and MSB justified standard are exactly the same. In the case that the channel length is greater than the data length, the valid data is aligned to LSB for LSB justified standard while the valid data is aligned to MSB for MSB justified standard. The timing diagrams for the cases that the channel length is greater than the data length are shown below.

**Figure 19-31. LSB justified standard timing diagram (DTLEN=01, CHLEN=1, CKPL=0)**

**Figure 19-32. LSB justified standard timing diagram (DTLEN=01, CHLEN=1, CKPL=1)**


When the packet type is 24-bit data packed in 32-bit frame, two write or read operations to or from the SPI\_DATA register are needed to complete a frame. In transmission mode, if a 24-bit data D [23:0] is going to be sent, the first data written to the SPI\_DATA register should be a 16-bit data. The higher 8 bits of the 16-bit data can be any value and the lower 8 bits should be D [23:16]. The second data written to the SPI\_DATA register should be D [15:0]. In reception mode, if a 24-bit data D [23:0] is received, the first data read from the SPI\_DATA register is a 16-bit data. The high 8 bits of this 16-bit data are zeros and the lower 8 bits are D [23:16]. The second data read from the SPI\_DATA register is D [15:0].

**Figure 19-33. LSB justified standard timing diagram (DTLEN=00, CHLEN=1, CKPL=0)**

**Figure 19-34. LSB justified standard timing diagram (DTLEN=00, CHLEN=1, CKPL=1)**


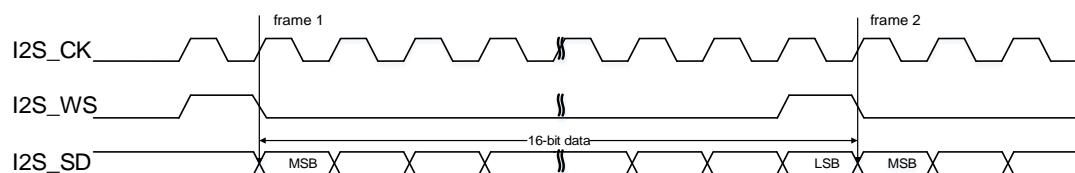
When the packet type is 16-bit data packed in 32-bit frame, only one write or read operation to or from the SPI\_DATA register is needed to complete a frame. The 16 remaining bits are forced by hardware to 0x0000 to extend the data to 32-bit format.

### **PCM standard**

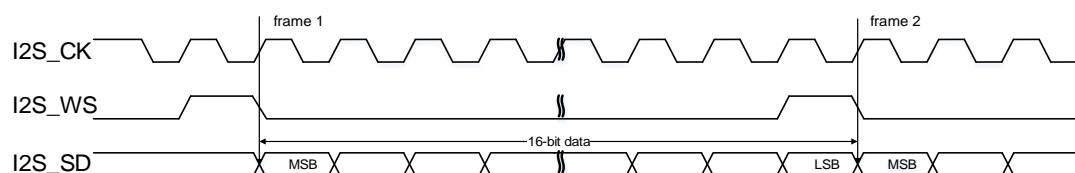
For PCM standard, I2S\_WS and I2S\_SD are updated on the rising edge of I2S\_CK, and the I2S\_WS signal indicates frame synchronization information. Both the short frame

synchronization mode and the long frame synchronization mode are available and configurable using the PCMSMOD bit in the SPI\_I2SCTL register. The SPI\_DATA register is handled in the exactly same way as that for I2S Phillips standard. The timing diagrams for each configuration of the short frame synchronization mode are shown below.

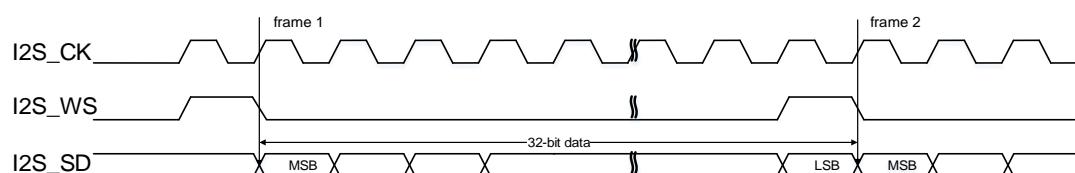
**Figure 19-35. PCM standard short frame synchronization mode timing diagram (DTLEN=00, CHLEN=0, CKPL=0)**



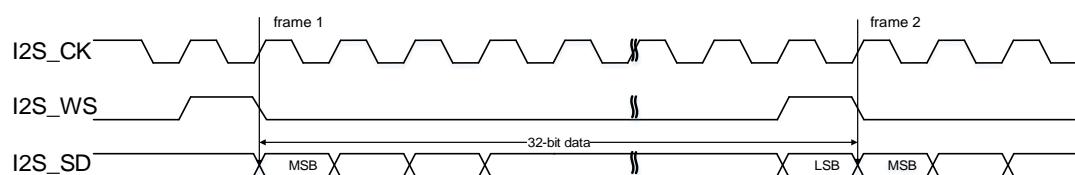
**Figure 19-36. PCM standard short frame synchronization mode timing diagram (DTLEN=00, CHLEN=0, CKPL=1)**



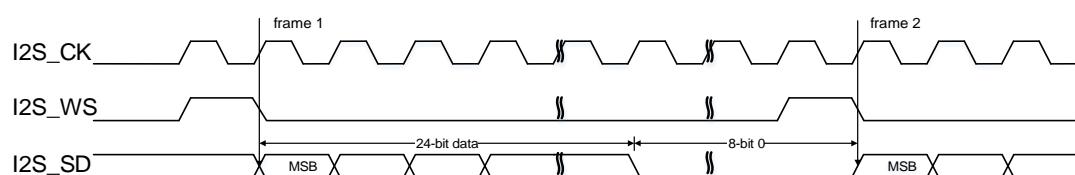
**Figure 19-37. PCM standard short frame synchronization mode timing diagram (DTLEN=10, CHLEN=1, CKPL=0)**



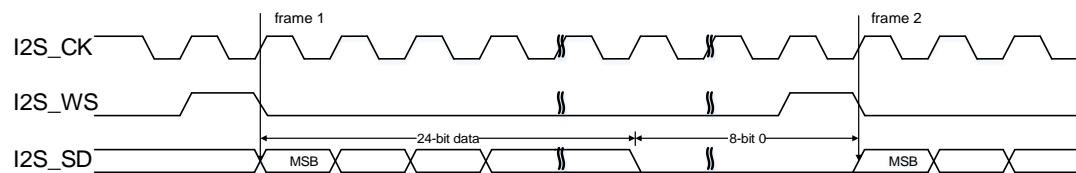
**Figure 19-38. PCM standard short frame synchronization mode timing diagram (DTLEN=10, CHLEN=1, CKPL=1)**



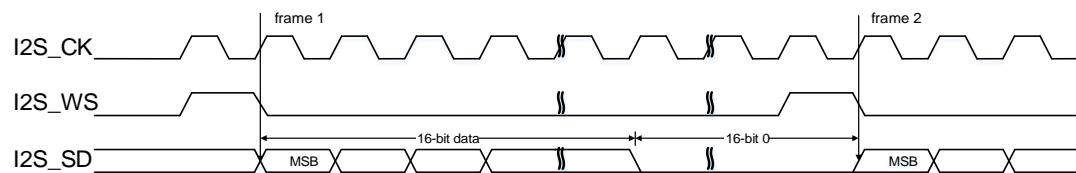
**Figure 19-39. PCM standard short frame synchronization mode timing diagram (DTLEN=01, CHLEN=1, CKPL=0)**



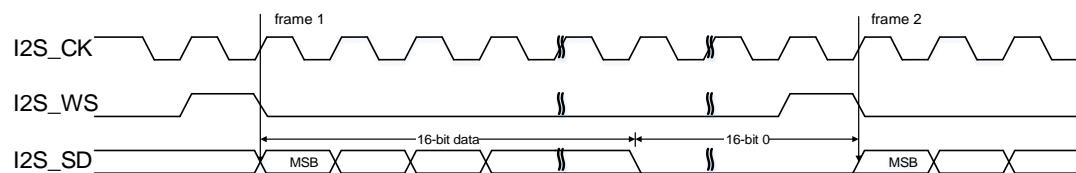
**Figure19-40. PCM standard short frame synchronization mode timing diagram (DTLEN=01, CHLEN=1, CKPL=1)**



**Figure 19-41. PCM standard short frame synchronization mode timing diagram (DTLEN=00, CHLEN=1, CKPL=0)**

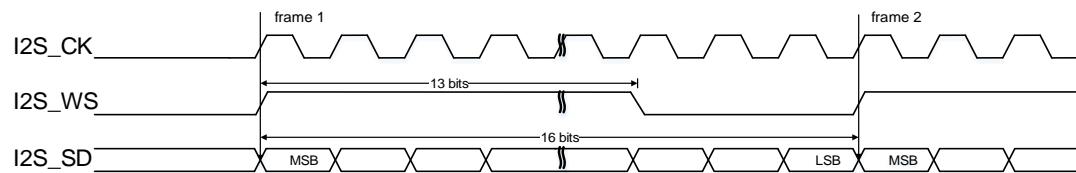


**Figure 19-42. PCM standard short frame synchronization mode timing diagram (DTLEN=00, CHLEN=1, CKPL=1)**

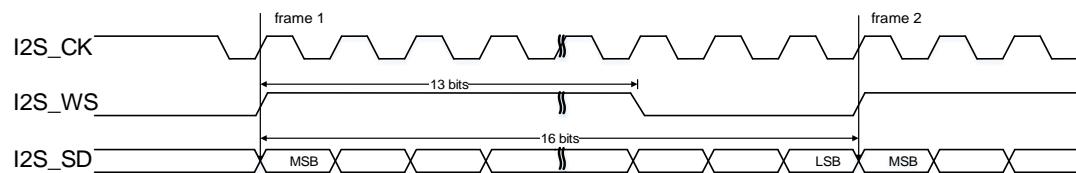


The timing diagrams for each configuration of the long frame synchronization mode are shown below.

**Figure 19-43. PCM standard long frame synchronization mode timing diagram (DTLEN=00, CHLEN=0, CKPL=0)**

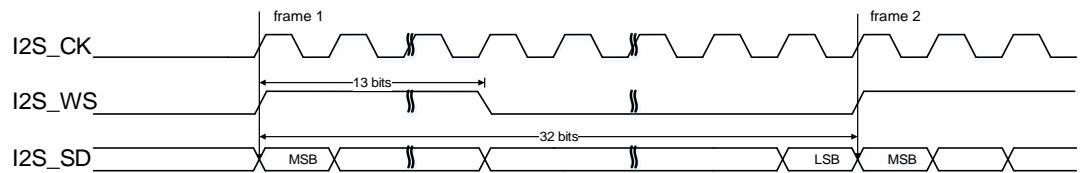


**Figure19-44. PCM standard long frame synchronization mode timing diagram (DTLEN=00, CHLEN=0, CKPL=1)**

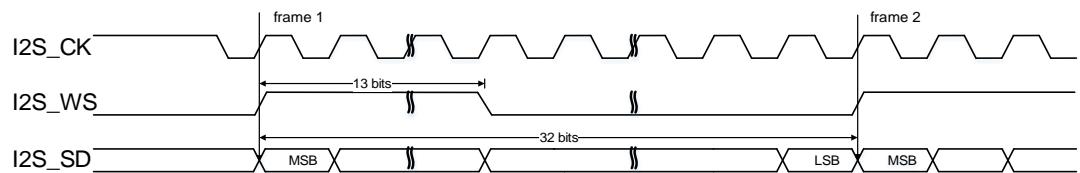


**Figure 19-45. PCM standard long frame synchronization mode timing diagram**

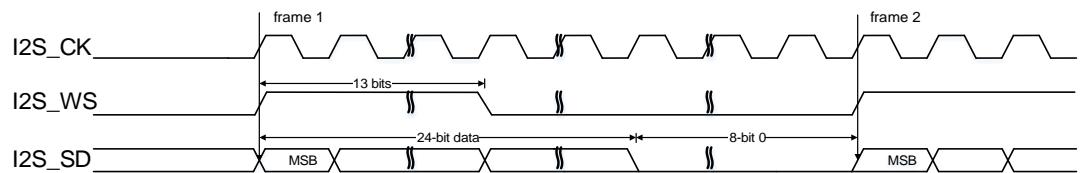
(DTLEN=10, CHLEN=1, CKPL=0)



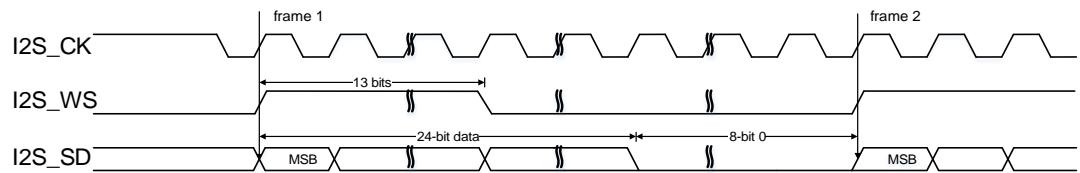
**Figure 19-46. PCM standard long frame synchronization mode timing diagram (DTLEN=10, CHLEN=1, CKPL=0)**



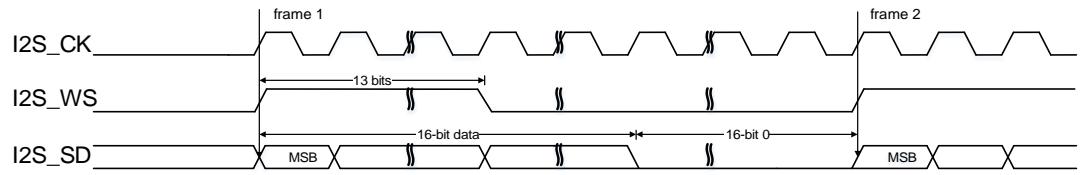
**Figure 19-47. PCM standard long frame synchronization mode timing diagram (DTLEN=01, CHLEN=1, CKPL=1)**



**Figure 19-48. PCM standard long frame synchronization mode timing diagram (DTLEN=01, CHLEN=1, CKPL=1)**

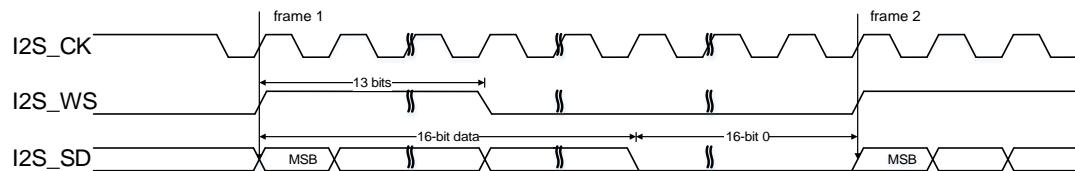


**Figure 19-49. PCM standard long frame synchronization mode timing diagram (DTLEN=00, CHLEN=1, CKPL=0)**



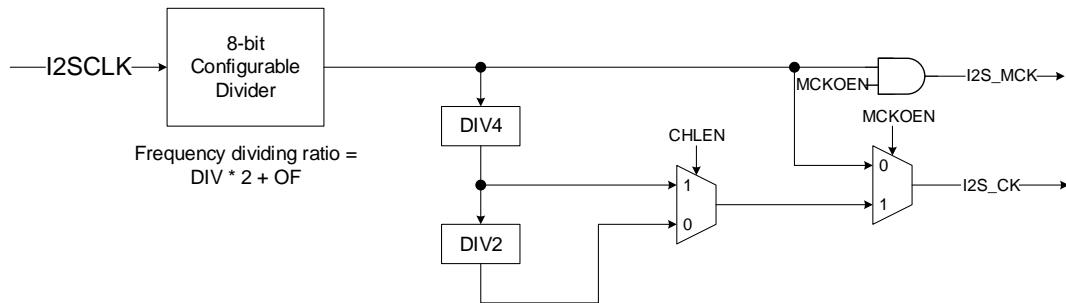
**Figure 19-50. PCM standard long frame synchronization mode timing diagram**

(DTLEN=00, CHLEN=1, CKPL=1)



#### 19.4.4. I2S clock

**Figure 19-51. Block diagram of I2S clock generator**



The block diagram of I2S clock generator is shown as [Figure 19-51. 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 I2S bitrate can be calculated by the formulas shown in [Table 19-7. I2S bitrate calculation formulas](#).

**Table 19-7. I2S bitrate calculation formulas**

MCKOEN	CHLEN	Formula
0	0	$\text{I2SCLK} / (\text{DIV} * 2 + \text{OF})$
0	1	$\text{I2SCLK} / (\text{DIV} * 2 + \text{OF})$
1	0	$\text{I2SCLK} / (8 * (\text{DIV} * 2 + \text{OF}))$
1	1	$\text{I2SCLK} / (4 * (\text{DIV} * 2 + \text{OF}))$

The relationship between audio sampling frequency ( $F_s$ ) and I2S bitrate is defined by the following formula:

$$F_s = \text{I2S bitrate} / (\text{number of bits per channel} * \text{number of channels})$$

So, in order to get the desired audio sampling frequency, the clock generator needs to be configured according to the formulas listed in [Table 19-8. Audio sampling frequency calculation formulas](#).

**Table 19-8. Audio sampling frequency calculation formulas**

MCKOEN	CHLEN	Formula
0	0	I2SCLK / (32 * (DIV * 2 + OF))
0	1	I2SCLK / (64 * (DIV * 2 + OF))
1	0	I2SCLK / (256 * (DIV * 2 + OF))
1	1	I2SCLK / (256 * (DIV * 2 + OF))

## 19.4.5. Operation

### Operation modes

The operation mode is selected by the I2SOPMOD bits in the SPI\_I2SCTL register. There are four available operation modes, including master transmission mode, master reception mode, slave transmission mode, and slave reception mode. The direction of I2S interface signals for each operation mode is shown in the [Table 19-9. Direction of I2S interface signals for each operation mode.](#)

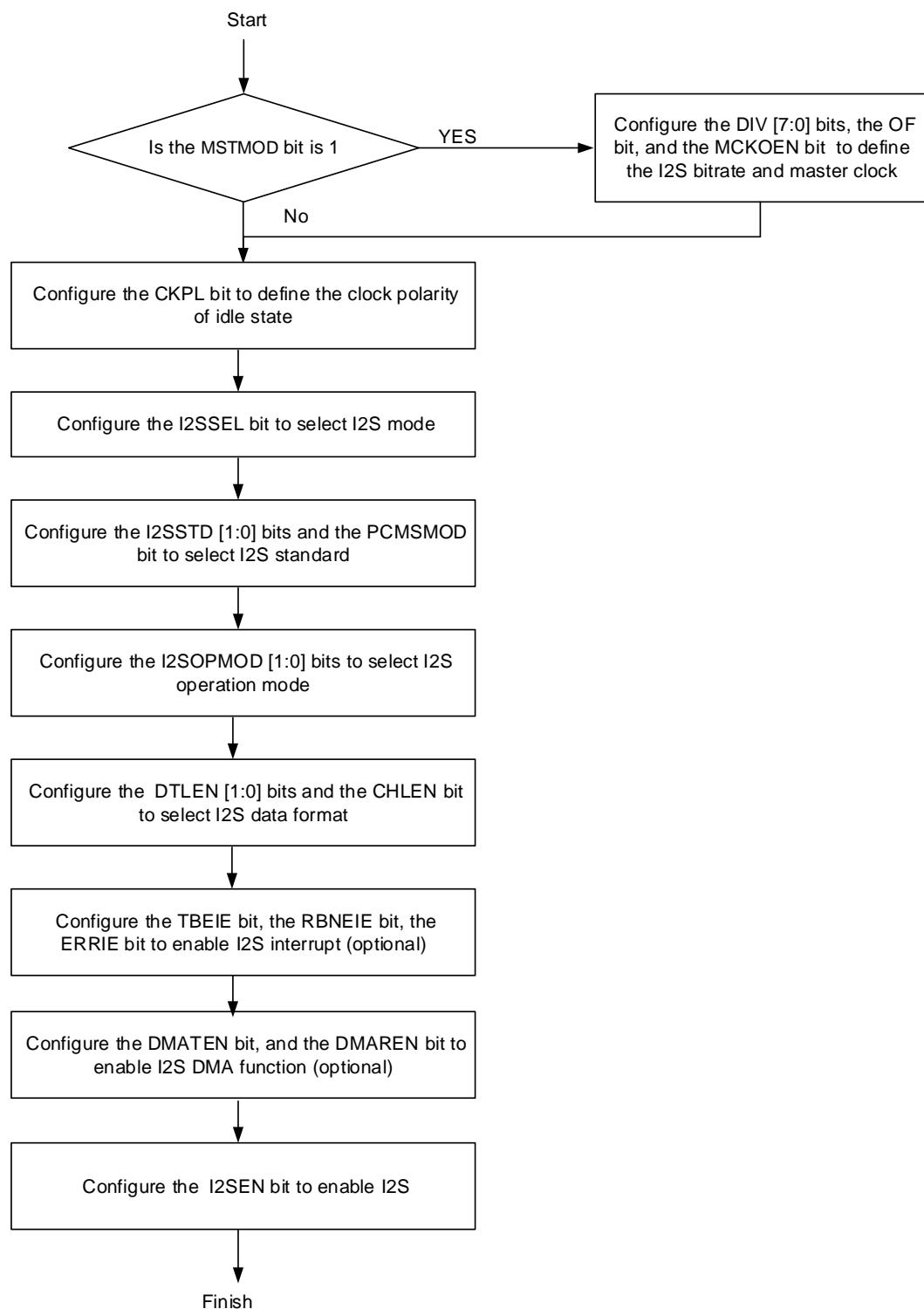
**Table 19-9. Direction of I2S interface signals for each operation mode**

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

- NU means the pin is not used by I2S and can be used by other functions.

### I2S initialization sequence

I2S initialization sequence is shown as below [Figure 19-52. I2S initialization sequence.](#)

**Figure 19-52. I2S initialization sequence**


### I2S master transmission sequence

The TBE flag is used to control the transmission sequence. As is mentioned before, the TBE flag indicates that the transmit buffer is empty, and an interrupt will be generated if the TBEIE bit in the SPI\_CTL1 register is set. At the beginning, the transmit buffer is empty (TBE is high)

and no transmission sequence is processing in the shift register. When a half word is written to the SPI\_DATA register (TBE goes low), the data is transferred from the transmit buffer to the shift register (TBE goes high) immediately. At the moment, the transmission sequence begins.

The data is parallel loaded into the 16-bit shift register, and shifted out serially to the I2S\_SD pin, MSB first. The next data should be written to the SPI\_DATA register, when the TBE flag is high. After a write operation to the SPI\_DATA register, the TBE flag goes low. When the current transmission finishes, the data in the transmit buffer is loaded into the shift register, and the TBE flag goes back high. Software should write the next audio data into SPI\_DATA register before the current data finishes, otherwise, the audio data transmission is not continuous.

For all standards except PCM, the I2SCH flag is used to distinguish the channel side to which the data to transfer belongs. The I2SCH flag is refreshed at the moment when the TBE flag goes high. At the beginning, the I2SCH flag is low, indicating the left channel data should be written to the SPI\_DATA register.

In order to switch off I2S, it is mandatory to clear the I2SEN bit after the TBE flag is high and the TRANS flag is low.

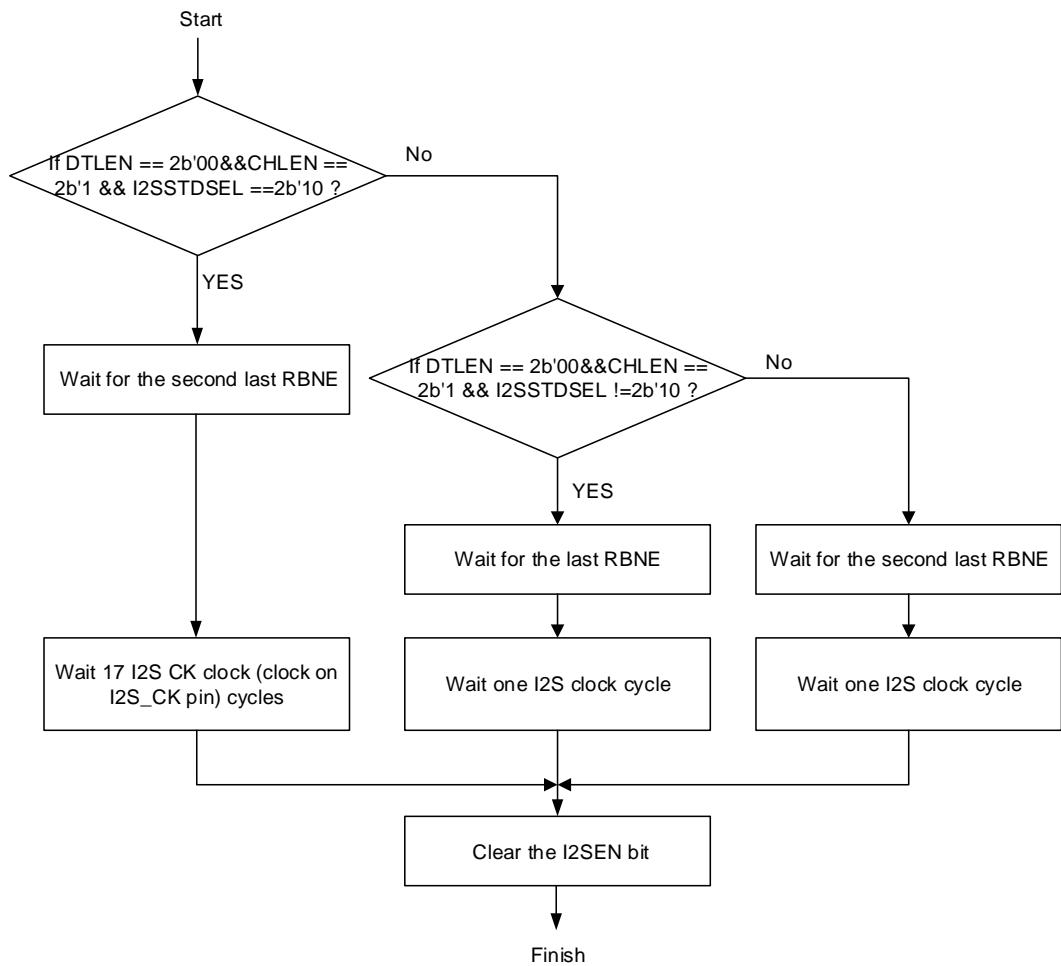
### I2S master reception sequence

The RBNE flag is used to control the reception sequence. As is mentioned before, the RBNE flag indicates the receive buffer is not empty, and an interrupt will be generated if the RBNEIE bit in the SPI\_CTL1 register is set. The reception sequence begins immediately when the I2SEN bit in the SPI\_I2SCTL register is set. At the beginning, the receive buffer is empty (RBNE is low). When a reception sequence finishes, the received data in the shift register is loaded into the receive buffer (RBNE goes high). The data should be read from the SPI\_DATA register, when the RBNE flag is high. After a read operation to the SPI\_DATA register, the RBNE flag goes low. It is mandatory to read the SPI\_DATA register before the end of the next reception. Otherwise, reception overrun error occurs. The RXORERR flag is set and an interrupt may be generated if the ERRIE bit in the SPI\_CTL1 register is set. In this case, it is necessary to switch off and then switch on I2S before resuming the communication.

For all standards except PCM, the I2SCH flag is used to distinguish the channel side to which the received data belongs. The I2SCH flag is refreshed at the moment when the RBNE flag goes high.

Different sequences are used to disable the I2S in different standards, data length and channel length. The sequences for each case are shown as below [Figure 19-53. I2S master reception disabling sequence](#).

Figure 19-53. I2S master reception disabling sequence



### I2S slave transmission sequence

The transmission sequence in slave mode is similar to that in master mode. The difference between them is described below.

In slave mode, the slave has to be enabled before the external master starts the communication. The transmission sequence begins when the external master sends the clock and when the I2S\_WS signal requests the transfer of data. The data has to be written to the SPI\_DATA register before the master initiates the communication. Software should write the next audio data into SPI\_DATA register before the current data finishes. Otherwise, transmission underrun error occurs. The TXURERR flag is set and an interrupt may be generated if the ERRIE bit in the SPI\_CTL1 register is set. In this case, it is mandatory to switch off and switch on I2S to resume the communication. In slave mode, I2SCH is sensitive to the I2S\_WS signal coming from the external master.

In order to switch off I2S, it is mandatory to clear the I2SEN bit after the TBE flag is high and the TRANS flag is low.

### I2S slave reception sequence

The reception sequence in slave mode is similar to that in master mode. The difference between them is described below.

In slave mode, the slave has to be enabled before the external master starts the communication. The reception sequence begins when the external master sends the clock and when the I2S\_WS signal indicates a start of the data transfer. In slave mode, I2SCH is sensitive to the I2S\_WS signal coming from the external master.

In order to switch off I2S, it is mandatory to clear the I2SEN bit immediately after receiving the last RBNE.

### 19.4.6. DMA function

DMA function is the same as SPI mode. The only difference is that the CRC function is not available in I2S mode.

### 19.4.7. I2S interrupts

#### Status flags

There are four status flags implemented in the SPI\_STAT register, including TBE, RBNE, TRANS and I2SCH. The user can use them to fully monitor the state of the I2S bus.

- Transmit buffer empty flag (TBE)

This bit is set when the transmit buffer is empty, the software can write the next data to the transmit buffer by writing the SPI\_DATA register.

- Receive buffer not empty flag (RBNE)

This bit is set when receive buffer is not empty, which means that one data is received and stored in the receive buffer, and software can read the data by reading the SPI\_DATA register.

- I2S Transmitting On-Going flag (TRANS)

TRANS is a status flag to indicate whether the transfer is on-going or not. It is set and cleared by internal hardware and not controlled by software. This flag doesn't generate any interrupt.

- I2S channel side flag (I2SCH)

This flag indicates the channel side information of the current transfer and has no meaning in PCM mode. It is updated when TBE rises in transmission mode or RBNE rises in reception mode. This flag doesn't generate any interrupt.

#### Error conditions

There are three error conditions:

- Transmission Underrun Error Flag (TXURERR)

In the slave transmit mode, when the valid SCK signal starts transmitting, if the transmit buffer is empty, TXURERR will be set.

#### ■ Reception Overrun Error Flag (RXORERR)

This condition occurs when the receive buffer is full and a newly incoming data has been completely received. When overrun occurs, the data in receive buffer is not updated and the newly incoming data is lost.

#### ■ 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 enabled bits are summed up in the [Table 19-10. I2S interrupt](#).

**Table 19-10. I2S interrupt**

Flag Name	Description	Clear Method	Interrupt Enable bit
TBE	Transmit buffer empty	Write SPI_DATA register	TBEIE
RBNE	Receive buffer not empty	Read SPI_DATA register	RBNEIE
TXURERR	Transmission underrun error	Read SPI_STAT register	ERRIE
RXORERR	Reception overrun error	Read SPI_DATA register and then read SPI_STAT register.	
FERR	I2S Format Error	Read SPI_STAT register	

## 19.5. Register definition

SPI0 base address: 0x4001 3000

SPI1/I2S1 base address: 0x4000 3800

SPI2/I2S2 base address: 0x4000 3C00

### 19.5.1. Control register 0 (SPI\_CTL0)

Address offset: 0x00

Reset value: 0x0000

This register has to be accessed by 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 (TCR) When the transfer is managed by DMA, CRC value is transferred by hardware. This bit should be cleared.

In full-duplex or transmit-only mode, set this bit after the last data is written to SPI\_DATA register. In receive only mode, set this bit after the second last data is received.

11	FF16	Data frame format 0: 8-bit data frame format 1: 16-bit data frame format
10	RO	Receive only When BDEN is cleared, this bit determines the direction of transfer. 0: Full-duplex 1: Receive-only
9	SWNSSEN	NSS software mode selection 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 has an effect 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    100: PCLK/32 001: PCLK/4    101: PCLK/64 010: PCLK/8    110: PCLK/128 011: PCLK/16    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
---	------	--

### 19.5.2. Control register 1 (SPI\_CTL1)

Address offset: 0x04

Reset value: 0x0000

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								TBEIE	RBNEIE	ERRIE	TMOD	NSSP	NSSDRV	DMATEN	DMAREN
rw															

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 or the CONFERR bit or 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 Disable 1: SPI NSS Pulse Mode Enable
2	NSSDRV	Drive NSS output 0: NSS output is disabled 1: NSS output is enabled If the NSS pin is configured as output, the NSS pin is pulled low in master mode when SPI is enabled.

If the NSS pin is configured as input, the NSS pin should be pulled high in master mode, and this bit has no effect.

1	DMATEN	Transmit buffer DMA enable 0: Transmit buffer DMA is disabled 1: Transmit buffer DMA is enabled. When the TBE bit in SPI_STAT is set, it will generate a DMA request at corresponding DMA channel.
0	DMAREN	Receive buffer DMA enable 0: Receive buffer DMA is disabled 1: Receive buffer DMA is enabled. When the RBNE bit in SPI_STAT is set, it will generate a DMA request at corresponding DMA channel.

### 19.5.3. Status register (SPI\_STAT)

Address offset: 0x08

Reset value: 0x0002

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								FERR	TRANS	RXORERR	CONFERR	CRCERR	TXURERR	I2SCH	TBE	RBNE
rc_w0      r      r      r      rc_w0      r      r      r																

Bits	Fields	Descriptions
31:9	Reserved	Must be kept at reset value.
8	FERR	Format error bit 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 is able to be cleared by writing 0.
7	TRANS	Transmitting on-going bit 0: SPI or I2S is idle. 1: SPI or I2S is currently transmitting and/or receiving a frame This bit is set and cleared by hardware.
6	RXORERR	Reception overrun error bit 0: No reception overrun error occurs.

		1: Reception overrun error occurs. This bit is set by hardware and cleared by a read operation on the SPI_DATA register followed by a read access to the SPI_STAT register.
5	CONFERR	SPI configuration error bit 0: No configuration fault occurs 1: Configuration fault occurred. (In master mode, the NSS pin is pulled low in NSS hardware mode or SWNSS bit is low in NSS software mode.) This bit is set by hardware and cleared by a read or write operation on the SPI_STAT register followed by a write access to the SPI_CTL0 register. This bit is not used in I2S mode.
4	CRCERR	SPI CRC error bit 0: The SPI_RCRC value is equal to the received CRC data at last. 1: The SPI_RCRC value is not equal to the received CRC data at last. This bit is set by hardware and is able to be cleared by writing 0. This bit is not used in I2S mode.
3	TXURERR	Transmission underrun error bit 0: No transmission underrun error occurs 1: Transmission underrun error occurs This bit is set by hardware and cleared by a read operation on the SPI_STAT register. This bit is not used in SPI mode.
2	I2SCH	I2S channel side 0: The next data needs to be transmitted or the data just received is channel left 1: The next data needs to be transmitted or the data just received is channel right This bit is set and cleared by hardware. This bit is not used in SPI mode, and has no meaning in the I2S PCM mode.
1	TBE	Transmit buffer empty 0: Transmit buffer is not empty 1: Transmit buffer is empty
0	RBNE	Receive buffer not empty 0: Receive buffer is empty 1: Receive buffer is not empty

#### 19.5.4. Data register (SPI\_DATA)

Address offset: 0x0C

Reset value: 0x0000

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
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-bits. If the Data frame format is set to 16-bit data, the SPI_DATA [15:0] is used for transmission and reception, transmit buffer and receive buffer are 16-bit.</p>

### 19.5.5. CRC polynomial register (SPI\_CRCPOLY)

Address offset: 0x10

Reset value: 0x0007

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
CPR [15:0]															

rw

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	CPR[15:0]	<p>CRC polynomial register</p> <p>This register contains the CRC polynomial and it is used for CRC calculation. The default value is 0007h.</p>

### 19.5.6. RX CRC register (SPI\_RCRC)

Address offset: 0x14

Reset value: 0x0000

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
RCRC[15:0]															

r

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	RCRC[15:0]	<p>RX CRC register</p> <p>When the CRCERRN bit of SPI_CTL0 is set, the hardware computes the CRC value of the received bytes and saves them in RCR register. If the Data frame format is set to 8-bit data, CRC calculation is based on CRC8 standard, and saves the value in 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 or the SPIEN bit in SPI_CTL0 register is cleared.</p>

### 19.5.7. TX CRC register (SPI\_TCRC)

Address offset: 0x18

Reset value: 0x0000

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
TCRC[15:0]															

r

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value.
15:0	TCRC[15:0]	<p>TX CRC register</p> <p>When the CRCEN bit of SPI_CTL0 is set, the hardware computes the CRC value of the transmitted bytes and saves them in TCR register. If the Data frame format is set to 8-bit data, CRC calculation is based on CRC8 standard, and saves the</p>

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]. The hardware computes the CRC value after each transmitted bit, when the TRANS is set, a read to this register could return an intermediate value. The different frame format (LF bit of the SPI\_CTL0) will get different CRC value. This register is reset when the CRCEN bit or the SPIEN bit in SPI\_CTL0 register is cleared.

### 19.5.8. I2S control register (SPI\_I2SCTL)

Address offset: 0x1C

Reset value: 0x0000

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	I2SEL	I2SEN	I2SOPMOD[1:0]	PCMSMOD	Reserved	I2STD[1:0]	CKPL	DTLEN[1:0]	CHLEN						

rw      rw

Bits	Fields	Descriptions
31:12	Reserved	Must be kept at reset value.
11	I2SEL	I2S mode selection 0: SPI mode 1: I2S mode This bit should be configured when SPI mode or I2S mode 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 mode is disabled. This bit is not used in SPI mode.
7	PCMSMOD	PCM frame synchronization mode 0: Short frame synchronization 1: long frame synchronization

		This bit has a meaning only when PCM standard is used.
		This bit should be configured when I2S mode is disabled.
		This bit is not used in SPI mode.
6	Reserved	Must be kept at reset value.
5:4	I2SSTD[1:0]	I2S standard selection 00: I2S Phillips standard 01: MSB justified standard 10: LSB justified standard 11: PCM standard
		These bits should be configured when I2S mode is disabled.
		These bits are not used in SPI mode.
3	CKPL	Idle state clock polarity 0: The idle state of I2S_CK is low level 1: The idle state of I2S_CK is high level
		This bit should be configured when I2S mode is disabled.
		This bit is not used in SPI mode.
2:1	DTLEN[1:0]	Data length 00: 16 bits 01: 24 bits 10: 32 bits 11: Reserved
		These bits should be configured when I2S mode is disabled.
		These bits are not used in SPI mode.
0	CHLEN	Channel length 0: 16 bits 1: 32 bits
		The channel length must be equal to or greater than the data length.
		This bit should be configured when I2S mode is disabled.
		This bit is not used in SPI mode.

### 19.5.9. I2S clock prescaler register (SPI\_I2SPSC)

Address offset: 0x20

Reset value: 0x0002

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	MCKOEN	OF	DIV[7:0]

Bits	Fields	Descriptions
31:10	Reserved	Must be kept at reset value.
9	MCKOEN	I2S_MCK output enable 0: I2S_MCK output is disabled 1: I2S_MCK output is enabled This bit should be configured when I2S mode is disabled. This bit is not used in SPI mode.
8	OF	Odd factor for the prescaler 0: Real divider value is DIV * 2 1: Real divider value is DIV * 2 + 1 This bit should be configured when I2S mode is disabled. This bit is not used in SPI mode.
7:0	DIV[7:0]	Dividing factor for the prescaler Real divider value is DIV * 2 + OF. DIV must not be 0. These bits should be configured when I2S mode is disabled. These bits are not used in SPI mode.

### 19.5.10. Quad-SPI mode control register (SPI\_QCTL) of SPI0

Address offset: 0x80

Reset value: 0x0000

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
													IO23_DRV	QRD	QMOD

Bits	Fields	Descriptions
31:3	Reserved	Must be kept at reset value.
2	IO23_DRV	Drive IO2 and IO3 enable 0: IO2 and IO3 are not driven in single wire mode 1: IO2 and IO3 are driven to high in single wire mode This bit is only available in SPI0.
1	QRD	Quad-SPI mode read select.

---

		0: SPI is in quad wire write mode 1: SPI is in quad wire read mode This bit should be only be configured when SPI is not busy (TRANS bit cleared) This bit is only available in SPI0.
0	QMOD	Quad-SPI mode enable 0: SPI is in single wire mode 1: SPI is in Quad-SPI mode This bit should only be configured when SPI is not busy (TRANS bit cleared). This bit is only available in SPI0.

## 20. Secure digital input/output interface (SDIO)

### 20.1. Introduction

The secure digital input/output interface (SDIO) defines the SD, SD I/O, MMC and CE-ATA card host interface, which provides command/data transfer between the AHB system bus and SD memory cards, SD I/O cards, Multimedia Card (MMC) and CE-ATA devices.

The supported SD memory card and SD I/O card system specifications are defined in the SD card Association website at [www.sdcard.org](http://www.sdcard.org).

The supported Multimedia Card system specifications are defined through the Multimedia Card Association website at [www.jedec.org](http://www.jedec.org), published by the JEDEC SOLID STATE TECHNOLOGY ASSOCIATION.

The supported CE-ATA system specifications are defined through the CE-ATA workgroup website at [www.ce-ata.org](http://www.ce-ata.org).

### 20.2. Main features

The SDIO features include the following:

- **MMC:** Full support for Multimedia Card System Specification Version 4.2(and previous versions) Card and three different data bus modes: 1-bit (default), 4-bit and 8-bit.
- **SD Card:** Full support for *SD Memory Card Specifications Version 2.0*.
- **SD I/O:** Full support for *SD I/O Card Specification Version 2.0* card and two different data bus modes: 1-bit (default) and 4-bit.
- **CE-ATA:** Full compliance with *CE-ATA digital protocol Version 1.1*.
- 48MHz data transfer frequency and 8-bit data transfer mode.
- Interrupt and DMA request to processor.
- Completion Signal enables and disable feature (CE-ATA).

**Note:** SDIO supports only one SD, SD I/O, MMC4.2 card or CE-ATA device at any one time and a stack of MMC 4.1 or previous.

## 20.3. SDIO bus topology

After a power-on reset, the host must initialize the card by a special message-based bus protocol.

Each message is represented by one of the following tokens:

**Command:** a command is a token which starts an operation. A command is sent from the host to a card. A command is transferred serially on the CMD line.

**Response:** a response is a token which is sent from the card to the host as an answer to a previously received command. A response is transferred serially on the CMD line.

**Data:** data can be transferred from the card to the host or vice versa. Data is transferred via the data lines. The number of data lines used for the data transfer can be 1(DAT0), 4(DAT0-DAT3) or 8(DAT0-DAT7).

The structure of commands, responses and data blocks is described in [Card functional description](#). One data transfer is a bus operation.

There are different types of operations. Addressed operations always contain a command and a response token. In addition, some operations have a data token; the others transfer their information directly within the command or response structure. In this case no data token is present in an operation. The bits on the DAT0-DAT7 and CMD lines are transferred synchronous to the host clock.

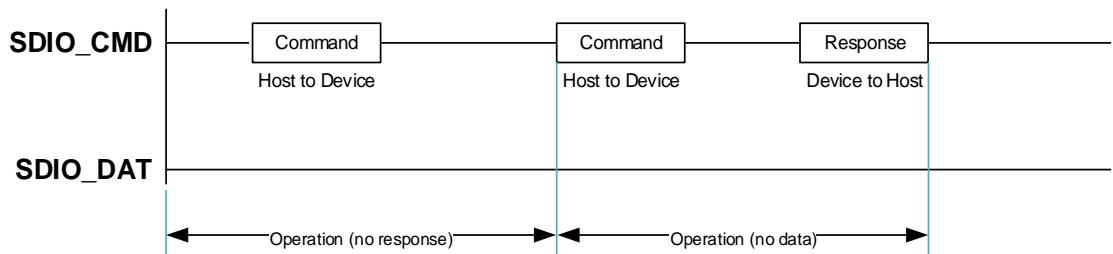
Two types of data transfer commands are defined:

- Stream commands: These commands initiate a continuous data stream; they are terminated only when a stop command follows on the CMD line. This mode reduces the command overhead to an absolute minimum (only MMC supports).
- Block-oriented commands: These commands send a data block successfully by CRC bits. Both read and write operations allow either single or multiple block transmission. A multiple block transmission is terminated when a stop command follows on the CMD line similarly to the sequential read.

The basic transaction on the bus is the command/response transaction (refer to [Figure 20-1. SDIO “no response” and “no data” operations](#)). This type of bus transaction transfers their information directly within the command or response structure. In addition, some operations have a data token. Data transfers to/from the Card/Device are done in

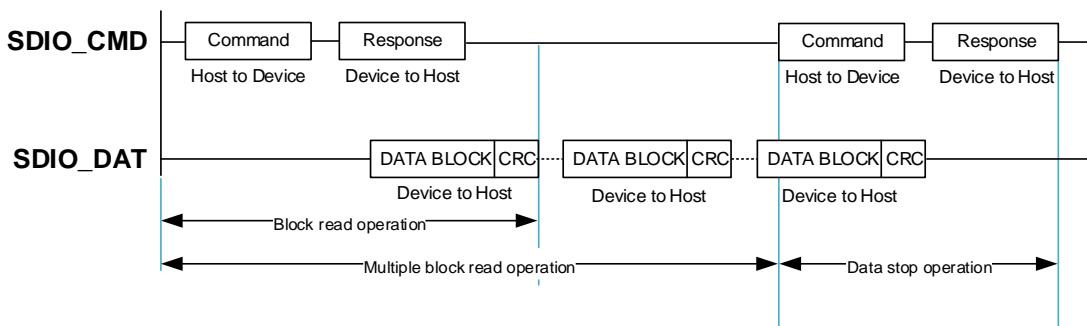
blocks.

**Figure 20-1. SDIO “no response” and “no data” operations**

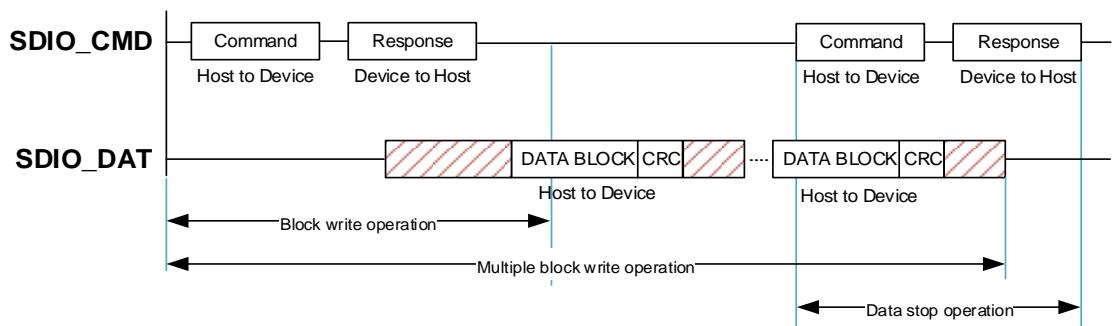


Note that the Multiple Block operation mode is faster than Single Block operation. A multiple block transmission is terminated when a stop command follows on the CMD line. Data transfer can be configured by the host to use single or multiple data lines. [Figure 20-2. SDIO multiple blocks read operation](#) is the multiple blocks read operation and [Figure 20-3. SDIO multiple blocks write operation](#) is the multiple block write operation. The block write operation uses a simple busy signal of the write operation duration on the data (DAT0) line. CE-ATA device has an optional busy before it is ready to receive the data.

**Figure 20-2. SDIO multiple blocks read operation**



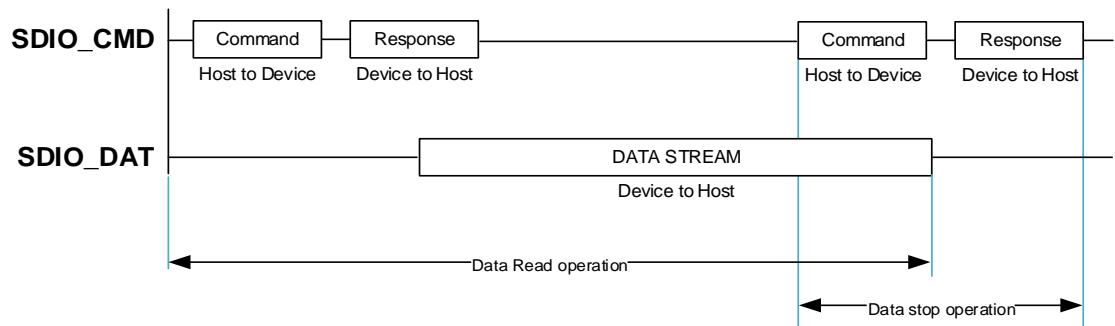
**Figure 20-3. SDIO multiple blocks write operation**



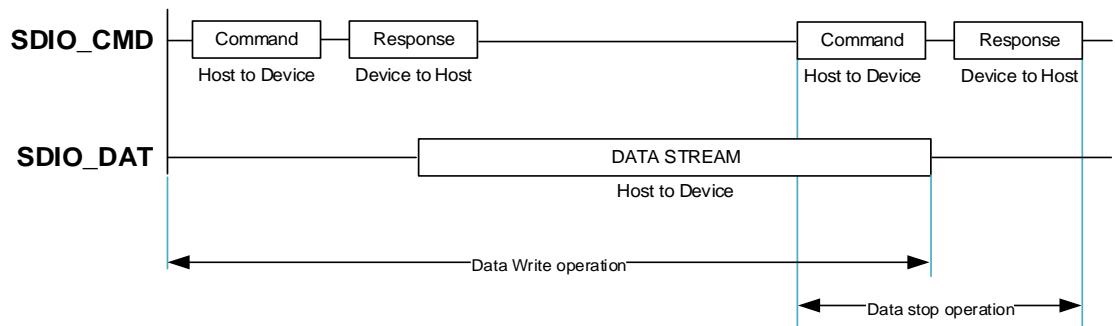
Data transfers to/from SD memory cards, SD I/O cards (both IO only card and combo card) and CE-ATA device are done in data blocks. Data transfers to/from MMC are done in data blocks or streams. [Figure 20-4. SDIO sequential read operation](#) and [Figure 20-5. SDIO](#)

[sequential write operation](#) are the stream read and write operation.

**Figure 20-4. SDIO sequential read operation**



**Figure 20-5. SDIO sequential write operation**

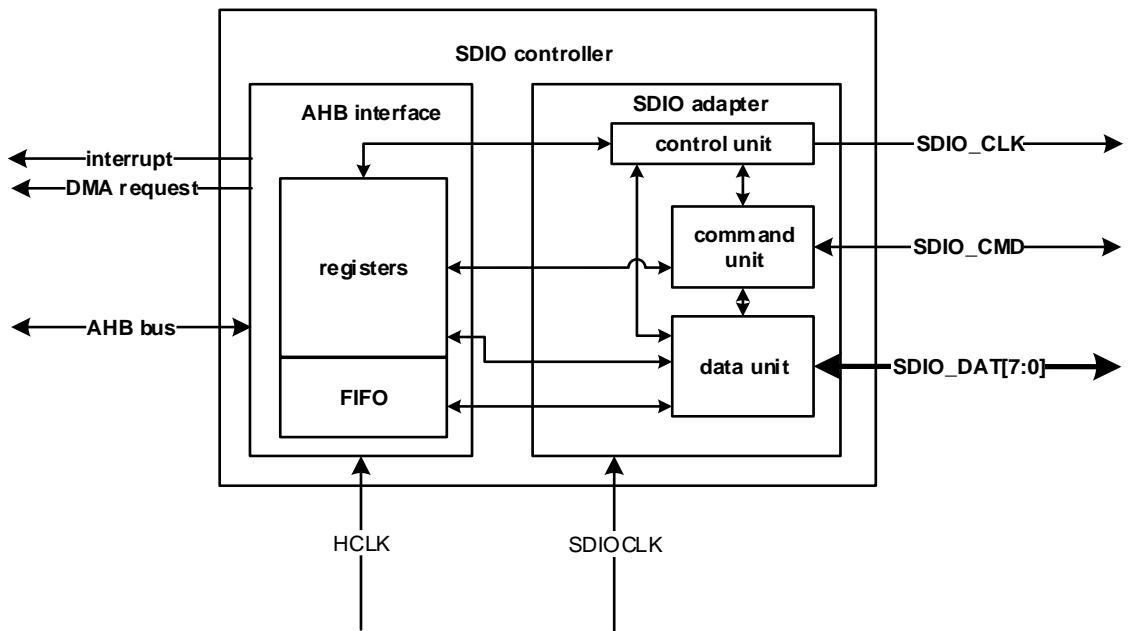


## 20.4. SDIO functional description

The following figure shows the SDIO structure. There have two main parts:

- The SDIO adapter block consists of control unit which manage clock, command unit which manage command transfer, data unit which manage data transfer.
- The AHB interface block contains access registers by AHB bus, contains FIFO unit which is data FIFO used for data transfer, and generates interrupt and DMA request signals.

Figure 20-6. SDIO block diagram



#### 20.4.1. SDIO adapter

The SDIO adapter contains control unit, command unit and data unit, and generates signals to cards. The signals are described below:

**SDIO\_CLK:** The SDIO\_CLK is the clock provided to the card. Each cycle of this signal directs a one bit transfer on the command line (SDIO\_CMD) and on all the data lines (SDIO\_DAT). The SDIO\_CLK frequency can vary between 0 MHz and 20 MHz for a Multimedia Card V3.31, between 0 and 48 MHz for a Multimedia Card V4.2, or between 0 and 25 MHz for an SD/SD I/O card.

The SDIO uses two clock signals: SDIO adapter clock (SDIOCLK = HCLK) and AHB bus clock (HCLK).

**SDIO\_CMD:** This signal is a bidirectional command channel used for card initialization and transfer of commands. Commands are sent from the SDIO controller to the card and responses are sent from the card to the host. The CMD signal has two operation modes: open-drain for initialization (only for MMC3.31 or previous), and push-pull for command transfer (SD/SD I/O card MMC4.2 use push-pull drivers also for initialization).

**SDIO\_DAT[7:0]:** These are bidirectional data channels. The DAT signals operate in push-pull mode. Only the card or the host is driving these signals at a time. By default, after power up or reset, only DAT0 is used for data transfer. A wider data bus can be configured for data transfer, using either DAT0-DAT3 or DAT0-DAT7 (just for MMC4.2), by the SDIO controller. The SDIO includes internal pull-ups for data lines DAT1-DAT7. Right after entering to the 4-bit mode the card disconnects the internal pull-ups of lines DAT1 and DAT2 (DAT3 internal pull-up is left connected due to the SPI mode CS usage). Correspondingly right after entering

to the 8-bit mode the card disconnects the internal pull-ups of lines DAT1, DAT2 and DAT4-DAT7.

**Table 20-1. SDIO I/O definitions**

Pin function	Direction	Description
SDIO_CLK	O	SD/SD I/O /MMC clock
SDIO_CMD	I/O	Command input/output
SDIO_DAT[7:0]	I/O	Data input/output for data lines DAT[7:0]

The SDIO adapter is an interface to SD, SD I/O, MMC and CE-ATA. It consists of three subunits:

### Control unit

The control unit contains the power management functions and the clock management functions for the memory card clock. The power management is controlled by SDIO\_PWRCTL register which implements power off or power on. The power saving mode configured by setting CLKPWRSAV bit in SDIO\_CLKCTL register, which implements close the SDIO\_CLK when the bus is idle. The clock management generates SDIO\_CLK to card. The SDIO\_CLK is generated by a divider of SDIOCLK when CLKBYP bit in SDIO\_CLKCTL register is 0, or directly SDIOCLK when CLKBYP bit in SDIO\_CLKCTL register is 1.

The Hardware clock control is enabled by setting HWCLKEN in SDIO\_CLKCTL register. This functionality is used to avoid FIFO underrun and overrun errors by hardware control the SDIO\_CLK on/off depending on the system bus is very busy or not. When the FIFO cannot receive or transmit data, the host will stop the SDIO\_CLK and freeze SDIO state machines to avoid the corresponded error. Only state machines are frozen, the AHB interface is still alive. So, the FIFO can access by AHB bus.

### Command unit

The command unit implements command transfer to the card. The data transfer flow is controlled by Command State Machine (CSM). After a write operation to SDIO\_CMDCTL register and CSMEN in SDIO\_CMDCTL register is 1, the command transfer starts. It firstly sends a command to the card. The command contains 48 bits send by SDIO\_CMD signal which sends 1 bits to card at one SDIO\_CLK. The 48 bits command contains 1 bit Start bit, 1 bit Transmission bit, 6 bits command index defined by CMDIDX bits in SDIO\_CMDCTL register, 32 bits argument defined in SDIO\_CMDAGMT register, 7 bits CRC, and 1 bit end bit. Then receive response from the card if CMDRESP in SDIO\_CMDCTL register is not 0b00/0b10. There are short response which have 48 bits or long response which have 136 bits. The response stores in SDIO\_RESP0 - SDIO\_RESP3 registers. The command unit also generates the command status flags defined in SDIO\_STAT register.

### Command state machine

CS_Idle	After reset, ready to send command.
---------	-------------------------------------

1.CSM enabled and WAITDEND enabled	→	CS_Pend
2.CSM enabled and WAITDEND disabled	→	CS_Send
3.CSM disabled	→	CS_Idle
<b>Note:</b> The state machine remains in the Idle state for at least eight SDIO_CLK periods to meet the Ncc and Nrc timing constraints. Ncc is the minimum delay between two host commands, and Nrc is the minimum delay between the host command and the response.		

CS_Pend	Waits for the end of data transfer.
1.The data transfer complete	→ CS_Send
2.CSM disabled	→ CS_Idle

CS_Send	Sending the command.
1.The command transmitted has response	→ CS_Wait
2.The command transmitted doesn't have response	→ CS_Idle
3.CSM disabled	→ CS_Idle

CS_Wait	Wait for the start bit of the response.
1.Receive the response(detected the start bit)	→ CS_Receive
2.Timeout is reached without receiving the response	→ CS_Idle
3.CSM disabled	→ CS_Idle
<b>Note:</b> The command timeout has a fixed value of 64 SDIO_CLK clock periods.	

CS_Receive	Receive the response and check the CRC.
1.Response Received in CE-ATA mode and interrupt disabled and wait for CE-ATA Command Completion signal enabled	→ CS_Waitcompl
2.Response Received in CE-ATA mode and interrupt disabled and wait for CE-ATA Command Completion signal disabled	→ CS_Pend
3.CSM disabled	→ CS_Idle
4.Response received	→ CS_Idle
5.Command CRC failed	→ CS_Idle

CS_Waitcompl	Wait for the Command Completion signal.
1.CE-ATA Command Completion signal received	→ CS_Idle
2.CSM disabled	→ CS_Idle
3.Command CRC failed	→ CS_Idle

## Data unit

The data unit performs data transfers to and from cards. The data transfer uses

SDIO\_DAT[7:0] signals when 8-bits data width (BUSMODE bits in SDIO\_CLKCTL register is 0b10), use SDIO\_DAT[3:0] signals when 4-bits data width (BUSMODE bits in SDIO\_CLKCTL register is 0b01), or SDIO\_DAT[0] signal when 1-bit data width (BUSMODE bits in SDIO\_CLKCTL register is 0b00). The data transfer flow is controlled by Date State Machine (DSM). After a write operation to SDIO\_DATACTL register and DATAEN in SDIO\_DATACTL register is 1, the data transfer starts. It sends data to card when DATADIR in SDIO\_DATACTL register is 0, or receive data from card when DATADIR in SDIO\_DATACTL register is 1. The data unit also generates the data status flags defined in SDIO\_STAT register.

### Data state machine

<b>DS_Idle</b>	The data unit is inactive, waiting for send and receive.	
1.DSM enabled and data transfer direction is from host to card	→	DS_WaitS
2.DSM enabled and data transfer direction is from card to host	→	DS_WaitR
3.DSM enabled and Read Wait Started and SD I/O mode enabled	→	DS_Readwait

<b>DS_WaitS</b>	Wait until the data FIFO empty flag is deasserted or data transfer ended.	
1.Data transfer ended	→	DS_Idle
2.DSM disabled	→	DS_Idle
3.Data FIFO empty flag is deasserted	→	DS_Send

<b>DS_Send</b>	Transmit data to the card.	
1.Data block transmitted	→	DS_Busy
2.DSM disabled	→	DS_Idle
3.Data FIFO underrun error occurs	→	DS_Idle
4. Internal CRC error	→	DS_Idle

<b>DS_Busy</b>	Waits for the CRC status flag.	
1.Receive a positive CRC status	→	DS_WaitS
2.Receive a negative CRC status	→	DS_Idle
3.DSM disabled	→	DS_Idle
4.Timeout occurs	→	DS_Idle
<b>Note:</b> The command timeout programmed in the data timer register (SDIO_DATATO).		

<b>DS_WaitR</b>	Wait for the start bit of the receive data.	
1.Data receive ended	→	DS_Idle
2.DSM disabled	→	DS_Idle

3.Data timeout reached	→	DS_Idle
4.Receives a start bit before timeout	→	DS_Receive
<b>Note:</b> The command timeout programmed in the data timer register (SDIO_DATATO).		

DS_Receive	Receive data from the card and write it to the data FIFO.	
1.Data block received	→	DS_WaitR
2.Data transfer ended	→	DS_WaitR
3.Data FIFO overrun error occurs	→	DS_Idle
4.Data received and Read Wait Started and SD I/O mode enabled	→	DS_Readwait
5.DSM disabled or CRC fails	→	DS_Idle

DS_Readwait	Wait for the read wait stop command.	
1.ReadWait stop enabled	→	DS_WaitR
2.DSM disabled	→	DS_Idle

#### 20.4.2. AHB interface

The AHB interface implements access to SDIO registers, data FIFO and generates interrupt and DMA request. It includes a data FIFO unit, registers unit, and the interrupt / DMA logic.

The interrupt logic generates interrupt when at least one of the selected status flags is high. An interrupt enable register is provided to allow the logic to generate a corresponding interrupt.

The DMA interface provides a method for fast data transfers between the SDIO data FIFO and memory. The following example describes how to implement this method:

1. Complete the card identification process
2. Increase the SDIO\_CLK frequency
3. Send CMD7 to select the card and configure the bus width
4. Configure the DMA1 as follows:

Enable DMA1 controller and clear any pending interrupts. Configure the DMA1\_Channel3 source address register with the memory base address and DMA1\_Channel3 destination address register with the SDIO\_FIFO register address. Program DMA1\_Channel3 control register (memory increment, not peripheral increment, peripheral and source width is word size, M2M disable).

5. Write block to card as follows:

Write the data size in bytes in the SDIO\_DATALEN register. Write the block size in bytes (BLKSZ) in the SDIO\_DATACTL register; the host sends data in blocks of size BLKSZ each. Program SDIO\_CMDAGMT register with the data address, where data should be written. Program the SDIO command control register (SDIO\_CMDCTL): CMDIDX with 24, CMDRESP

with 1 (SDIO card host waits for a short response); CSMEN with ‘1’ (enable to send a command). Other fields are their reset value.

When the CMDRECV flag is set, program the SDIO data control register (SDIO\_DATACTL): DATAEN with 1 (enable to send data); DATADIR with 0 (from controller to card); TRANSMOD with 0 (block data transfer); DMAEN with 1 (DMA enabled); BLKSZ with 0x9 (512 bytes). Other bits don’t care.

Wait for DTBLKEND flag is set. Check that no channels are still enabled by polling the DMA Interrupt Flag register.

It consists the following subunits:

### Register unit

The register unit which contains all system registers generates the signals to control the communication between the controller and card.

### Data FIFO

The data FIFO unit has a data buffer, uses as transmit and receive FIFO. The FIFO contains a 32-bit wide, 32-word deep data buffer. The transmit FIFO is used when write data to card and TXRUN in SDIO\_STAT register is 1. The data to be transferred is written to transmit FIFO by AHB bus, the data unit in SDIO adapter read data from transmit FIFO, and then send the data to card. The receive FIFO is used when read data from card and RXRUN in SDIO\_STAT register is 1. The data to be transferred is read from the card and then write to receive FIFO. The data in receive FIFO is read to AHB bus when needed. This unit also generates FIFO flags in SDIO\_STAT registers.

## 20.5. Card functional description

### 20.5.1. Card registers

Within the card interface registers are defined: OCR, CID, CSD, EXT\_CSD, RCA, DSR and SCR. These can be accessed only by corresponding commands. The OCR, CID, CSD and SCR registers carry the card/content specific information, while the RCA and DSR registers are configuration registers storing actual configuration parameters. The EXT\_CSD register carries both, card specific information and actual configuration parameters. For specific information, please refer to the relevant specifications.

**OCR register:** The 32-bit operation conditions register (OCR) stores the  $V_{DD}$  voltage profile of the card and the access mode indication (MMC). In addition, this register includes a status information bit. This status bit is set if the card power up procedure has been finished. The register is a little different between MMC and SD card. The host can use CMD1 (MMC), ACMD41 (SD memory), CMD5 (SD I/O) to get the content of this register.

**CID register:** The Card Identification (CID) register is 128 bits wide. It contains the card identification information used during the card identification phase. Every individual Read/Write (RW) card shall have a unique identification number. The host can use CMD2 and CMD10 to get the content of this register.

**CSD register:** The Card-Specific Data register provides information regarding access to the card contents. The CSD defines the data format, error correction type, maximum data access time, data transfer speed, whether the DSR register can be used, etc. The programmable part of the register can be changed by CMD27. The host can use CMD9 to get the content of this register.

**Extended CSD Register:** Just MMC4.2 has this register. The Extended CSD register defines the card properties and selected modes. It is 512 bytes long. The most significant 320 bytes are the Properties segment, which defines the card capabilities and cannot be modified by the host. The lower 192 bytes are the Modes segment, which defines the configuration the card is working in. These modes can be changed by the host by means of the SWITCH command. The host can use CMD8 (just MMC supports this command) to get the content of this register.

**RCA register:** The writable 16-bit relative card address register carries the card address that is published by the card during the card identification. This address is used for the addressed host-card communication after the card identification procedure. The host can use CMD3 to ask the card to publish a new relative address (RCA).

**Note:** The default value of the RCA register is 0x0001(MMC) or 0x0000(SD/SD I/O). The default value is reserved to set all cards into the Stand-by State with CMD7.

**DSR register (Optional):** The 16-bit driver stage register can be optionally used to improve the bus performance for extended operating conditions (depending on parameters like bus length, transfer rate or number of cards). The CSD register carries the information about the DSR register usage. The default value of the DSR register is 0x404. The host can use CMD4 to get the content of this register.

**SCR register:** Just SD/SD I/O (if has memory port) have this register. In addition to the CSD register, there is another configuration register named SD CARD Configuration Register (SCR), which is only for SD card. SCR provides information on the SD Memory Card's special features that were configured into the given card. The size of SCR register is 64 bits. This register shall be set in the factory by the SD Memory Card manufacturer. The host can use ACMD51 to get the content of this register.

## 20.5.2. Commands

### Commands types

There are four kinds of commands defined to control the Card:

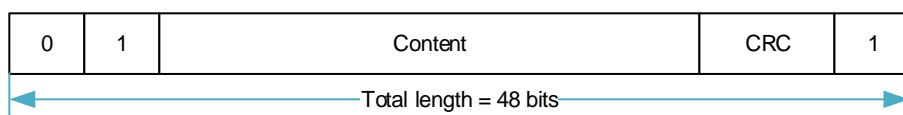
- Broadcast commands (bc), no response

- Broadcast commands with response (bcr) response from all cards simultaneously
- Addressed (point-to-point) commands (ac) no data transfer on DAT
- Addressed (point-to-point) data transfer commands (adtc) data transfer on DAT

### **Command format**

All commands have a fixed code length of 48 bits, as show in [Figure 20-7. Command Token Format](#), needing a transmission time of 1.92 $\mu$ s (25 MHz) 0.96 $\mu$ s (50 MHz) and 0.92us (52 MHz).

**Figure 20-7. Command Token Format**



**Table 20-2. Command format**

Bit position	47	46	[45:40]	[39:8]	[7:1]	0
Width	1	1	6	32	7	1
Value	'0'	'1'	x	x	x	'1'
Description	start bit	transmission bit	command index	argument	CRC7	end bit

A command always starts with a start bit (always 0), followed by the bit indicating the direction of transmission (host = 1). The next 6 bits indicate the index of the command, this value being interpreted as a binary coded number (between 0 and 63). Some commands need an argument (e.g. an address), which is coded by 32 bits. A value denoted by 'x' in the table above indicates this variable is dependent on the command. All commands are protected by a CRC7. Every command code word is terminated by the end bit (always 1).

### **Command classes**

The command set of the Card system is divided into several classes (See [Table 20-3. Card command classes \(CCCs\)](#)). Each class supports a set of card functionalities. [Table 20-3. Card command classes \(CCCs\)](#) determines the setting of CCC from the card supported commands.

For SD cards, Class 0, 2, 4, 5 and 8 are mandatory and shall be supported. Class 7 except CMD40 is mandatory for SDHC. The other classes are optional. The supported Card Command Classes (CCC) are coded as a parameter in the card specific data (CSD) register of each card, providing the host with information on how to access the card.

For MMC cards, Class 0 is mandatory and shall be supported. The other classes are either mandatory only for specific card types or optional. By using different classes, several configurations can be chosen (e.g. a block writable card or a stream readable card). The supported Card Command Classes (CCC) are coded as a parameter in the card specific data (CSD) register of each card, providing the host with information on how to access the card.

For CE-ATA device, the device shall support the MMC commands required to achieve the transfer state during device initialization. Other interface configuration settings, such as bus width, may require additional MMC commands also be supported. See the MMC reference. CE-ATA makes use of the following MMC commands: CMD0 - GO\_IDLE\_STATE, CMD12 - STOP\_TRANSMISSION, CMD39 - FAST\_IO, CMD60 - RW\_MULTIPLE\_REGISTER, CMD61 - RW\_MULTIPLE\_BLOCK. GO\_IDLE\_STATE (CMD0), STOP\_TRANSMISSION (CMD12), and FAST\_IO (CMD39) are as defined in the MMC reference. RW\_MULTIPLE\_REGISTER (CMD60) and RW\_MULTIPLE\_BLOCK (CMD61) are MMC commands defined by CE-ATA.

**Table 20-3. Card command classes (CCCs)**

	<b>Card command class(CCC)</b>	0	1	2	3	4	5	6	7	8	9	10	11
<b>Supported command</b>	<b>Class description</b>	basic	Stream read	Block read	Stream write	Block write	erase	write protection	Lock card	application specific	I/O mode	switch	reserved
<b>CMD0</b>	M	+											
<b>CMD1</b>	M	+											
<b>CMD2</b>	M	+											
<b>CMD3</b>	M	+											
<b>CMD4</b>	M	+											
<b>CMD5</b>	O												+
<b>CMD6</b>	M												+
<b>CMD7</b>	M	+											
<b>CMD8</b>	M	+											
<b>CMD9</b>	M	+											
<b>CMD10</b>	M	+											
<b>CMD11</b>	M		+										
<b>CMD12</b>	M	+											
<b>CMD13</b>	M	+											
<b>CMD14</b>	M	+											
<b>CMD15</b>	M	+											
<b>CMD16</b>	M			+		+				+			
<b>CMD17</b>	M			+									
<b>CMD18</b>	M			+									
<b>CMD19</b>	M	+											
<b>CMD20</b>	M				+								
<b>CMD23</b>	M			+		+							
<b>CMD24</b>	M						+						

CMD25	M					+						
CMD26	M					+						
CMD27	M					+						
CMD28	M						+					
CMD29	M						+					
CMD30	M						+					
CMD32	M					+						
CMD33	M					+						
CMD34	O											+
CMD35	O											+
CMD36	O											+
CMD37	O											+
CMD38	M					+						
CMD39												+
CMD40												+
CMD42							+					
CMD50	O											+
CMD52	O											+
CMD53	O											+
CMD55	M						+					
CMD56	M						+					
CMD57	O											+
CMD60	M							+				
CMD61	M							+				
ACMD6	M							+				
ACMD13	M							+				
ACMD22	M							+				
ACMD23	M							+				
ACMD41	M							+				
ACMD42	M								+			
ACMD51	M								+			

**Note:** 1.CMD1, CMD11, CMD14, CMD19, CMD20, CMD23, CMD26, CMD39 and CMD40 are only available for MMC.CMD5, CMD32-34, CMD50, CMD52, CMD53, CMD57 and ACMDx are only available for SD card. CMD60, CMD61 are only available for CE-ATA device.

2. All the ACMDs shall be preceded with APP\_CMD command (CMD55).
3. CMD8 has different meaning for MMC and SD memory.

### Detailed command description

The following tables describe in detail all bus commands. The responses R1-R7 are defined in [Responses](#). The registers CID, CSD and DSR are described in [Card registers](#). The card shall ignore stuff bits and reserved bits in an argument.

**Table 20-4. Basic commands (class 0)**

<b>Cmd index</b>	<b>type</b>	<b>argument</b>	<b>Response format</b>	<b>Abbreviation</b>	<b>Description</b>
CMD0	bc	[31:0] stuff bits	-	GO_IDLE_STATE	Resets all cards to idle state
CMD1	bc	[31:0] OCR without busy	R3	SEND_OP_COND	Asks the card, in idle state, to send its Operating Conditions Register contents in the response on the CMD line.
CMD2	bcr	[31:0] stuff bits	R2	ALL_SEND_CID	Asks any card to send the CID numbers on the CMD line (any card that is connected to the host will respond)
CMD3	bcr	[31:0] stuff bits	R6	SEND_RELATIVE_ADDR	Ask the card to publish a new relative address (RCA)
CMD4	bc	[31:16] DSR [15:0] stuff bits	-	SET_DSR	Programs the DSR of all cards
CMD5	bcr	[31:25]reserved bits [24]S18R [23:0] I/O OCR	R4	IO_SEND_OP_COND	Only for I/O cards. It is similar to the operation of ACMD41 for SD memory cards, used to inquire about the voltage range needed by the I/O card.
CMD6	ac	[31:26] Set to 0 [25:24] Access [23:16] Index [15:8] Value [7:3] Set to 0 [2:0] Cmd Set	R1b	SWITCH	Only for MMC. Switches the mode of operation of the selected card or modifies the EXT_CSD registers.
CMD7	ac	[31:16] RCA [15:0] stuff bits	R1b	SELECT/DESELECT_CARD	Command toggles a card between the stand-by and transfer states or between the programming and disconnects states. In both cases the card is selected by its own relative address and gets deselected by any other address; address 0 deselects the card.
CMD8	bcr	[31:12]reserved bits [11:8]supply voltage(VHS) [7:0]check pattern	R7	SEND_IF_COND	Sends SD Memory Card interface condition, which includes host supply voltage information and asks the card whether card supports voltage. Reserved bits shall be set to '0'.

CMD8	adtc	[31:0] stuff bits	R1	SEND_EXT_CSD	For MMC only. The card sends its EXT_CSD register as a block of data.
CMD9	ac	[31:16] RCA [15:0] stuff bits	R2	SEND_CSD	Addressed card sends its card-specific data (CSD) on the CMD line.
CMD10	ac	[31:16] RCA [15:0] stuff bits	R2	SEND_CID	Addressed card sends its card identification (CID) on CMD the line.
CMD12	ac	[31:0] stuff bits	R1b	STOP TRANSMISSION	Forces the card to stop transmission
CMD13	ac	[31:16] RCA [15:0] stuff bits	R1	SEND_STATUS	Addressed card sends its status register.
CMD14	adtc	[31:0] stuff bits	R1	BUSTEST_R	A host reads the reversed bus testing data pattern from a card.
CMD15	ac	[31:16] RCA [15:0] reserved bits	-	GO_INACTIVE_ STATE	Sends an addressed card into the Inactive State. This command is used when the host explicitly wants to deactivate a card.
CMD19	adtc	[31:0] stuff bits	R1	BUSTEST_W	A host sends the bus test data pattern to a card.

**Table 20-5. Block-Oriented read commands (class 2)**

Cmd index	type	argument	Response format	Abbreviation	Description
CMD16	ac	[31:0] block length	R1	SET_BLOCKLEN	<p>In the case of a Standard Capacity SD and MMC, this command sets the block length (in bytes) for all following block commands (read, write, lock). Default is 512 Bytes. Set length is valid for memory access commands only if partial block read operation are allowed in CSD.</p> <p>In the case of a High Capacity SD Memory Card, block length set by CMD16 command does not affect the memory read and write commands. Always 512 Bytes fixed block length is used. In both</p>

					cases, if block length is set larger than 512Bytes, the card sets the BLOCK_LEN_ERROR bit.
CMD17	adtc	[31:0] data address	R1	READ_SINGLE_BLOCK	In the case of a Standard Capacity SD and MMC, this command reads a block of the size selected by the SET_BLOCKLEN command. In the case of a High Capacity Card, block length is fixed 512 Bytes regardless of the SET_BLOCKLEN command.
CMD18	adtc	[31:0] data address	R1	READ_MULTIPLE_BLOCK	Continuously transfers data blocks from card to host until interrupted by a STOP_TRANSMISSION command. Block length is specified the same as READ_SINGLE_BLOCK command.
<b>Note:</b> The transferred data must not cross a physical block boundary, unless READ_BLK_MISALIGN is set in the CSD register					

**Table 20-6. Stream read commands (class 1) and stream write commands (class 3)**

Cmd index	type	argument	Response format	Abbreviation	Description
CMD11	adtc	[31:0] data address	R1	READ_DAT_UNTIL_STOP	Reads data stream from the card, starting at the given address, until a STOP_TRANSMISSION follows.
CMD20	adtc	[31:0] data address	R1	WRITE_DAT_UNTIL_STOP	Writes data stream from the host, starting at the given address, until a STOP_TRANSMISSION follows.
<b>Note:</b> The transferred data must not cross a physical block boundary, unless READ_BLK_MISALIGN is set in the CSD register					

**Table 20-7. Block-Oriented write commands (class 4)**

Cmd index	type	argument	Response format	Abbreviation	Description
CMD16	ac	[31:0] block length	R1	SET_BLOCKLEN	See description in <a href="#">Table 20-5. Block-Oriented read commands (class 2)</a> .

<b>Cmd index</b>	<b>type</b>	<b>argument</b>	<b>Response format</b>	<b>Abbreviation</b>	<b>Description</b>
CMD23	ac	[31:16] set to 0 [15:0] number of blocks	R1	SET_BLOCK_COUNT	Defines the number of blocks which are going to be transferred in the immediately succeeding multiple block read or write command. If the argument is all 0s, the subsequent read/write operation will be open-ended.
CMD24	adtc	[31:0] data address	R1	WRITE_BLOCK	In the case of a Standard Capacity SD, this command writes a block of the size selected by the SET_BLOCKLEN command. In the case of a SDHC, block length is fixed 512 Bytes regardless of the SET_BLOCKLEN command.
CMD25	adtc	[31:0] data address	R1	WRITE_MULTIPL E_BLOCK	Continuously writes blocks of data until a STOP_TRANSMISSION follows. Block length is specified the same as WRITE_BLOCK command.
CMD26	adtc	[31:0] stuff bits	R1	PROGRAM_CID	Programming of the card identification register. This command shall be issued only once. The card contains hardware to prevent this operation after the first programming. Normally this command is reserved for the manufacturer.
CMD27	adtc	[31:0] stuff bits	R1	PROGRAM_CSD	Programming of the programmable bits of the CSD.
<b>Note:</b> 1.The data transferred shall not cross a physical block boundary unless WRITE_BLK_MISALIGN is set in the CSD. In the case that write partial blocks is not supported, then the block length=default block length (given in CSD). 2. Data address is in byte units in a Standard Capacity SD Memory Card and in block (512 Byte) units in a High Capacity SD Memory Card.					

**Table 20-8. Erase commands (class 5)**

<b>Cmd</b>	<b>type</b>	<b>argument</b>	<b>Response</b>	<b>Abbreviation</b>	<b>Description</b>
------------	-------------	-----------------	-----------------	---------------------	--------------------

<b>index</b>			<b>format</b>		
CMD32	ac	[31:0] data address	R1	ERASE_WR_BLK _START	Sets the address of the first write block to be erased.(SD)
CMD33	ac	[31:0] data address	R1	ERASE_WR_BLK _END	Sets the address of the last write block of the continuous range to be erased.(SD)
CMD35	ac	[31:0]data address	R1	ERASE_GROUP_ START	Sets the address of the first erase group within a range to be selected for erase.(MMC)
CMD36	ac	[31:0]data address	R1	ERASE_GROUP_ END	Sets the address of the last erase group within a continuous range to be selected for erase.(MMC)
CMD38	ac	[31:0] stuff bits	R1b	ERASE	Erases all previously selected write blocks.
<b>Note:</b> 1.CMD34 and CMD37 are reserved in order to maintain backwards compatibility with older versions of the MMC. 2. Data address is in byte units in a Standard Capacity SD Memory Card and in block (512 Byte) units in a High Capacity SD Memory Card.					

**Table 20-9. Block oriented write protection commands (class 6)**

<b>Cmd index</b>	<b>type</b>	<b>argument</b>	<b>Response format</b>	<b>Abbreviation</b>	<b>Description</b>
CMD28	ac	[31:0] data address	R1b	SET_WRITE_PRO T	If the card has write protection features, this command sets the write protection bit of the addressed group. The properties of write protection are coded in the card specific data (WP_GRP_SIZE).  A High Capacity SD Memory Card does not support this command.
CMD29	ac	[31:0] data address	R1b	CLR_WRITE_PRO T	If the card provides write protection features, this command clears the write protection bit of the addressed group.
CMD30	adtc	[31:0] write protect data address	R1	SEND_WRITE_PR OT	If the card provides write protection features, this command asks the card to send the status of the write protection bits.

**Note:** 1. High Capacity SD Memory Card does not support these three commands.

**Table 20-10. Lock card (class 7)**

Cmd index	type	argument	Response format	Abbreviation	Description
CMD16	ac	[31:0] block length	R1	SET_BLOCK_LEN	See description in <a href="#">Table 20-5. Block-Oriented read commands (class 2).</a>
CMD42	adtc	[31:0] Reserved bits (Set all 0)	R1	LOCK_UNLOCK	Used to set/reset the password or lock/unlock the card. The size of the data block is set by the SET_BLOCK_LEN command. Reserved bits in the argument and in Lock Card Data Structure shall be set to 0.

**Table 20-11. Application-specific commands (class 8)**

Cmd index	type	argument	Response format	Abbreviation	Description
ACMD41	bcr	[31]reserved bit [30]HCS [29:24]reserved bits [23:0]V <sub>DD</sub> Voltage Window(OCR[23:0])	R3	SD_SEND_OP_COND	Sends host capacity support information (HCS) and asks the accessed card to send its operating condition register(OCR) content in the response. HCS is effective when card receives SEND_IF_COND command. CCS bit is assigned to OCR[30].
ACMD42	ac	[31:1] stuff bits [0]set_cd	R1	SET_CLR_CD_DETECT	Connect[1]/Disconnect[0] the 50K pull-up resistor on CD/DAT3 (pin 1) of the card.
ACMD51	adtc	[31:0] stuff bits	R1	SEND_SCR	Reads the SD Configuration Register (SCR).
CMD55	ac	[31:16] RCA [15:0] stuff bits	R1	APP_CMD	Indicates to the card that the next command is an application specific command rather than a standard command.
CMD56	adtc	[31:1] stuff bits. [0] RD/WR	R1	GEN_CMD	Used either to transfer a data block to the card or to get a

<b>Cmd index</b>	<b>type</b>	<b>argument</b>	<b>Response format</b>	<b>Abbreviation</b>	<b>Description</b>
					data block from the card for general purpose/application specific command. The host sets RD/WR=1 for reading data from the card and sets to 0 for writing data to the card.
CMD60	adtc	[31] WR [23:18] Address [7:2] Byte Count Other bits are reserved bits.	R1(read)/ R1b(write)	RW_MULTIPLE_REGISTER	Read or write register in address range.
CMD61	adtc	[31] WR [15:0] Data Unit Count Other bits are reserved bits	R1(read)/ R1b(write)	RW_MULTIPLE_BLOCK	Read or write data block in address range.
<b>Note:</b> 1.ACMDx is Application-specific Commands for SD memory. 2. CMD60, CMD61 are Application-specific Commands for CE-ATA device.					

**Table 20-12. I/O mode commands (class 9)**

<b>Cmd index</b>	<b>type</b>	<b>argument</b>	<b>Response format</b>	<b>Abbreviation</b>	<b>Description</b>
CMD39	ac	[31:16] RCA [15] register write flag [14:8] register address [7:0] register data	R4	FAST_IO	Used to write and read 8 bit (register) data fields. The command addresses a card and a register and provides the data for writing if the write flag is set. The R4 response contains data read from the addressed register if the write flag is cleared to 0. This command accesses application dependent registers which are not defined in the MMC standard.
CMD40	bcr	[31:0] stuff bits	R5	GO_IRQ_STAT_E	Sets the system into interrupt mode
CMD52	adtc	[31] R/W Flag [30:28] Function Number	R5	IO_RW_DIRECT	The IO_RW_DIRECT is the simplest means to access a single register within the total

<b>Cmd index</b>	<b>type</b>	<b>argument</b>	<b>Response format</b>	<b>Abbreviation</b>	<b>Description</b>
		[27] RAW Flag [26] Stuff Bits [25:9] Register Address [8] Stuff Bits [7:0] Write Data/Stuff Bits			128K of register space in any I/O function, including the common I/O area (CIA). This command reads or writes 1 byte using only 1 command/response pair. A common use is to initialize registers or monitor status values for I/O functions. This command is the fastest means to read or write single I/O registers, as it requires only a single command/response pair.
CMD53	adtc	[31] R/W Flag [30:28] Function Number [27] Block Mode [26] OP code [25:9] Register Address [8:0] Byte/Block Count		IO_RW_EXTENDED	This command allows the reading or writing of a large number of I/O registers with a single command.
<b>Note:</b> 1.CMD39, CMD40 are only for MMC. 2. CMD52, CMD53 are only for SD I/O card.					

**Table 20-13. Switch function commands (class 10)**

<b>Cmd index</b>	<b>type</b>	<b>argument</b>	<b>Response format</b>	<b>Abbreviation</b>	<b>Description</b>
CMD6	adtc	[31] Mode 0:Check function 1:Switch function [30:24] reserved [23:20] reserved for function group 6 (0h or Fh) [19:16] reserved for function group 5 (0h or Fh) [15:12] reserved for function group 4 (0h or Fh) [11:8] reserved for function group 3 (0h or Fh)	R1	SWITCH_FUNC	Only for SD memory and SD I/O. Checks switchable function (mode 0) and switch card function (mode 1).

		[7:4] function group 2 for command system [3:0] function group 1 for access mode		
--	--	---	--	--

### 20.5.3. Responses

All responses are sent on the CMD line. The response transmission always starts with the left bit of the bit string corresponding to the response code word. The code length depends on the response type.

#### Responses types

There are 7 types of responses show as follows.

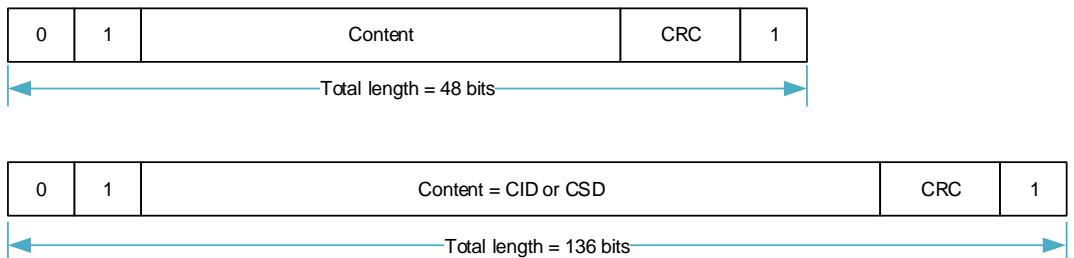
- **R1 / R1b** : normal response command.
- **R2** : CID, CSD register.
- **R3** : OCR register.
- **R4** : Fast I/O.
- **R5** : Interrupt request.
- **R6** : Published RCA response.
- **R7** : Card interface condition.

The SD Memory Card support five types of them, R1 / R1b, R2, R3, R6, R7. And the SD I/O Card and MMC supports additional response types named R4 and R5, but they are not exactly the same for SD I/O Card and MMC.

#### Responses format

Responses have two formats, as show in [Figure 20-8. Response Token Format](#), all responses are sent on the CMD line. The code length depends on the response type. Except R2 is 136 bits length, others are all 48 bits length.

**Figure 20-8. Response Token Format**



A response always starts with a start bit (always 0), followed by the bit indicating the direction of transmission (card = 0). A value 'x' in the tables below indicates a variable entry. All responses except for the type R3 are protected by a CRC. Every command code word is

terminated by the end bit (always 1).

### R1 (normal response command)

Code length is 48 bits. The bits 45:40 indicate the index of the command to be responded to, this value being interpreted as a binary coded number (between 0 and 63). The status of the card is coded in 32 bits. Note that if a data transfer to the card is involved, then a busy signal may appear on the data line after the transmission of each block of data. The host shall check for busy after data block transmission. The card status is described in [Two status fields of the card.](#)

**Table 20-14. Response R1**

<b>Bit position</b>	47	46	[45:40]	[39:8]	[7:1]	0
<b>Width</b>	1	1	6	32	7	1
<b>Value</b>	'0'	'0'	x	x	x	'1'
<b>description</b>	start bit	transmission bit	command index	card status	CRC7	end bit

### R1b

R1b is identical to R1 with an optional busy signal transmitted on the data line DAT0. The card may become busy after receiving these commands based on its state prior to the command reception. The Host shall check for busy at the response.

### R2 (CID, CSD register)

Code length is 136 bits. The contents of the CID register are sent as a response to the commands CMD2 and CMD10. The contents of the CSD register are sent as a response to CMD9. Only the bits [127..1] of the CID and CSD are transferred, the reserved bit [0] of these registers is replaced by the end bit of the response.

**Table 20-15. Response R2**

<b>Bit position</b>	135	134	[133:128]	[127:1]	0
<b>Width</b>	1	1	6	127	1
<b>Value</b>	'0'	'0'	'111111'	x	'1'
<b>description</b>	start bit	transmission bit	reserved	CID or CSD register and internal CRC7	end bit

### R3 (OCR register)

Code length is 48 bits. The contents of the OCR register are sent as a response to ACMD41 (SD memory), CMD1 (MMC). The response of different cards may have a little different.

**Table 20-16. Response R3**

<b>Bit position</b>	47	46	[45:40]	[39:8]	[7:1]	0
<b>Width</b>	1	1	6	32	7	1
<b>Value</b>	'0'	'0'	'111111'	x	'1111111'	'1'
<b>description</b>	start bit	transmission bit	reserved	OCR register	reserved	end bit

### R4 (Fast I/O)

For MMC only. Code length 48 is bits. The argument field contains the RCA of the addressed card, the register address to be read out or written to, and its contents. The status bit in the argument is set if the operation was successful.

**Table 20-17. Response R4 for MMC**

<b>Bit position</b>	47	46	[45:40]	[39:8] Argument field					[7:1]	0
<b>Width</b>	1	1	6	16 1 7 8					7	1
<b>Value</b>	'0'	'0'	'100111'	x x x x					x	'1'
<b>description</b>	start bit	transmission bit	CMD39	RCA [31:16]	status [15]	register address [14:8]	read register contents [7:0]	CRC7	end bit	

### R4b

For SD I/O only. Code length is 48 bits. The SDIO card receive the CMD5 will respond with a unique SD I/O response R4.

**Table 20-18. Response R4 for SD I/O**

<b>Bit position</b>	47	46	[45:40]	39	[38:36]	35	[34:32]	31	[30:8]	[7:1]	0
<b>Width</b>	1	1	6	1	3	1	3	1	23	7	1
<b>Value</b>	'0'	'0'	'1111 11'	x	x	x	'000'	x	x	'1111 111'	1
<b>description</b>	start bit	transmission bit	Reserved	C	Number of I/O functions	Memory Present	Stuff Bits	S18 A	I/O OCR	Reserved	end bit

### R5 (Interrupt request)

For MMC only. Code length is 48 bits. If the response is generated by the host, the RCA field in the argument will be 0x0.

**Table 20-19. Response R5 for MMC**

<b>Bit position</b>	47	46	[45:40]	[39:8] Argument field					[7:1]	0
<b>Width</b>	1	1	6	16 16					7	1
<b>Value</b>	'0'	'0'	'101000'	x x					x	'1'

<b>description</b>	start bit	transmission bit	CMD40	RCA [31:16] of winning card or of the host	[15:0]	Not defined. May be used for IRQ data	CRC7	end bit
--------------------	-----------	------------------	-------	--	--------	---------------------------------------	------	---------

### R5b

For SD I/O only. The SDIO card's response to CMD52 and CMD53 is R5. If the communication between the card and host is in the 1-bit or 4-bit SD mode, the response shall be in a 48-bit response (R5).

**Table 20-20. Response R5 for SD I/O**

<b>Bit position</b>	47	46	[45:40]	[39:24]	[23:16]	[15:8]	[7:1]	0
<b>Width</b>	1	1	6	16	8	8	7	1
<b>Value</b>	'0'	'0'	'11020X'	'0'	x	x	x	'1'
<b>description</b>	start bit	transmission bit	CMD52/53	Stuff Bits	Response Flags	Read or Write Data	CRC7	end bit

### R6 (Published RCA response)

Code length is 48 bit. The bits [45:40] indicate the index of the command to be responded to (CMD3). The 16 MSB bits of the argument field are used for the Published RCA number.

**Table 20-21. Response R6**

<b>Bit position</b>	47	46	[45:40]	[39:8] Argument field	[7:1]	0	
<b>Width</b>	1	1	6	16	16	7	
<b>Value</b>	'0'	'0'	'000011'	x	x	x	
<b>description</b>	start bit	transmission bit	CMD3	New published RCA of the card	card status bits:23,22,19,12:0	CRC7	end bit

### R7 (Card interface condition)

For SD memory only. Code length is 48 bits. The card support voltage information is sent by the response of CMD8. Bits 19-16 indicate the voltage range that the card supports. The card that accepted the supplied voltage returns R7 response. In the response, the card echoes back both the voltage range and check pattern set in the argument.

**Table 20-22. Response R7**

<b>Bit position</b>	47	46	[45:40]	[39:20]	[19:16]	[15:8]	[7:1]	0
<b>Width</b>	1	1	6	20	4	8	7	1
<b>Value</b>	'0'	'0'	'001000'	'00000h'	x	x	x	'1'
<b>description</b>	start bit	transmission bit	CMD8	Reserved bits	Voltage accepted	echo-back of check pattern	CRC7	end bit

## 20.5.4. Data packets format

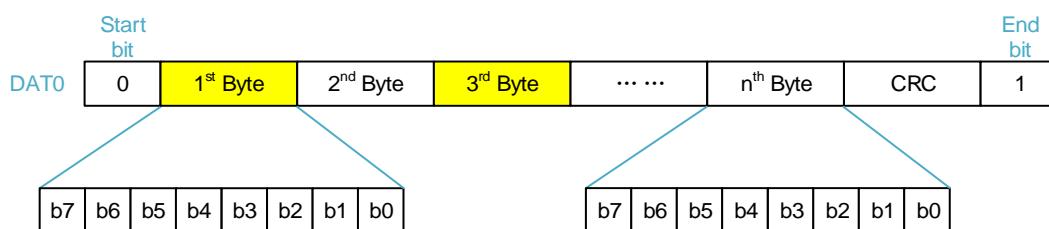
There are 3 data bus mode, 1-bit, 4-bit and 8-bit width. 1-bit mode is mandatory, 4-bit and 8-

bit mode is optional. Although using 1-bit mode, DAT3 also need to notify card current working mode is SDIO or SPI, when card reset and initialize.

### 1-bit data packet format

After card reset and initialize, only DAT0 pin is used to transfer data. And other pin can be used freely. [Figure 20-9. 1-bit data bus width](#), [Figure 20-10. 4-bit data bus width](#) and [Figure 20-11. 8-bit data bus width](#) show the data packet format when data bus wide is 1-bit, 4-bit and 8-bit.

**Figure 20-9. 1-bit data bus width**



### 4-bit data packet format

**Figure 20-10. 4-bit data bus width**

	Start bit	1 <sup>st</sup> Byte	2 <sup>nd</sup> Byte	3 <sup>rd</sup> Byte			n <sup>th</sup> Byte	CRC	End bit
DAT3	0	b7 b3	b7 b3	b7 b3	....		b7 b3	CRC	1
DAT2	0	b6 b2	b6 b2	b6 b2	....		b6 b2	CRC	1
DAT1	0	b5 b1	b5 b1	b5 b1	....		b5 b1	CRC	1
DAT0	0	b4 b0	b4 b0	b4 b0	....		b4 b0	CRC	1

### 8-bit data packet format

**Figure 20-11. 8-bit data bus width**

	Start bit	1 <sup>st</sup> Byte	2 <sup>nd</sup> Byte	3 <sup>rd</sup> Byte				n <sup>th</sup> Byte	CRC	End bit
DAT7	0	b7	b7	b7	....			b7	CRC	1
DAT6	0	b6	b6	b6	....			b6	CRC	1
DAT5	0	b5	b5	b5	....			b5	CRC	1
DAT4	0	b4	b4	b4	....			b4	CRC	1
DAT3	0	b7	b3	b7	....			b3	CRC	1
DAT2	0	b6	b2	b6	....			b2	CRC	1
DAT1	0	b5	b1	b5	....			b1	CRC	1
DAT0	0	b4	b0	b4	....			b0	CRC	1

### **20.5.5. Two status fields of the card**

The SD Memory supports two status fields and others just support the first one:

Card Status: Error and state information of a executed command, indicated in the response

SD Status: Extended status field of 512 bits that supports special features of the SD Memory Card and future Application-Specific features.

#### **Card status**

The response format R1 contains a 32-bit field named card status. This field is intended to transmit the card's status information (which may be stored in a local status register) to the host. If not specified otherwise, the status entries are always related to the previous issued command.

The type and clear condition fields in the table are abbreviated as follows:

##### **Type**

- E: Error bit. Send an error condition to the host. These bits are cleared as soon as the response (reporting the error) is sent out.
- S: Status bit. These bits serve as information fields only, and do not alter the execution of the command being responded to. These bits are persistent, they are set and cleared in accordance with the card status.
- R: Exceptions are detected by the card during the command interpretation and validation phase (Response Mode).
- X: Exceptions are detected by the card during command execution phase (Execution Mode).

##### **Clear condition**

- A: According to current state of the card.
- B: Always related to the previous command. Reception of a valid command will clear it (with a delay of one command).
- C: Cleared by read

**Table 20-23. Card status**

<b>Bits</b>	<b>Identifier</b>	<b>Type</b>	<b>Value</b>	<b>Description</b>	<b>Clear Condition</b>
31	OUT_OF_RANGE	ERX	'0'= no error '1'= error	The command's argument was out of the allowed range for this card.	C
30	ADDRESS_ERROR	ERX	'0'= no error '1'= error	A misaligned address which did not match the block length was used in the command.	C

29	BLOCK_LEN_ERROR	ERX	'0'= no error '1'= error	The transferred block length is not allowed for this card, or the number of transferred bytes does not match the block length.	C
28	ERASE_SEQ_ERROR	ER	'0'= no error '1'= error	An error in the sequence of erase commands occurred.	C
27	ERASE_PARAM	ERX	'0'= no error '1'= error	An invalid selection of write-blocks for erase occurred.	C
26	WP_VIOLATION	ERX	'0'= not protected '1'= protected	Set when the host attempts to write to a protected block or to the temporary or permanent write protected card.	C
25	CARD_IS_LOCKED	SX	'0' = card unlocked '1' = card locked	When set, signals that the card is locked by the host	A
24	LOCK_UNLOCK_FAIL ED	ERX	'0'= no error '1'= error	Set when a sequence or password error has been detected in lock/unlock card command.	C
23	COM_CRC_ERROR	ER	'0'= no error '1'= error	The CRC check of the previous command failed.	B
22	ILLEGAL_COMMAND	ER	'0'= no error '1'= error	Command not legal for the card state.	B
21	CARD_ECC_FAILED	ERX	'0'= success '1'= failure	Card internal ECC was applied but failed to correct the data.	C
20	CC_ERROR	ERX	'0'= no error '1'= error	Internal card controller error.	C
19	ERROR	ERX	'0'= no error '1'= error	A general or an unknown error occurred during the operation.	C
18	UNDERRUN	ERX	'0'= no error '1'= error	Only for MMC. The card could not sustain data transfer in stream read mode.	C
17	OVERRUN	ERX	'0'= no error '1'= error	Only for MMC. The card could not sustain data programming in stream write mode.	C
16	CID/ CSD_OVERWRITE	ERX	'0'= no error '1'= error	Can be either one of the following errors: <ul style="list-style-type: none"><li>- The read only section of the CSD does not match the card content.</li><li>- An attempt to reverse the copy (set as original) or permanent WP(unprotected)</li></ul>	C

				bits was made.	
15	WP_ERASE_SKIP	ERX	'0'= not protected '1'= protected	Set when only partial address space was erased due to existing write protected blocks or the temporary or permanent write protected card was erased.	C
14	CARD_ECC_DISABLE D	SX	'0'= enabled '1'= disabled	The command has been executed without using the internal ECC.	A
13	ERASE_RESET	SR	'0'= cleared '1'= set	An erase sequence was cleared before executing because an out of erase sequence command was received.	C
[12: 9]	CURRENT_STATE	SX	0 = idle 1 = ready 2 = identification 3 = stand by 4 = transfer 5 = send data 6 = receive data 7 = programming 8 = disconnect 9-14 = reserved 15 = reserved for I/O mode	The state of the card when receiving the command. If the command execution causes a state change, it will be visible to the host in the response to the next command. The four bits are interpreted as a binary coded number between 0 and 15.	B
8	READY_FOR_DATA	SX	'0'= not ready '1'= ready	Corresponds to buffer empty signaling on the bus.	A
7	SWITCH_ERROR	EX	'0'= no error '1'= switch error	If set, the card don't switch to the expected mode as requested by the SWITCH command.	B
6	Reserved				
5	APP_CMD	SR	'0'= enabled '1'= disabled	The card will expect ACMD, or an indication that the command has been interpreted as ACMD.	C
4	Reserved				
3	AKE_SEQ_ERROR	ER	'0'= no error '1'= error	Only for SD memory. Error in the sequence of the authentication process.	C
2	Reserved for application specific commands.				

[1:0]	Reserved for manufacturer test mode.
-------	--------------------------------------

**Note:** 18, 17, 7 bits are only for MMC. 14, 3 bits are only for SD memory.

### SD status register

The SD Status contains status bits that are related to the SD Memory Card proprietary features and may be used for future application-specific usage. The size of the SD Status is one data block of 512 bits. The content of this register is transmitted to the Host over the DAT bus along with a 16-bit CRC. The SD Status is sent to the host over the DAT bus as a response to ACMD13 (CMD55 followed with CMD13). ACMD13 can be sent to a card only in ‘transfer state’ (card is selected). The SD Status structure is described below.

The same abbreviation for ‘type’ and ‘clear condition’ were used as for the Card Status above.

**Table 20-24. SD status**

Bits	Identifier	Type	Value	Description	Clear Condition
[511: 510]	DAT_BUS_WIDTH	SR	'00'= 1 (default) '01'= reserved '10'= 4 bit width '11'= reserved	Shows the currently defined data bus width that was defined by SET_BUS_WIDTH command	A
509	SECURED_MODE	SR	'0'= Not in the mode '1'= In Secured Mode	Card is in Secured Mode of operation (refer to the “SD Security Specification”).	A
[508: 496]	reserved				
[495: 480]	SD_CARD_TYPE	SR	The following cards are currently defined: '0000'= Regular SD RD/WR Card. '0001'= SD ROM Card '0002'= OTP	In the future, the 8 LSBs will be used to define different variations of an SD Memory Card (Each bit will define different SD Types). The 8 MSBs will be used to define SD Cards that do not comply with current SD Physical Layer Specification.	A
[479: 448]	SIZE_OF_PROTECT ED_AREA	SR	Size of protected area	(See below)	A
[447: 440]	SPEED_CLASS	SR	Speed class of the card	(See below)	A
[439: 432]	PERFORMANCE_M OVE	SR	Performance of move indicated by 1 [MB/s] step.	(See below)	A

<b>Bits</b>	<b>Identifier</b>	<b>Type</b>	<b>Value</b>	<b>Description</b>	<b>Clear Condition</b>
[431: 428]	AU_SIZE	SR	Size of AU	(See below)	A
[427: 424]	reserved				
[423: 408]	ERASE_SIZE	SR	Number of AUs to be erased at a time	(See below)	A
[407: 402]	ERASE_TIMEOUT	SR	Timeout value for erasing areas specified by UNIT_OF_ERASE_AU	(See below)	A
[401: 400]	ERASE_OFFSET	SR	Fixed offset value added to erase time.	(See below)	A
[399: 312]	reserved				
[311: 0]	reserved for manufacturer				

### **SIZE\_OF\_PROTECTED\_AREA**

Setting this field differs between SDSC and SDHC/SDXC.

In case of SDSC Card, the capacity of protected area is calculated as follows:

Protected Area = SIZE\_OF\_PROTECTED\_AREA\_\* MULT \* BLOCK\_LEN.

SIZE\_OF\_PROTECTED\_AREA is specified by the unit in MULT\*BLOCK\_LEN.

In case of SDHC and SDXC Cards, the capacity of protected area is calculated as follows:

Protected Area = SIZE\_OF\_PROTECTED\_AREA

SIZE\_OF\_PROTECTED\_AREA is specified by the unit in byte.

### **SPEED\_CLASS**

This 8-bit field indicates the Speed Class.

00h: Class 0

01h: Class 2

02h: Class 4

03h: Class 6

04h: Class 10

05h–FFh: Reserved

### **PERFORMANCE\_MOVE**

This 8-bit field indicates Pm and the value can be set by 1 [MB/sec] step. If the card does not move using RUs, Pm should be considered as infinity. Setting to FFh means infinity. The minimum value of Pm is defined in [Table 20-25. Performance move field](#).

**Table 20-25. Performance move field**

PERFORMANCE_MOVE	Value Definition
00h	Sequential Write
01h	1 [MB/sec]
02h	2 [MB/sec]
.....	.....
FEh	254 [MB/sec]
FFh	Infinity

### **AU\_SIZE**

This 4-bit field indicates AU Size and the value can be selected from 16 KB.

**Table 20-26. AU\_SIZE field**

AU_SIZE	Value Definition
0h	Not Defined
1h	16 KB
2h	32 KB
3h	64 KB
4h	128 KB
5h	256 KB
6h	512 KB
7h	1 MB
8h	2 MB
9h	4 MB
Ah	8 MB
Bh	12 MB
Ch	16 MB
Dh	24 MB
Eh	32 MB
Fh	64 MB

The maximum AU size, depends on the card capacity, is defined in [Table 20-26. AU\\_SIZE field](#). The card can set any AU size specified in [Table 20-27. Maximum AU size](#) that is less than or equal to the maximum AU size. The card should set smaller AU size as possible.

**Table 20-27. Maximum AU size**

Card Capacity	up to 64MB	up to 256MB	up to 512MB	up to 32GB	up to 2TB
Maximum AU	512 KB	1 MB	2 MB	4 MB	64MB

<b>Size</b>					
-------------	--	--	--	--	--

### **ERASE\_SIZE**

This 16-bit field indicates  $N_{ERASE}$ . When  $N_{ERASE}$  of AUs are erased, the timeout value is specified by ERASE\_TIMEOUT (Refer to ERASE\_TIMEOUT). The host should determine proper number of AUs to be erased in one operation so that the host can indicate progress of erase operation. If this field is set to 0, the erase timeout calculation is not supported.

**Table 20-28. Erase size field**

<b>ERASE_SIZE</b>	<b>Value Definition</b>
0000h	Erase Time-out Calculation is not supported.
0001h	1 AU
0002h	2 AU
0003h	3 AU
.....	.....
FFFFh	65535 AU

### **ERASE\_TIMEOUT**

This 6-bit field indicates the  $T_{ERASE}$  and the value indicates erase timeout from offset when multiple AUs are erased as specified by ERASE\_SIZE. The range of ERASE\_TIMEOUT can be defined as up to 63 seconds and the card manufacturer can choose any combination of ERASE\_SIZE and ERASE\_TIMEOUT depending on the implementation. Once ERASE\_TIMEOUT is determined, it determines the ERASE\_SIZE. The host can determine timeout for any number of AU erase by the equation below.

$$\text{Erase timeout of } X \text{ AU} = \frac{T_{ERASE}}{N_{ERASE}} * X + T_{OFFSET} \quad (20-1)$$

**Table 20-29. Erase timeout field**

<b>ERASE_TIMEOUT</b>	<b>Value Definition</b>
00	Erase Time-out Calculation is not supported.
01	1 [sec]
02	2 [sec]
03	3 [sec]
.....	.....
63	63 [sec]

If ERASE\_SIZE field is set to 0, this field shall be set to 0.

### **ERASE\_OFFSET**

This 2-bit field indicates the  $T_{OFFSET}$  and one of four values can be selected. This field is meaningless if ERASE\_SIZE and ERASE\_TIMEOUT fields are set to 0.

**Table 20-30. Erase offset field**

<b>ERASE_OFFSET</b>	<b>Value Definition</b>
0h	0 [sec]

1h	1 [sec]
2h	2 [sec]
3h	3 [sec]

## 20.6. Programming sequence

### 20.6.1. Card identification

The host will be in card identification mode after reset and while it is looking for new cards on the bus. While in card identification mode the host resets all the cards, validates operation voltage range, identifies cards and asks them to publish Relative Card Address (RCA). This operation is done to each card separately on its own CMD line. All data communication in the Card Identification Mode uses the command line (CMD) only.

During the card identification process, the card shall operate in the clock frequency of the identification clock rate  $F_{OD}$  (400 kHz).

#### Card reset

The command GO\_IDLE\_STATE (CMD0) is the software reset command and sets MMC and SD memory card into Idle State regardless of the current card state. The reset command (CMD0) is only used for memory or the memory portion of Combo cards. In order to reset an I/O only card or the I/O portion of a combo card, use CMD52 to write 1 to the RES bit in the CCCR. Cards in Inactive State are not affected by this command.

After power-on by the host, all cards are in Idle State, including the cards that have been in Inactive State before. After power-on or CMD0, all cards' CMD lines are in input mode, waiting for start bit of the next command. The cards are initialized with a default relative card address (RCA) and with a default driver strength with 400 KHz clock frequency.

#### Operating voltage range validation

At the start of communication between the host and the card, the host may not know the card supported voltage and the card may not know whether it supports the current supplied voltage. To verify the voltage, the following commands are defined in the related specification.

The SEND\_OP\_COND (CMD1 for MMC), SD\_SEND\_OP\_COND (ACMD41 for SD memory), IO\_SEND\_OP\_COND (CMD5 for SD I/O) command is designed to provide hosts with a mechanism to identify and reject cards which do not match the  $V_{DD}$  range desired by the host. This is accomplished by the host sending the required  $V_{DD}$  voltage window as the operand of this command. If the card cannot perform data transfer in the specified range it must discard itself from further bus operations and go into Inactive State. Otherwise, the card shall respond sending back its  $V_{DD}$  range.

If the card can operate on the supplied voltage, the response echoes back the supply voltage and the check pattern that were set in the command argument.

If the card cannot operate on the supplied voltage, it returns no response and stays in idle state. It is mandatory to issue CMD8 prior to ACMD41 to initialize SDHC Card. Receipt of CMD8 makes the cards realize that the host supports the Physical Layer Version 2.00 and the card can enable new functions.

### Card identification process

The card identification process differs in different cards. The card can be of the type MMC, CE-ATA, SD, or SD I/O. All types of SD I/O cards are supported, that is, SDIO\_IO\_ONLY, SDIO\_MEM\_ONLY, and SDIO COMBO cards. The identification process sequence includes the following steps:

1. Check if the card is connected.
2. Identify the card type; SD, MMC(CE-ATA), or SD I/O.
  - Send CMD5 first. If a response is received, then the card is SD I/O
  - If not, send ACMD41; if a response is received, then the card is SD.
  - Otherwise, the card is an MMC or CE-ATA.
3. Initialization the card according to the card type.

Use a clock source with a frequency =  $F_{OD}$  (that is, 400 KHz) and use the following command sequence:

- SD card - Send CMD0, ACMD41, CMD2, CMD3.
  - SDHC card - send CMD0, CMD8, ACMD41, CMD2, CMD3.
  - SD I/O - Send CMD52, CMD0, CMD5, if the card doesn't have memory port, send CMD3; otherwise send ACMD41, CMD11 (optional), CMD2, and CMD3.
  - MMC/CE-ATA - Send CMD0, CMD1, CMD2, CMD3.
4. Identify the MMC/CE-ATA device.
    - CPU should query the byte 504 (S\_CMD\_SET) of EXT\_CSD register by sending CMD8. If bit 4 is set to 1, then the device supports ATA mode.
    - If ATA mode is supported, the CPU should select the ATA mode by setting the ATA bit (bit 4) in the EXT\_CSD register slice 191(CMD\_SET) to activate the ATA command set. The CPU selects the command set using the SWITCH (CMD6) command.
    - In the presence of a CE-ATA device, the FAST\_IO (CMD39) and RW\_MULTIPLE\_REGISTER (CMD60) commands will succeed and the returned data will be the CE-ATA reset signature.

### 20.6.2. No data commands

To send any non-data command, the software needs to program the SDIO\_CMDCTL register

and the SDIO\_CMDAGMT register with appropriate parameters. Using these two registers, the host forms the command and sends it to the command bus. The host reflects the errors in the command response through the error bits of the SDIO\_STAT register.

When a response is received the host sets the CMDRECV (CRC check passed) or CCRCERR (CRC check error) bit in the SDIO\_STAT register. A short response is copied in SDIO\_RESP0, while a long response is copied to all four response registers. The SDIO\_RESP3 bit 31 represents the MSB, and the SDIO\_RESP0 bit 0 represents the LSB of a long response.

### 20.6.3. Single block or multiple block write

During block write (CMD24 - 27) one or more blocks of data are transferred from the host to the card. The block consists of start bits (1 or 4 bits LOW), data block, CRC and end bits(1 or 4 bits HIGH). If the CRC fails, the card indicates the failure on the SDIO\_DAT line and the transferred data are discarded and not written, and all further transmitted blocks are ignored.

If the host uses partial blocks whose accumulated length is not block aligned, block misalignment is not allowed (CSD parameter WRITE\_BLK\_MISALIGN is not set), the card will detect the block misalignment error before the beginning of the first misaligned block. The card shall set the ADDRESS\_ERROR error bit in the status register, and while ignoring all further data transfer. The write operation will also be aborted if the host tries to write data on a write protected area. In this case, however, the card will set the WP\_VIOLATION bit (in the status register).

Programming of the CID and CSD registers does not require a previous block length setting. The transferred data is also CRC protected. If a part of the CSD or CID register is stored in ROM, then this unchangeable part must match the corresponding part of the receive buffer. If this match fails, then the card reports an error and does not change any register contents.

Some cards may require long and unpredictable time to write a block of data. After receiving a block of data and completing the CRC check, the card will begin writing and hold the DAT0 line low if its write buffer is full and unable to accept new data from a new WRITE\_BLOCK command. The host may poll the status of the card with a SEND\_STATUS command (CMD13) at any time, and the card will respond with its status. The status bit READY\_FOR\_DATA indicates whether the card can accept new data or whether the write process is still in progress). The host may deselect the card by issuing CMD7 (to select a different card) which will displace the card into the Disconnect State and release the DAT line without interrupting the write operation. When reselecting the card, it will reactivate busy indication by pulling DAT to low if programming is still in progress and the write buffer is unavailable.

For SD card. Setting a number of write blocks to be pre-erased (ACMD23) will make a following Multiple Block Write operation faster compared to the same operation without preceding ACMD23. The host will use this command to define how many number of write blocks are going to be send in the next write operation.

Steps involved in a single-block or multiple-block write are:

1. Write the data size in bytes in the SDIO\_DATALEN register.
2. Write the block size in bytes (BLKSZ) in the SDIO\_DATACTL register; the host sends data in blocks of size BLKSZ.
3. Program SDIO\_CMDAGMT register with the data address to which data should be written.
4. Program the SDIO\_CMDCTL register. For SD memory and MMC cards, use CMD24 for a single-block write and CMD25 for a multiple-block write. For SD I/O cards, use CMD53 for both single-block and multiple-block transfers. For CE-ATA, first use CMD60 to write the ATA task file, then use CMD61 to write the data. After writing to the CMD register, host starts executing a command, when the command is sent to the bus, the CMDRECV flag is set.
5. Write data to SDIO\_FIFO.
6. Software should look for data error interrupts. If required, software can terminate the data transfer by sending the STOP command (CMD12).
7. When a DTEND interrupt is received, the data transfer is over. For an open-ended block transfer, if the byte count is 0, the software must send the STOP command. If the byte count is not 0, then upon completion of a transfer of a given number of bytes, the host should send the STOP command.

#### 20.6.4. Single block or multiple block read

Block read is block oriented data transfer. The basic unit of data transfer is a block whose maximum size is defined in the CSD (READ\_BL\_LEN), it is always 512 bytes. If READ\_BL\_PARTIAL(in the CSD) is set, smaller blocks whose starting and ending address are entirely contained within 512 bytes boundary may be transmitted.

CMD17 (READ\_SINGLE\_BLOCK) initiates a block read and after completing the transfer, the card returns to the Transfer state. CMD18 (READ\_MULTIPLE\_BLOCK) starts a transfer of several consecutive blocks. CRC is appended to the end of each block, ensuring data transfer integrity.

Block Length set by CMD16 can be set up to 512 bytes regardless of READ\_BL\_LEN.

Blocks will be continuously transferred until a STOP\_TRANSMISSION command (CMD12) is issued. The stop command has an execution delay due to the serial command transmission. The data transfer stops after the end bit of the stop command.

When the last block of user area is read using CMD18, the host should ignore OUT\_OF\_RANGE error that may occur even the sequence is correct.

If the host uses partial blocks whose accumulated length is not block aligned and block misalignment is not allowed, the card shall detect a block misalignment at the beginning of the first misaligned block, set the ADDRESS\_ERROR error bit in the status register, abort transmission and wait in the Data State for a stop command.

Steps involved in a single block or multiple block read are:

1. Write the data size in bytes in the SDIO\_DATALEN register.
2. Write the block size in bytes (BLKSZ) in the SDIO\_DATACTL register. The host expects data from the card in blocks of size BLKSZ each.
3. Program the SDIO\_CMDAGMT register with the data address of the beginning of a data read.
4. Program the SDIO\_CMDCTL. For SD and MMC cards, using CMD17 for a single-block read and CMD18 for a multiple-block read. For SD I/O cards, using CMD53 for both single-block and multiple-block transfers. For CE-ATA, first using CMD60 to write the ATA task file, then using CMD 61 to read the data. After writing to the CMD register, the host starts executing the command, when the command is sent to the bus, the CMDRECV flag is set.
5. Software should look for data error interrupts. If required, software can terminate the data transfer by sending a STOP command.
6. The software should read data from the FIFO and make space in the FIFO for receiving more data.
7. When a DTEND interrupt is received, the software should read the remaining data in the FIFO.

### **20.6.5. Stream write and stream read (MMC only)**

#### **Stream write**

Stream write (CMD20) starts the data transfer from the host to the card beginning from the starting address until the host issues a stop command. If partial blocks are allowed (if CSD parameter WRITE\_BL\_PARTIAL is set) the data stream can start and stop at any address within the card address space, otherwise it shall start and stop only at block boundaries. Since the amount of data to be transferred is not determined in advance, CRC cannot be used.

If the host provides an out of range address as an argument to CMD20, the card will reject the command, remain in Tran state and respond with the ADDRESS\_OUT\_OF\_RANGE bit set.

Note that the stream write command works only on a 1 bit bus configuration (on DAT0). If CMD20 is issued in other bus configurations, it is regarded as an illegal command.

In order to sustain data transfer in stream mode of the card, the time it takes to receive the data (defined by the bus clock rate) must be less than the time it takes to program it into the main memory field (defined by the card in the CSD register). Therefore, the maximum clock frequency for the stream write operation is given by the following formula:

$$\text{max write frequency} = \min \left( \text{TRAN\_SPEED}, \frac{8 * 2^{\text{WRITE\_BL\_LEN}} - 100 * \text{NSAC}}{\text{TAAC} * \text{R2W\_FACTOR}} \right) \quad (20-2)$$

**TRAN\_SPEED:** Max bus clock frequency.

**WRITE\_BL\_LEN:** Max write data block length.

**NSAC:** Data read access-time 2 in CLK cycles.

**TAAC:** Data read access-time 1.

**R2W\_FACTOR:** Write speed factor.

All the parameters are defined in CSD register. If the host attempts to use a higher frequency, the card may not be able to process the data and will stop programming, and while ignoring all further data transfer, wait (in the Receive-data-State) for a stop command. As the host sends CMD12, the card will respond with the TXURE bit set and return to Transfer state

### Stream read

There is a stream oriented data transfer controlled by READ\_DAT\_UNTIL\_STOP (CMD11). This command instructs the card to send its data, starting at a specified address, until the host sends a STOP\_TRANSMISSION command (CMD12). The stop command has an execution delay due to the serial command transmission. The data transfer stops after the end bit of the stop command.

If the host provides an out of range address as an argument to CMD11, the card will reject the command, remain in Transfer state and respond with the ADDRESS\_OUT\_OF\_RANGE bit set.

Note that the stream read command works only on a 1 bit bus configuration (on DAT0). If CMD11 is issued in other bus configurations, it is regarded as an illegal command.

If the end of the memory range is reached while sending data, and no stop command has been sent yet by the host, the contents of the further transferred payload is undefined. As the host sends CMD12 the card will respond with the ADDRESS\_OUT\_OF\_RANGE bit set and return to Tran state.

In order to sustain data transfer in stream mode of the card, the time it takes to transmit the data (defined by the bus clock rate) must be less than the time it takes to read it out of the main memory field (defined by the card in the CSD register). Therefore, the maximum clock frequency for stream read operation is given by the following formula:

$$\text{max read frequency} = \min \left( \text{TRAN\_SPEED}, \frac{8 * 2^{\text{READ\_BL\_LEN}} - 100 * \text{NSAC}}{\text{TAAC} * \text{R2W\_FACTOR}} \right) \quad (20-3)$$

**TRAN\_SPEED:** Max bus clock frequency.

**READ\_BL\_LEN:** Max read data block length.

**NSAC:** Data read access-time 2 in CLK cycles.

**TAAC:** Data read access-time 1.

**R2W\_FACTOR:** Write speed factor.

All the parameters are defined in CSD register. If the host attempts to use a higher frequency,

the card may not be able to process the data and will stop programming, and while ignoring all further data transfer, wait (in the Receive-data-State) for a stop command. As the host sends CMD12, the card will respond with the RXORE bit set and return to Transfer state

## 20.6.6. Erase

The erasable unit of the MMC/SD memory is the “Erase Group”; Erase group is measured in write blocks which are the basic writable units of the card. The size of the Erase Group is a card specific parameter and defined in the CSD.

The host can erase a contiguous range of Erase Groups. Starting the erase process is a three steps sequence. First the host defines the start address of the range using the ERASE\_GROUP\_START (CMD35)/ERASE\_WR\_BLK\_START(CMD32) command, next it defines the last address of the range using the ERASE\_GROUP\_END (CMD36)/ERASE\_WR\_BLK\_END(CMD33) command and finally it starts the erase process by issuing the ERASE (CMD38) command. The address field in the erase commands is an Erase Group address in byte units. The card will ignore all LSB's below the Erase Group size, effectively rounding the address down to the Erase Group boundary.

If an erase command (CMD35, CMD36, and CMD38) is received out of the defined erase sequence, the card shall set the ERASE\_SEQ\_ERROR bit in the status register and reset the whole sequence.

If the host provides an out of range address as an argument to CMD35 or CMD36, the card will reject the command, respond with the ADDRESS\_OUT\_OF\_RANGE bit set and reset the whole erase sequence.

If an ‘non erase’ command (neither of CMD35, CMD36, CMD38 or CMD13) is received, the card shall respond with the ERASE\_RESET bit set, reset the erase sequence and execute the last command.

If the erase range includes write protected blocks, they shall be left intact and only the non-protected blocks shall be erased. The WP\_ERASE\_SKIP status bit in the status register shall be set.

As described above for block write, the card will indicate that an erase is in progress by holding DAT0 low. The actual erase time may be quite long, and the host may issue CMD7 to deselect the card.

## 20.6.7. Bus width selection

After the host has verified the functional pins on the bus it should change the bus width configuration.

For MMC, using the SWITCH command (CMD6).The bus width configuration is changed by writing to the BUS\_WIDTH byte in the Modes Segment of the EXT\_CSD register (using the SWITCH command to do so). After power-on or software reset, the contents of the

BUS\_WIDTH byte is 0x00. If the host tries to write an invalid value, the BUS\_WIDTH byte is not changed and the SWITCH\_ERROR bit is set. This register is write only.

For SD memory, using SET\_BUS\_WIDTH command (ACMD6) to change the bus width. The default bus width after power up or GO\_IDLE\_STATE command (CMD0) is 1 bit. SET\_BUS\_WIDTH (ACMD6) is only valid in a transfer state, which means that the bus width can be changed only after a card is selected by SELECT/DESELECT\_CARD (CMD7).

## 20.6.8. Protection management

In order to allow the host to protect data against erase or write, three methods for the cards are supported in the card:

### CSD register for card protection (optional)

The entire card may be write protected by setting the permanent or temporary write protect bits in the CSD. Some cards support write protection of groups of sectors by setting the WP\_GRP\_ENABLE bit in the CSD. It is defined in units of WP\_GRP\_SIZE erase groups as specified in the CSD. The SET\_WRITE\_PROT command sets the write protection of the addressed write protected group, and the CLR\_WRITE\_PROT command clears the write protection of the addressed write protected group.

The High Capacity SD Memory Card does not support Write Protection and does not respond to write protection commands (CMD28, CMD29 and CMD30).

### Write protect switch on the card (SD memory and SD I/O card)

A mechanical sliding tablet on the side of the card will be used by the user to indicate that a given card is write protected or not. If the sliding tablet is positioned in such a way that the window is open it means that the card is write protected. If the window is closed the card is not write protected.

### Password card Lock/Unlock Operation

The Password Card Lock/Unlock protection is described in [Card Lock/Unlock operation](#).

## 20.6.9. Card Lock/Unlock operation

The password protection feature enables the host to lock a card while providing a password, which later will be used for unlocking the card. The password and its size are kept in a 128-bit PWD and 8-bit PWD\_LEN registers, respectively. These registers are non-volatile so that a power cycle will not erase them.

Locked cards respond to (and execute) all commands in the basic command class (class 0), ACMD41, CMD16 and lock card command class (class 7). Thus, the host is allowed to reset, initialize, select, query for status, but not to access data on the card. If the password was previously set (the value of PWD\_LEN is not 0), the card will be locked automatically after power on.

Similar to the existing CSD register write commands, the lock/unlock command is available in "transfer state" only. This means that it does not include an address argument and the card shall be selected before using it.

The card lock/unlock command has the structure and bus transaction type of a regular single block write command. The transferred data block includes all the required information of the command (password setting mode, PWD itself, card lock/unlock etc.). [\*\*Table 20-31. Lock card data structure\*\*](#) describes the structure of the command data block.

**Table 20-31. Lock card data structure**

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Reserved(all set to 0)				ERASE	LOCK_UNLOCK	CLR_PWD	SET_PWD
1	PWDS_LEN							
2								
.....	Password data(PWD)							
PWDS_LEN+1								

**ERASE:** 1 Defines Forced Erase Operation. In byte 0, bit 3 will be set to 1 (all other bits shall be 0). All other bytes of this command will be ignored by the card.

**LOCK/UNLOCK:** 1 = Locks the card. 0 = Unlock the card (note that it is valid to set this bit together with SET\_PWD but it is not allowed to set it together with CLR\_PWD).

**CLR\_PWD:** 1 = Clears PWD.

**SET\_PWD:** 1 = Set new password to PWD.

**PWDS\_LEN:** Defines the following password(s) length (in bytes). In case of a password change, this field includes the total password length of old and new passwords. The password length is up to 16 bytes. In case of a password change, the total length of the old password and the new password can be up to 32 bytes.

**Password data:** In case of setting a new password, it contains the new password. In case of a password change, it contains the old password followed by the new password.

## Setting the password

- Select a card (CMD7), if not previously selected.
- Define the block length (CMD16), given by the 8-bit card lock/unlock mode, the 8-bit password size (in bytes), and the number of bytes of the new password. In the case that a password replacement is done, then the block size shall consider that both passwords (the old and the new one) are sent with the command.
- Send the Card Lock/Unlock command with the appropriate data block size on the data line including the 16-bit CRC. The data block shall indicate the mode (SET\_PWD), the length (PWDS\_LEN) and the password itself. In the case that a password replacement is done, then the length value (PWDS\_LEN) shall include both passwords (the old and the new one) and the password data field shall include the old password (currently used) followed by the new password. Note that the card shall handle the calculation of the new

password length internally by subtracting the old password length from PWDS\_LEN field.

- In the case that the sent old password is not correct (not equal in size and content), then the LOCK\_UNLOCK\_FAILED error bit will be set in the status register and the old password does not change. In the case that the sent old password is correct (equal in size and content), then the given new password and its size will be saved in the PWD and PWD\_LEN registers, respectively.

### Reset the password

- Select a card (CMD7), if not previously selected.
- Define the block length (CMD16), given by the 8-bit card lock/unlock mode, the 8-bit password size (in bytes), and the number of bytes of the currently used password.
- Send the card lock/unlock command with the appropriate data block size on the data line including the 16-bit CRC. The data block shall indicate the mode CLR\_PWD, the length (PWDS\_LEN), and the password itself. If the PWD and PWD\_LEN content match the sent password and its size, then the content of the PWD register is cleared and PWD\_LEN is set to 0. If the password is not correct, then the LOCK\_UNLOCK\_FAILED error bit will be set in the status register.

### Locking a card

- Select a card (CMD7), if not previously selected.
- Define the block length (CMD16), given by the 8-bit card lock/unlock mode, the 8-bit password size (in bytes), and the number of bytes of the currently used password.
- Send the card lock/unlock command with the appropriate data block size on the data line including the 16-bit CRC. The data block shall indicate the mode LOCK, the length (PWDS\_LEN) and the password itself.

If the PWD content is equal to the sent password, then the card will be locked and the card-locked status bit will be set in the status register. If the password is not correct, then the LOCK\_UNLOCK\_FAILED error bit will be set in the status register.

### Unlocking the card

- Select a card (CMD7), if not previously selected.
- Define the block length (CMD16), given by the 8-bit card lock/unlock mode, the 8-bit password size (in bytes), and the number of bytes of the currently used password.
- Send the card lock/unlock command with the appropriate data block size on the data line including the 16-bit CRC. The data block shall indicate the mode UNLOCK, the length (PWDS\_LEN) and the password itself.

If the PWD content is equal to the sent password, then the card will be unlocked and the card-locked status bit will be cleared in the status register. If the password is not correct, then the LOCK\_UNLOCK\_FAILED error bit will be set in the status register.

## 20.7. Specific operations

### 20.7.1. SD I/O specific operations

The SD I/O only card and SD I/O combo card support these specific operations:

**Read Wait operation**

**Suspend/resume operation**

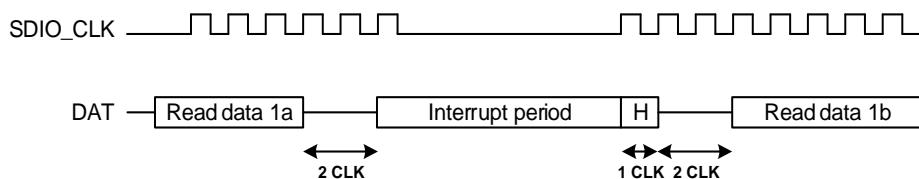
**Interrupts**

The SD I/O supports these operations only if the SDIO\_DATACTL[11] bit is set, except for read suspend that does not need specific hardware implementation.

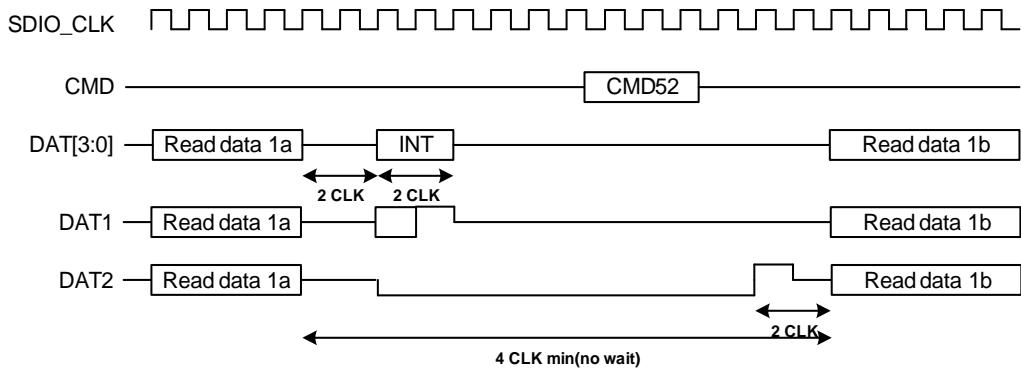
#### SD I/O read wait operation

The optional Read Wait (RW) operation is defined only for the SD 1-bit and 4-bit modes. The Read Wait operation allows a host to signal a card that is executing a read multiple (CMD53) operation to temporarily stall the data transfer while allowing the host to send commands to any function within the SD I/O card. To determine if a card supports the Read Wait protocol, the host shall test SRW capability bit in the Card Capability byte of the CCCR. The timing for Read Wait is based on the Interrupt Period. If a card does not support the Read Wait protocol, the only means a host has to stall (not abort) data in the middle of a read multiple command is to control the SDIO\_CLK. The limitation of this method is that with the clock stopped, the host cannot issue any commands, and so cannot perform other operations during the delay time. Read Wait support is mandatory for the card to support suspend/resume. [Figure 20-12. Read wait control by stopping SDIO\\_CLK](#) and [Figure 20-13. Read wait operation using SDIO\\_DAT\[2\]](#) show the Read Wait mode about stop the SDIO\_CLK and use SDIO\_DAT[2].

**Figure 20-12. Read wait control by stopping SDIO\_CLK**



**Figure 20-13. Read wait operation using SDIO\_DAT[2]**



We can start the Read Wait interval before the data block is received: when the data unit is enabled (SDIO\_DATACTL[0] bit set), the SD I/O specific operation is enabled (SDIO\_DATACTL[11] bit set), Read Wait starts (SDIO\_DATACTL[10] = 0 and SDIO\_DATACTL[8] = 1) and data direction is from card to SD I/O (SDIO\_DATACTL[1] = 1), the DSM directly moves from Idle to Read Wait. In Read Wait the DSM drives SDIO\_DAT[2] to 0 after 2 SDIO\_CLK clock cycles. In this state, when you set the RWSTOP bit (SDIO\_DATACTL[9]), the DSM remains in Wait for two more SDIO\_CLK clock cycles to drive SDIO\_DAT[2] to 1 for one clock cycle. The DSM then starts waiting again until it receives data from the card. The DSM will not start a Read Wait interval while receiving a block even if Read Wait start is set: the Read Wait interval will start after the CRC is received. The RWSTOP bit has to be cleared to start a new Read Wait operation. During the Read Wait interval, the SDIO can detect SD I/O interrupts on SDIO\_DAT[1].

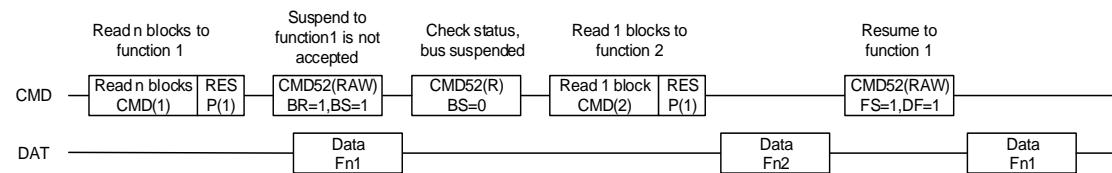
### SD I/O suspend/resume operation

Within a multi-function SD I/O or a Combo card, there are multiple devices (I/O and memory) that share access to the SD bus. In order to allow the sharing of access to the host among multiple devices, SD I/O and combo cards can implement the optional concept of suspend/resume. If a card supports suspend/resume, the host may temporarily halt a data transfer operation to one function or memory (suspend) in order to free the bus for a higher priority transfer to a different function or memory. Once this higher-priority transfer is completed, the original transfer is re-started where it left off (resume).

**Figure 20-14. Function2 read cycle inserted during function1 multiple read cycle** shows a condition where the first suspend request is not immediately accepted. The host then checks the status of the request with a read and determines that the bus has now been released (BS=0). At this time, a read to function 2 is started. Once that single block read is

completed, the resume is issued to function, causing the data transfer to resume (DF=1).

**Figure 20-14. Function2 read cycle inserted during function1 multiple read cycle**



When the host sends data to the card, the host can suspend the write operation. The SDIO\_CMDCTL[11] bit is set and indicates to the CSM that the current command is a suspend command. The CSM analyzes the response and when the response is received from the card (suspend accepted), it acknowledges the DSM that goes Idle after receiving the CRC token of the current block.

To suspend a read operation, the DSM waits in the WaitR state, when the function to be suspended sends a complete packet just before stopping the data transaction. The application should continue reading receive FIFO until the FIFO is empty, and the DSM goes Idle state automatically.

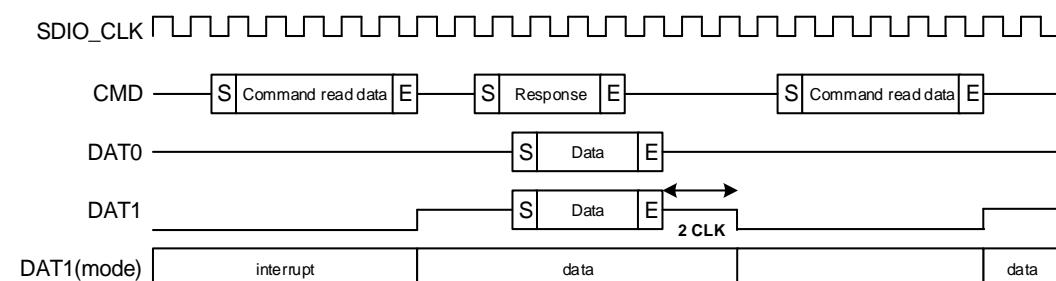
## Interrupts

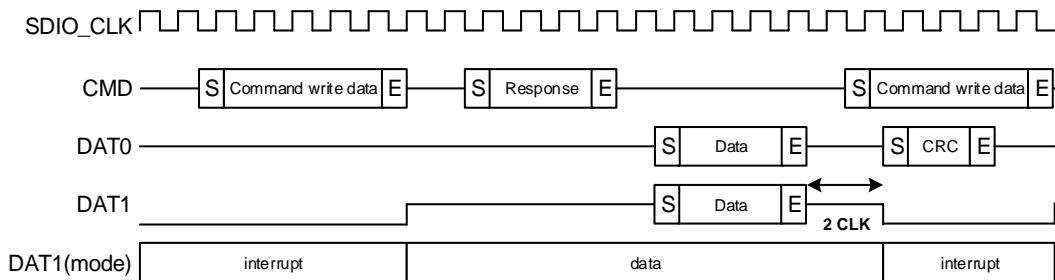
In order to allow the SD I/O card to interrupt the host, an interrupt function is added to a pin on the SD interface. Pin number 8, which is used as SDIO\_DAT[1] when operating in the 4-bit SD mode, is used to signal the card's interrupt to the host. The use of interrupt is optional for each card or function within a card. The SD I/O interrupt is “level sensitive”, that is, the interrupt line shall be held active (low) until it is either recognized and acted upon by the host or de-asserted due to the end of the Interrupt Period. Once the host has serviced the interrupt, it is cleared via function unique I/O operation.

When setting the SDIO\_DATACTL[11] bit SD I/O interrupts can detect on the SDIO\_DAT[1] line.

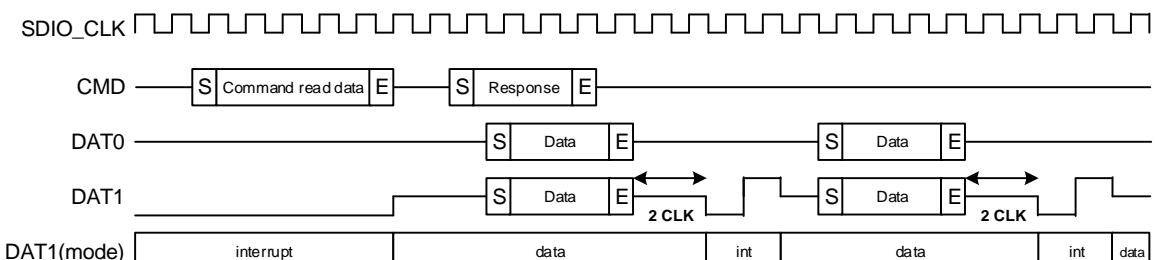
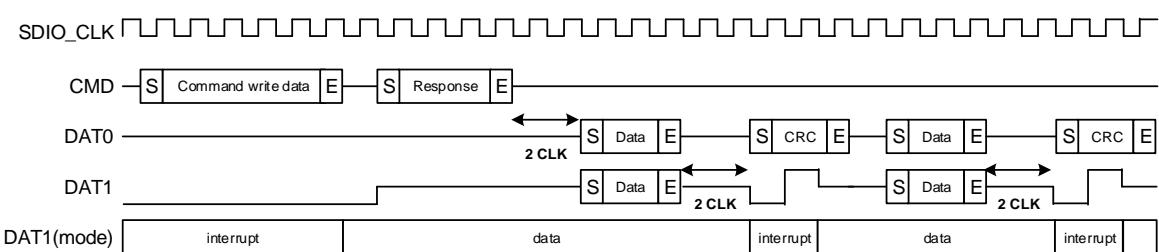
**Figure 20-15. Read Interrupt cycle timing** shows the timing of the interrupt period for single data transaction read cycles.

**Figure 20-15. Read Interrupt cycle timing**



**Figure 20-16. Write interrupt cycle timing**


When transferring multiple blocks of data in the 4-bit SD mode, a special definition of the interrupt period is required. In order to allow the highest speed of communication, the interrupt period is limited to a 2-clock interrupt period. Card that wants to send an interrupt signal to the host shall assert DAT1 low for the first clock and high for the second clock. The card shall then release DAT1 into the hi-Z State. [Figure 20-17. Multiple block 4-Bit read interrupt cycle timing](#) shows the operation for an interrupt during a 4-bit multi-block read and [Figure 20-18. Multiple block 4-Bit write interrupt cycle timing](#) shows the operation for an interrupt during a 4-bit multi-block write

**Figure 20-17. Multiple block 4-Bit read interrupt cycle timing**

**Figure 20-18. Multiple block 4-Bit write interrupt cycle timing**


## 20.7.2. CE-ATA specific operations

The CE-ATA device supports these specific operations:

**Receive command completion signal**

**Send command completion disable signal**

The SDIO supports these operations only when SDIO\_CMDCTL[14] is set.

## Command completion signal

CE-ATA defines a command completion signal that the device uses to notify the host upon normal ATA command completion or when ATA command termination has occurred due to an error condition the device has encountered.

If the ‘enable CMD completion’ bit SDIO\_CMDCTL[12] is set and the ‘not interrupt Enable’ bit SDIO\_CMDCTL[13] is reset, the CSM waits for the command completion signal in the Waitcompl state.

When start bit is received on the CMD line, the CSM enters the Idle state. No new command can be sent for 7 bit cycles. Then, for the last 5 cycles (out of the 7) the CMD line is driven to ‘1’ in push-pull mode.

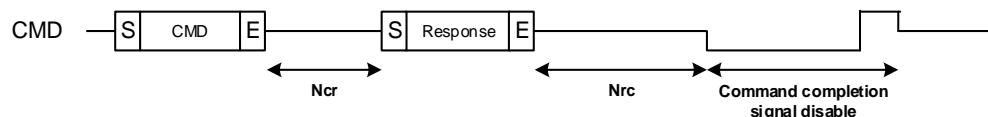
After the host detects a command completion signal from the device, it should issue a FAST\_IO (CMD39) command to read the ATA Status register to determine the ending status for the ATA command.

## Command completion disable signal

The host may cancel the ability for the device to return a command completion signal by issuing the command completion signal disable. The host shall only issue the command completion signal disable when it has received an R1b response for an outstanding RW\_MULTIPLE\_BLOCK (CMD61) command.

Command completion signal disable is sent 8 bit cycles after the reception of a short response if the ‘enable CMD completion’ bit, SDIO\_CMDCTL[12] is not set and the ‘not interrupt Enable’ bit SDIO\_CMDCTL[13] is reset.

**Figure 20-19. The operation for command completion disable signal**



## 20.8. SDIO registers

SDIO base address: 0x4001 8000

### 20.8.1. Power control register (SDIO\_PWRCTL)

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															PWRCTL[1:0]

rw

Bits	Fields	Descriptions
31:2	Reserved	Must be kept at reset value
1:0	PWRCTL[1:0]	<p>SDIO power control bits.</p> <p>These bits control the SDIO state, card input or output.</p> <p>00: SDIO power off: SDIO cmd/data state machine reset to IDLE, clock to card stopped, no cmd/data output to card</p> <p>01: Reserved</p> <p>10: Reserved</p> <p>11: SDIO Power on</p>

**Note:** Between Two write accesses to this register, it needs at least 3 SDIOCLK + 2 pclk2 which used to sync the registers to SDIOCLK clock domain.

### 20.8.2. Clock control register (SDIO\_CLKCTL)

Address offset: 0x04

Reset value: 0x0000 0000

This register controls the output clock SDIO\_CLK.

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
DIV[8]	Reserved														
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	HWCLKEN	CLKEDGE	CLKEDGE	BUSMODE[1:0]	CLKBYP	CLKPWRSAV	CLKEN	DIV[7:0]							
	rw	rw	rw	rw	rw	rw	rw	rw							

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31	DIV[8]	MSB of Clock division  This field defines the MSB division between the input clock (SDIOCLK) and the output clock, refer to bit 7:0 of SDIO_CLKCTL
30:15	Reserved	Must be kept at reset value
14	HWCLKEN	Hardware Clock Control enable bit  If this bit is set, hardware controls the SDIO_CLK on/off depending on the system bus is very busy or not. There is no underrun/overrun error when this bit is set, because hardware can close the SDIO_CLK when almost underrun/overrun. 0: HW Clock control is disabled 1: HW Clock control is enabled
13	CLKEDGE	SDIO_CLK clock edge selection bit  0: Select the rising edge of the SDIOCLK to generate SDIO_CLK 1: Select the falling edge of the SDIOCLK to generate SDIO_CLK
12:11	BUSMODE[1:0]	SDIO card bus mode control bit  00: 1-bit SDIO card bus mode selected 01: 4-bit SDIO card bus mode selected 10: 8-bit SDIO card bus mode selected
10	CLKBYP	Clock bypass enable bit  This bit defines the SDIO_CLK is directly SDIOCLK or not. 0: NO bypass, the SDIO_CLK refers to DIV bits in SDIO_CLKCTL register. 1: Clock bypass, the SDIO_CLK is directly from SDIOCLK (SDIOCLK/1).
9	CLKPWRSAV	SDIO_CLK clock dynamic switch on/off for power saving.  This bit controls SDIO_CLK clock dynamic switch on/off when the bus is idle for power saving 0: SDIO_CLK clock is always on 1: SDIO_CLK closed when bus idle
8	CLKEN	SDIO_CLK clock output enable bit  0: SDIO_CLK is disabled 1: SDIO_CLK is enabled
7:0	DIV[7:0]	Clock division  This field and DIV[8] bit defines the division factor to generator SDIO_CLK clock to card. The SDIO_CLK is divider from SDIOCLK if CLKBYP bit is 0, and the SDIO_CLK frequency = SDIOCLK / (DIV[8:0] + 2).

**Note:** Between Two write accesses to this register, it needs at least 3 SDIOCLK + 2 pclk2 which used to sync the registers to SDIOCLK clock domain.

### 20.8.3. Command argument register (SDIO\_CMDAGMT)

Address offset: 0x08

Reset value: 0x0000 0000

This register defines 32 bit command argument, which will be used as part of the command (bit 39 to bit 8).

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

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

Bits	Fields	Descriptions
31:0	CMDAGMT[31:0]	<p>SDIO card command argument</p> <p>This field defines the SDIO card command argument which sent to card. This field is the bits [39:8] of command message. If the command message contains an argument, this field must update before writing SDIO_CMDCTL register when sending a command.</p>

### 20.8.4. Command control register (SDIO\_CMDCTL)

Address offset: 0x0C

Reset value: 0x0000 0000

The SDIO\_CMDCTL register contains the command index and other command control bits to control command state machine.

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															
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	ATAEN	NINTEN	ENCMDC	SUSPEND	CSMEN	WAITDEN	INTWAIT	CMDRESP[1:0]	CMDIDX[5:0]						
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw						

Bits	Fields	Descriptions
31:15	Reserved	Must be kept at reset value
14	ATAEN	<p>CE-ATA command enable(CE-ATA only)</p> <p>If this bit is set, the host enters the CE-ATA mode, and the CSM transfers CMD61.</p> <p>0: CE-ATA disable</p> <p>1: CE-ATA enable</p>

13	NINTEN	No CE-ATA Interrupt (CE-ATA only)  This bit defines if there is CE-ATA interrupt or not. This bit is only used when CE-ATA card.  0: CE-ATA interrupt enable 1: CE_ATA interrupt disable
12	ENCMDC	CMD completion signal enabled (CE-ATA only)  This bit defines if there is command completion signal or not in CE-ATA card.  0: no completion signal 1: have completion signal
11	SUSPEND	SD I/O suspend command(SD I/O only)  This bit defines whether the CSM to send a suspend command or not. This bit is only used for SDIO card.  0: no effect 1: suspend command
10	CSMEN	Command state machine (CSM) enable bit  0: Command state machine disable (stay on CS_Idle) 1: Command state machine enable
9	WAITDEND	Waits for ends of data transfer.  If this bit is set, the command state machine starts to send a command must wait the end of data transfer.  0: no effect 1: Wait the end of data transfer
8	INTWAIT	Interrupt wait instead of timeout  This bit defines the command state machine to wait card interrupt at CS_Wait state in command state machine. If this bit is set, no command wait timeout generated.  0: Not wait interrupt. 1: Wait interrupt.
7:6	CMDRESP[1:0]	Command response type bits  These bits define the response type after sending a command message.  00: No response 01: Short response 10: No response 11: Long response
5:0	CMDIDX[5:0]	Command index  This field defines the command index to be sent to SDIO card.

**Note:** Between Two write accesses to this register, it needs at least 3 SDIOCLK + 2 pclk2 which used to sync the registers to SDIOCLK clock domain.

### 20.8.5. Command index response register (SDIO\_RSPCMDIDX)

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								RSPCMDIDX[5:0]							

r

Bits	Fields	Descriptions
31:6	Reserved	Must be kept at reset value
5:0	RSPCMDIDX[5:0]	Last response command index  Read-only bits field. This field contains the command index of the last command response received. If the response doesn't have the command index (long response and short response of R3), the content of this register is undefined.

### 20.8.6. Response register (SDIO\_RESPx x=0..3)

Address offset: 0x14+(4\*x), x=0..3

Reset value: 0x0000 0000

These register contains the content of the last card response received.

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

Bits	Fields	Descriptions
31:0	RESPx[31:0]	Card state. The content of the response, see <a href="#">Table 20-32. SDIO_RESPx register at different response type</a> .

The short response is 32 bits, the long response is 127 bits (bit 128 is the end bit 0).

**Table 20-32. SDIO\_RESPx register at different response type**

Register	Short response	Long response
SDIO_RESP0	Card response[31:0]	Card response[127:96]
SDIO_RESP1	reserved	Card response [95:64]

<b>Register</b>	<b>Short response</b>	<b>Long response</b>
SDIO_RESP2	reserved	Card response [63:32]
SDIO_RESP3	reserved	Card response [31:1],plus bit 0

### 20.8.7. Data timeout register (SDIO\_DATATO)

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

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:0	DATATO[31:0]	<p>Data timeout period</p> <p>These bits define the data timeout period count by SDIO_CLK. When the DSM enter the state WaitR or BUSY, the internal counter which loads from this register starts decrement. The DSM timeout and enter the state Idle and set the DTTMOUT flag when the counter decreases to 0.</p>

**Note:** The data timer register and the data length register must be updated before being written to the data control register when need a data transfer.

### 20.8.8. Data length register (SDIO\_DATALEN)

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								DATALEN[24:16]							
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATALEN[15:0]								rw							

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:25	Reserved	Must be kept at reset value
24:0	DATALEN[24:0]	<p>Data transfer length</p> <p>This register defined the number of bytes to be transferred. When the data transfer</p>

starts, the data counter loads this register and starts decrement.

**Note:** If block data transfer selected, the content of this register must be a multiple of the block size (refer to SDIO\_DATACTL). The data timer register and the data length register must be updated before being written to the data control register when need a data transfer.

### 20.8.9. Data control register (SDIO\_DATACTL)

Address offset: 0x2C

Reset value: 0x0000 0000

This register controls the DSM.

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		IOEN	RWTYPE	RWSTOP	RWEN		BLKSZ[3:0]		DMAEN	TRANSMOD	DATADIR	DATAEN			

rw      rw

Bits	Fields	Descriptions
31:12	Reserved	Must be kept at reset value
11	IOEN	SD I/O specific function enable(SD I/O only) 0: Not SD I/O specific function 1: SD I/O specific function
10	RWTYPE	Read wait type(SD I/O only) 0: Read Wait control using SDIO_DAT[2] 1: Read Wait control by stopping SDIO_CLK
9	RWSTOP	Read wait stop(SD I/O only) 0: No effect 1: Stop the read wait process if RWEN bit is set
8	RWEN	Read wait mode enabled(SD I/O only) 0: Read wait mode is disabled 1: Read wait mode is enabled
7:4	BLKSZ[3:0]	Data block size These bits defined the block size when data transfer is block transfer. 0000: block size = $2^0$ = 1 byte 0001: block size = $2^1$ = 2 bytes 0010: block size = $2^2$ = 4 bytes 0011: block size = $2^3$ = 8 bytes 0100: block size = $2^4$ = 16 bytes

0101: block size =  $2^5$  = 32 bytes  
 0110: block size =  $2^6$  = 64 bytes  
 0111: block size =  $2^7$  = 128 bytes  
 1000: block size =  $2^8$  = 256 bytes  
 1001: block size =  $2^9$  = 512 bytes  
 1010: block size =  $2^{10}$  = 1024 bytes  
 1011: block size =  $2^{11}$  = 2048 bytes  
 1100: block size =  $2^{12}$  = 4096 bytes  
 1101: block size =  $2^{13}$  = 8192 bytes  
 1110: block size =  $2^{14}$  = 16384 bytes  
 1111: reserved

3	DMAEN	DMA enable bit 0: DMA is disabled. 1: DMA is enabled.
2	TRANSMOD	Data transfer mode 0: Block transfer 1: Stream transfer or SDIO multibyte transfer
1	DATADIR	Data transfer direction 0: Write data to card. 1: Read data from card.
0	DATAEN	Data transfer enable bit Write 1 to this bit to start data transfer regardless this bit is 0 or 1. The DSM moves to Readwait state if RWEN is set or to the WaitS, WaitR state depend on DATADIR bit. Start a new data transfer, it not need to clear this bit to 0.

**Note:** Between Two write accesses to this register, it needs at least 3 SDIOCLK + 2 pclk2 which used to sync the registers to SDIOCLK clock domain.

### 20.8.10. Data counter register (SDIO\_DATACNT)

Address offset: 0x30

Reset value: 0x0000 0000

This register is read only. When the DSM from Idle to WaitR or WaitS, it loads value from data length register (SDIO\_DATALEN). It decrements with the data transferred, when it reaches 0, the flag DTEND is 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
Reserved								DATACNT[24:16]							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATACNT[15:0]								r							

Bits	Fields	Descriptions
31:25	Reserved	Must be kept at reset value
24:0	DATACNT[24:0]	Data count value Read-only bits field. When these bits are read, the number of remaining data bytes to be transferred is returned.

### 20.8.11. Status register (SDIO\_STAT)

Address offset: 0x34

Reset value: 0x0000 0000

This register is read only. The following descripts the types of flag:

The flags of bit [23:22, 10:0] can only be cleared by writing 1 to the corresponding bit in interrupt clear register (SDIO\_INTC).

The flags of bit [21:11] are changing depend on the hardware logic.

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								ATAEND	SDIOINT	RXDTVAL	TXDTVAL	RFE	TFE	RFF	TFF
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
RFH	TFH	RXRUN	TXRUN	CMDRUN	DTBLKE ND	STBITE	DTEND	CMDSEN D	CMDREC V	RXORE	TXURE	DTTMOU T	CMDTMO UT	DTCRCE RR	CCRCER R
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits	Fields	Descriptions
31:24	Reserved	Must be kept at reset value
23	ATAEND	CE-ATA command completion signal received (only for CMD61)
22	SDIOINT	SD I/O interrupt received
21	RXDTVAL	Data is valid in receive FIFO
20	TXDTVAL	Data is valid in transmit FIFO
19	RFE	Receive FIFO is empty
18	TFE	Transmit FIFO is empty When HW Flow control is enabled, TFE signals becomes activated when the FIFO contains 2 words.
17	RFF	Receive FIFO is full When HW Flow control is enabled, RFF signals becomes activated 2 words before the FIFO is full.

16	TFF	Transmit FIFO is full
15	RFH	Receive FIFO is half full: at least 8 words can be read in the FIFO
14	TFH	Transmit FIFO is half empty: at least 8 words can be written into the FIFO
13	RXRUN	Data reception in progress
12	TXRUN	Data transmission in progress
11	CMDRUN	Command transmission in progress
10	DTBLKEND	Data block sent/received (CRC check passed)
9	STBITE	Start bit error in the bus.
8	DTEND	Data end (data counter, SDIO_DATACNT, is zero)
7	CMDSEND	Command sent (no response required)
6	CMDRECV	Command response received (CRC check passed)
5	RXORE	Received FIFO overrun error occurs
4	TXURE	Transmit FIFO underrun error occurs
3	DTTMOUT	Data timeout The data timeout period depends on the SDIO_DATATO register.
2	CMDTMOUT	Command response timeout The command timeout period has a fixed value of 64 SDIO_CLK clock periods.
1	DTCRCERR	Data block sent/received (CRC check failed)
0	CCRCERR	Command response received (CRC check failed)

### 20.8.12. Interrupt clear register (SDIO\_INTC)

Address offset: 0x38

Reset value: 0x0000 0000

This register is write only. Writing 1 to the bit can clear the corresponding bit in the SDIO\_STAT register.

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						ATAEND C	SDIOINT C	Reserved													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0						
Reserved					DTBLKE NDC	STBITEC	DTENDC	CMDSEN DC	CMDREC VC	RXOREC	TXUREC	DTTMOU TC	CMDETMO UTC	DTCRCE RRC	CCRCE RC						

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:24	Reserved	Must be kept at reset value
23	ATAENDC	ATAEND flag clear bit Write 1 to this bit to clear the flag.
22	SDIOINTC	SDIOINT flag clear bit Write 1 to this bit to clear the flag.
21:11	Reserved	Must be kept at reset value
10	DTBLKENDC	DTBLKEND flag clear bit Write 1 to this bit to clear the flag.
9	STBITEC	STBITE flag clear bit Write 1 to this bit to clear the flag.
8	DTENDC	DTEND flag clear bit Write 1 to this bit to clear the flag.
7	CMDSENDC	CMDSEND flag clear bit Write 1 to this bit to clear the flag.
6	CMDRECVC	CMDRECV flag clear bit Write 1 to this bit to clear the flag.
5	RXOREC	RXORE flag clear bit Write 1 to this bit to clear the flag.
4	TXUREC	TXURE flag clear bit Write 1 to this bit to clear the flag.
3	DTTMOUTC	DTTMOUT flag clear bit Write 1 to this bit to clear the flag.
2	CMDTMOUTC	CMDTMOUT flag clear bit Write 1 to this bit to clear the flag.
1	DTCRCERRC	DTCRCERR flag clear bit Write 1 to this bit to clear the flag.
0	CCRCERRC	CCRCERR flag clear bit Write 1 to this bit to clear the flag.

### 20.8.13. Interrupt enable register (SDIO\_INTEN)

Address offset: 0x3C

Reset value: 0x0000 0000

This register enables the corresponding interrupt in the SDIO\_STAT register.

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						ATAENDI E	SDIOINTI LIE	RXDTVA IE	TXDTVAL IE	RFEIE	TFEIE	RFFIE	TFFIE		
						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
RFHIE	TFHIE	RXRUNIE	TXRUNIE	CMDRUNIE	DTBLKENDIE	STBITEIE	DTENDIE	CMDSENIE	CMDRECIE	RXOREIE	TXUREIE	DTTMOUTIE	CMDTMOUTIE	DTCRCERIE	CCRCERIE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:24	Reserved	Must be kept at reset value
23	ATAENDIE	CE-ATA command completion signal received interrupt enable Write 1 to this bit to enable the interrupt.
22	SDIOINTIE	SD I/O interrupt received interrupt enable Write 1 to this bit to enable the interrupt.
21	RXDTVALIE	Data valid in receive FIFO interrupt enable Write 1 to this bit to enable the interrupt.
20	TXDTVALIE	Data valid in transmit FIFO interrupt enable Write 1 to this bit to enable the interrupt.
19	RFEIE	Receive FIFO empty interrupt enable Write 1 to this bit to enable the interrupt.
18	TFEIE	Transmit FIFO empty interrupt enable Write 1 to this bit to enable the interrupt.
17	RFFIE	Receive FIFO full interrupt enable Write 1 to this bit to enable the interrupt.
16	TFFIE	Transmit FIFO full interrupt enable Write 1 to this bit to enable the interrupt.
15	RFHIE	Receive FIFO half full interrupt enable Write 1 to this bit to enable the interrupt.
14	TFHIE	Transmit FIFO half empty interrupt enable Write 1 to this bit to enable the interrupt.
13	RXRUNIE	Data reception interrupt enable Write 1 to this bit to enable the interrupt.
12	TXRUNIE	Data transmission interrupt enable Write 1 to this bit to enable the interrupt.
11	CMDRUNIE	Command transmission interrupt enable Write 1 to this bit to enable the interrupt.
10	DTBLKENDIE	Data block end interrupt enable

		Write 1 to this bit to enable the interrupt.
9	STBITEIE	Start bit error interrupt enable Write 1 to this bit to enable the interrupt.
8	DTENDIE	Data end interrupt enable Write 1 to this bit to enable the interrupt.
7	CMDSENDIE	Command sent interrupt enable Write 1 to this bit to enable the interrupt.
6	CMDRECVIE	Command response received interrupt enable Write 1 to this bit to enable the interrupt.
5	RXOREIE	Received FIFO overrun error interrupt enable Write 1 to this bit to enable the interrupt.
4	TXUREIE	Transmit FIFO underrun error interrupt enable Write 1 to this bit to enable the interrupt.
3	DTTMOUTIE	Data timeout interrupt enable Write 1 to this bit to enable the interrupt.
2	CMDTMOUTIE	Command response timeout interrupt enable Write 1 to this bit to enable the interrupt.
1	DTCRCERRIE	Data CRC fail interrupt enable Write 1 to this bit to enable the interrupt.
0	CCRCERRIE	Command response CRC fail interrupt enable Write 1 to this bit to enable the interrupt.

#### 20.8.14. FIFO counter register (SDIO\_FIFOCNT)

Address offset: 0x48

Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31:24	Reserved	Must be kept at reset value
23:0	FIFOCNT[23:0]	FIFO counter.

These bits define the remaining number words to be written or read from the FIFO.  
 It loads the data length register (SDIO\_DATALEN[24:2] if SDIO\_DATALEN is word-aligned or SDIO\_DATALEN[24:2]+1 if SDIO\_DATALEN is not word-aligned) when DATAEN is set, and start count decrement when a word write to or read from the FIFO.

### **20.8.15. FIFO data register (SDIO\_FIFO)**

Address offset: 0x80

Reset value: 0x0000 0000

This register occupies 32 entries of 32-bit words, the address offset is from 0x80 to 0xFC.

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

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:0	FIFODT[31:0]	<p>Receive FIFO data or transmit FIFO data</p> <p>These bits are the data of receive FIFO or transmit FIFO. Write to or read from this register is write data to FIFO or read data from FIFO.</p>

## 21. External memory controller (EXMC)

### 21.1. Overview

The external memory controller EXMC, is used as a translator for MCU to access a variety of external memory. By configuring the related registers, it can automatically convert AMBA memory access protocol into a specific memory access protocol, such as SRAM, ROM, NOR Flash, NAND Flash and PC Card. Users can also adjust the timing parameters in the configuration registers to improve 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.

### 21.2. Characteristics

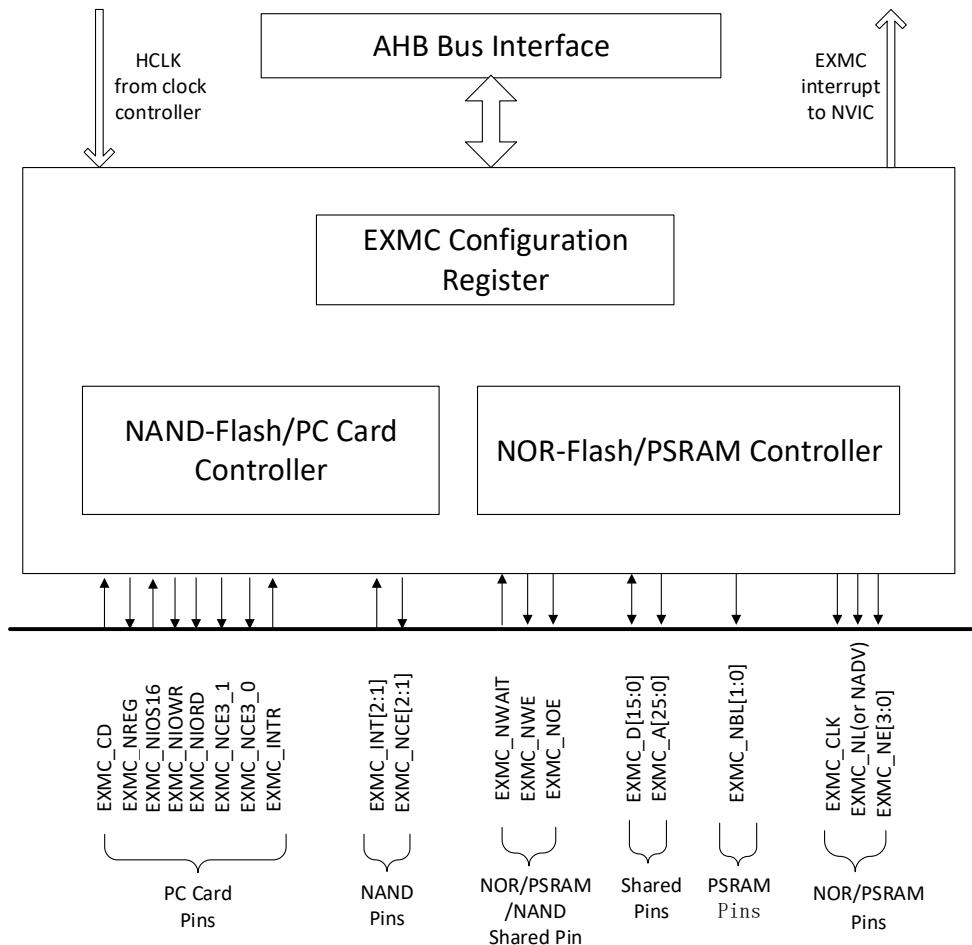
- Supported external memory:
  - SRAM
  - PSRAM
  - ROM
  - NOR Flash
  - 8-bit or 16-bit NAND Flash
  - 16-bit PC Card
- Protocol translation between the AMBA and the multitude of external memory protocol.
- Offering a variety of programmable timing parameters to meet user's specific needs.
- Each bank has its own chip-select signal which can be configured independently.
- Independent read/write timing configuration to a sub-set memory type.
- Embedded ECC hardware for NAND Flash access.
- 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.

### 21.3. Function overview

#### 21.3.1. Block diagram

EXMC is the combination of five modules: The AHB bus interface, EXMC configuration registers, NOR/PSRAM controller, NAND/PC Card controller and external device interface. AHB clock (HCLK) is the reference clock.

Figure 21-1. The EXMC block diagram



### 21.3.2. 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 follows 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 will 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.

### 21.3.3. External device address mapping

**Figure 21-2. EXMC memory banks**

Address	Banks	Supported memory type
0x6000 0000	Bank0(4x64M)	NOR/PSRAM
0x6FFF FFFF		
0x7000 0000	Bank1(256M)	NAND
0x7FFF FFFF		
0x8000 0000	Bank2(256M)	PC Card
0x8FFF FFFF		
0x9000 0000	Bank3(256M)	
0x9FFF FFFF		

EXMC access space is divided into multiple banks. Each bank is 256 Mbytes. The first bank (Bank0) is further divided into four regions, and each region is 64 Mbytes. Bankx(x=1, 2) is divided into two spaces, the attribute memory space and the common memory space. Bank3 is divided into three spaces, which are the attribute memory space, the common memory space and the I/O memory space.

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.

Bank1 and bank2 are used to access NAND Flash exclusively.

Bank3 is used for PC Card access.

#### NOR/PSRAM address mapping

[\*\*Figure 21-3. Four regions of bank0 address mapping\*\*](#) reflects the address mapping of the four regions of bank0. Internal AHB address lines HADDR [27:26] bit are used to select the four regions.

**Figure 21-3. Four regions of bank0 address mapping**

HADDR[27:26]	Address	Regions	Supported memory type
00	{ 0x60000000 0x63FF FFFF 0x64000000	Region0	NOR/PSRAM
01	{ 0x67FF FFFF 0x68000000	Region1	NOR/PSRAM
10	{ 0x6BFF FFFF 0x6C000000	Region2	NOR/PSRAM
11	{ 0x6FFF FFFF	Region3	NOR/PSRAM

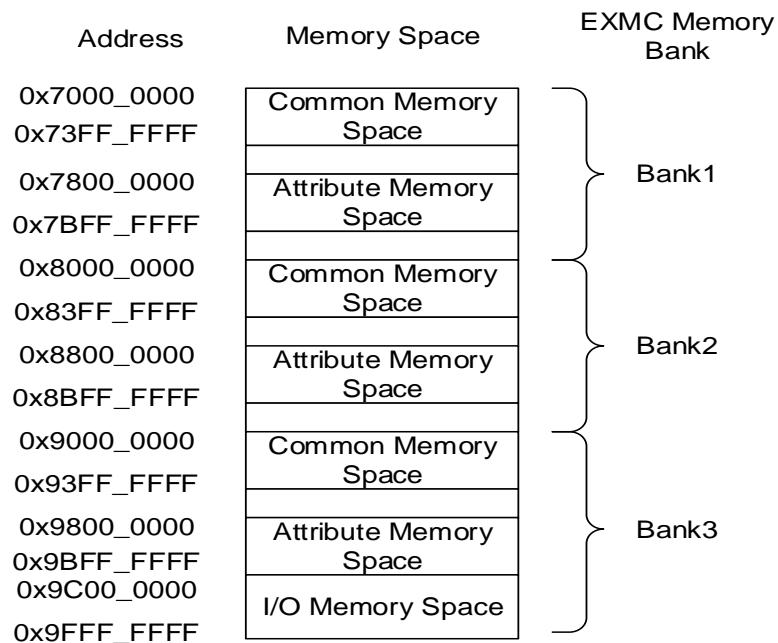
HADDR[25:0] is the byte address whereas the external memory may not be byte accessed, this will lead to address inconsistency. EXMC can adjust HADDR to accommodate the data width of the external memory according to the following rules.

- 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.

### NAND/PC Card address mapping

Bank1 and bank2 are designed to access NAND Flash, and bank3 is designed to access PC Card. Each bank is further divided into several memory spaces as shown in [Figure 21-4. NAND/PC Card address mapping.](#)

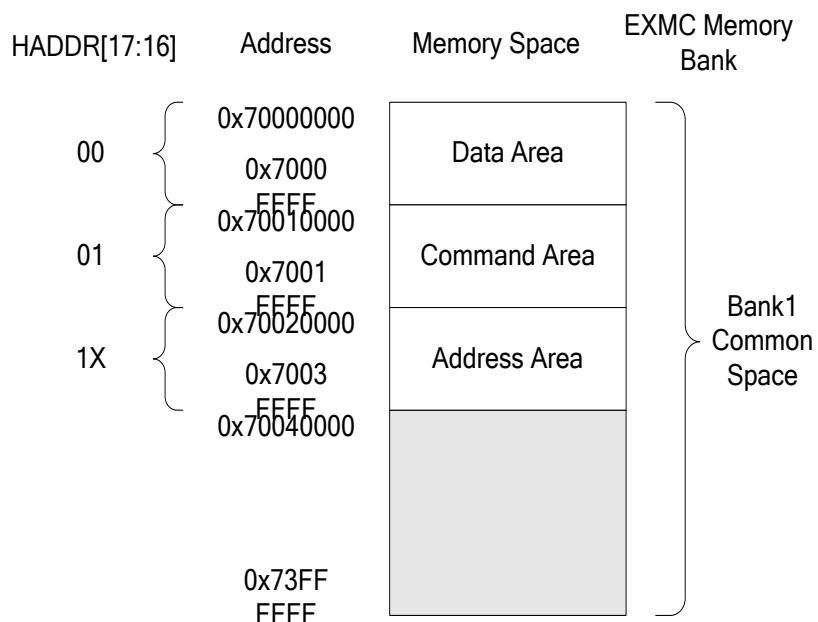
Figure 21-4. NAND/PC Card address mapping



### NAND address mapping

For NAND Flash, the common space and the attribute space are further-divided into three areas individually, the data area, the command area and the address area as shown in [Figure 21-5. Diagram of bank1 common space.](#)

Figure 21-5. Diagram of bank1 common space



HADDR [17:16] bits are used to select one of the three areas.

- When HADDR [17:16] = 00, the data area is selected.
- When HADDR [17:16] = 01, the command area is selected.
- When HADDR [17:16] = 1X, the address area is selected.

Application software uses these three areas to access NAND Flash, their definitions are as follows.

- Address area: This area is where the NAND Flash access address should be issued by software, the EXMC will pull the address latch enable (ALE) signal automatically in address transfer phase. ALE is mapped to EXMC\_A [17].
- Command area: This area is where the NAND Flash access command should be issued by the software, the EXMC will pull the command latch enable (CLE) signal automatically in command transfer phase. CLE is mapped to EXMC\_A [16].
- Data area: This area is where the NAND Flash read/write data should be accessed. When the EXMC is in data transfer mode, software should write the data to be transferred to the NAND Flash in this area. When the EXMC is in data reception mode, software should read the data from the NAND Flash by reading this area. Data access address is incremented automatically in consecutive mode, users need not to be concerned with access address area.

#### **21.3.4. 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 4 independent chip-select signals for each of the 4 sub-banks within bank0, named NE[x] (x = 0, 1, 2, 3). 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).

In synchronous mode, all output data of controller will change on the fall edge of external memory device clock (EXMC\_CLK).

#### **NOR/PSRAM memory device interface description**

**Table 21-1. NOR Flash interface signals description**

<b>EXMC Pin</b>	<b>Direction</b>	<b>Mode</b>	<b>Functional description</b>
EXMC_CLK	Output	Sync	Clock signal for sync
Non-muxed EXMC_A[25:0]	Output	Async/Sync	Address bus signal
Muxed EXMC_A[25:16]			
EXMC_D[15:0]	Input/output	Async/Sync	Address/Data bus

<b>EXMC Pin</b>	<b>Direction</b>	<b>Mode</b>	<b>Functional description</b>
		(muxed)	
	Input/output	Async/Sync (non-muxed)	Data bus
EXMC_NE[x]	Output	Async/Sync	Chip selection, x=0/1/2/3
EXMC_NOE	Output	Async/Sync	Read enable
EXMC_NWE	Output	Async/Sync	Write enable
EXMC_NWAIT	Input	Async/Sync	Wait input signal
EXMC_NL(NADV)	Output	Async/Sync	Address valid

**Table 21-2. PSRAM non-muxed signal description**

<b>EXMC Pin</b>	<b>Direction</b>	<b>Mode</b>	<b>Functional description</b>
EXMC_CLK	Output	Sync	Clock signal for sync
EXMC_A[25:0]	Output	Async/Sync	Address Bus
EXMC_D[15:0]	Input/output	Async/Sync	Data Bus
EXMC_NE[x]	Output	Async/Sync	Chip selection, x=0/1/2/3
EXMC_NOE	Output	Async/Sync	Read enable
EXMC_NWE	Output	Async/Sync	Write enable
EXMC_NWAIT	Input	Async/Sync	Wait input signal
EXMC_NL(NADV)	Output	Async/Sync	Latch enable (address valid enable, NADV)
EXMC_NBL[1]	Output	Async/Sync	Upper byte enable
EXMC_NBL[0]	Output	Async/Sync	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 21-3. EXMC bank 0 supports all transactions**

<b>Memory</b>	<b>Access Mode</b>	<b>R/W</b>	<b>AHB Transaction Size</b>	<b>Memory Transaction Size</b>	<b>Comments</b>
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
	Sync	R	16	16	
	Sync	R	32	16	
PSRAM	Async	R	8	16	

<b>Memory</b>	<b>Access Mode</b>	<b>R/W</b>	<b>AHB Transaction Size</b>	<b>Memory Transaction Size</b>	<b>Comments</b>
SRAM and ROM	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
	Sync	R	16	16	
	Sync	R	32	16	
	Sync	W	8	16	Use of byte lanes EXMC_NBL[1:0]
	Sync	W	16	16	
	Sync	W	32	16	
NOR Flash/PSRAM	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 21-4. NOR / PSRAM controller timing parameters**

Parameter	Function	Access mode	Unit	Min	Max
CKDIV	Sync Clock divide ratio	Sync	HCLK	2	16
DLAT	Data latency	Sync	EXMC_CLK	2	17
BUSLAT	Bus latency	Async/Sync read	HCLK	1	16

Parameter	Function	Access mode	Unit	Min	Max
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

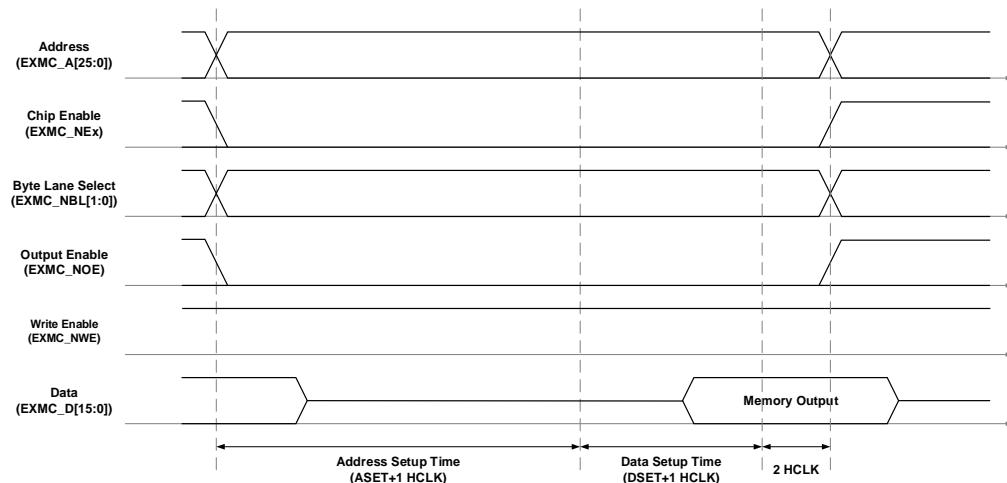
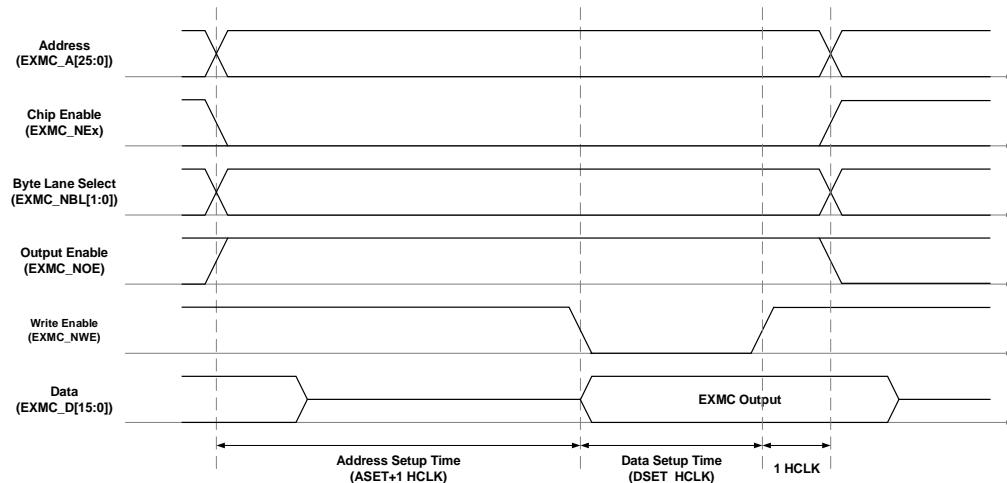
**Table 21-5. EXMC\_timing models**

Timing model		Extend mode	Mode description	Write timing parameter	Read timing parameter
Async	Mode 1	0	SRAM/PSRAM/CRAM	DSET ASET	DSET ASET
	Mode 2	0	NOR Flash	DSET ASET	DSET ASET
	Mode A	1	SRAM/PSRAM/CRAM with EXMC_NOE toggling on data phase	WDSET WASET	DSET ASET
	Mode B	1	NOR Flash	WDSET WASET	DSET ASET
	Mode C	1	NOR Flash with EXMC_NOE toggling on data phase	WDSET WASET	DSET ASET
	Mode D	1	With address hold capability	WDSET WAHLD WASET	DSET AHLD ASET
	Mode AM	0	NOR Flash address/data mux	DSET AHLD ASET BUSLAT	DSET AHLD ASET BUSLAT
Sync	Mode E	0	NOR/PSRAM/CRAM synchronous read PSRAM/CRAM synchronous write	DLAT CKDIV	DLAT CKDIV
	Mode SM	0	NOR Flash address/data mux	DLAT CKDIV	DLAT CKDIV

As shown in [Table 21-5. EXMC\\_timing models](#), EXMC NOR Flash / PSRAM controller provides a variety of timing model, users can modify those parameters listed in [Table 21-4. NOR / PSRAM controller timing parameters](#) to satisfy different external memory type and user's requirements. When extended mode is enabled via the EXMODEN bit in EXMC\_SNCTLx register, different timing patterns for read and write access could be generated independently according to EXMC\_SNTCFGx and EXMC\_SNWTCFGx register's configuration.

### Asynchronous access timing diagram

Mode 1 - SRAM/CRAM

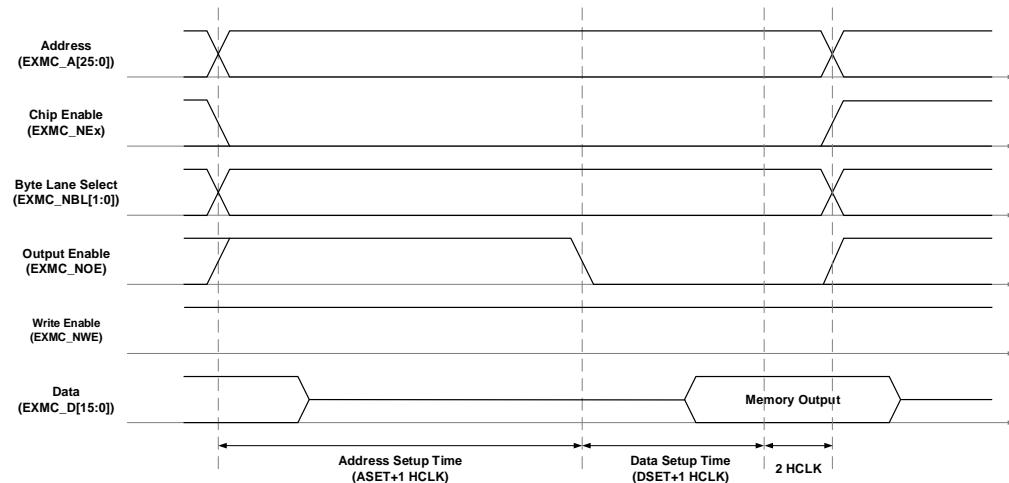
**Figure 21-6. Mode 1 read access**

**Figure 21-7. Mode 1 write access**

**Table 21-6. Mode 1 related registers configuration**

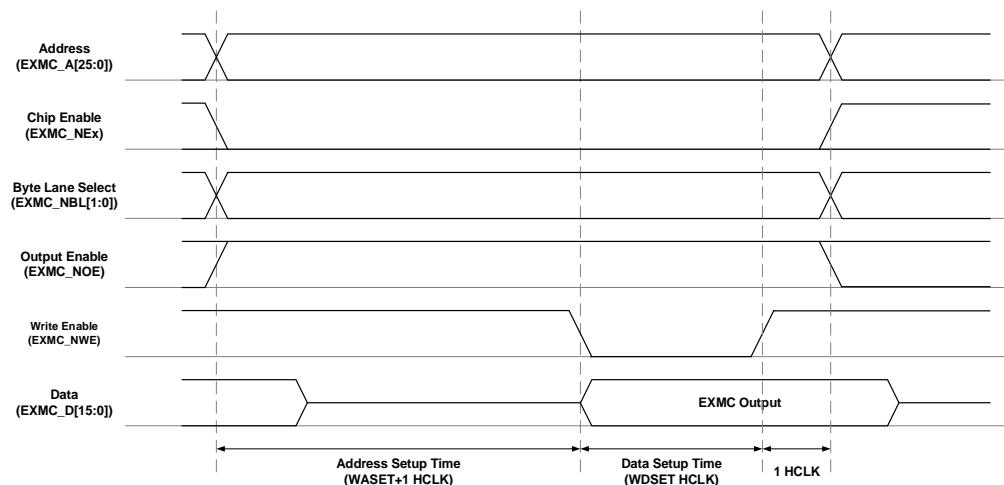
Bit Position	Bit Name	Reference Setting Value
<b>EXMC_SNCTLx</b>		
31-20	Reserved	0x000
19	SYNCWR	0x0
18-16	CPS	0x0
15	ASYNCWAIT	Depends on memory
14	EXMODEN	0x0
13	NRWTEN	0x0
12	WEN	Depends on user
11	NRWTCFG	No effect
10	WRAPEN	0x0
9	NRWTPOL	Meaningful only when the bit 15 is set to 1
8	SBRSTEN	0x0
7	Reserved	0x1

Bit Position	Bit Name	Reference Setting Value
6	NREN	No effect
5-4	NRW	Depends on memory
3-2	NRTP	Depends on memory, except 2(Nor Flash)
1	NRMUX	0x0
0	NRBKEN	0x1
<b>EXMC_SNTCFGx</b>		
31-30	Reserved	0x0000
29-28	ASYNCMOD	No effect
27-24	DLAT	No effect
23-20	CKDIV	No effect
19-16	BUSLAT	Time between EXMC_NE[x] rising edge to EXMC_NE[x] falling edge
15-8	DSET	Depends on memory and user (DSET+1 HCLK for write, DSET+3 HCLK for read)
7-4	AHLD	No effect
3-0	ASET	Depends on memory and user

Mode A - SRAM/PSRAM(CRAM) OE toggling

**Figure 21-8. Mode A read access**



**Figure 21-9. Mode A write access**


The different between mode A and mode 1 write timing is that read/write timing is specified by the same set of timing configuration, while mode A write timing configuration is independent of its read configuration.

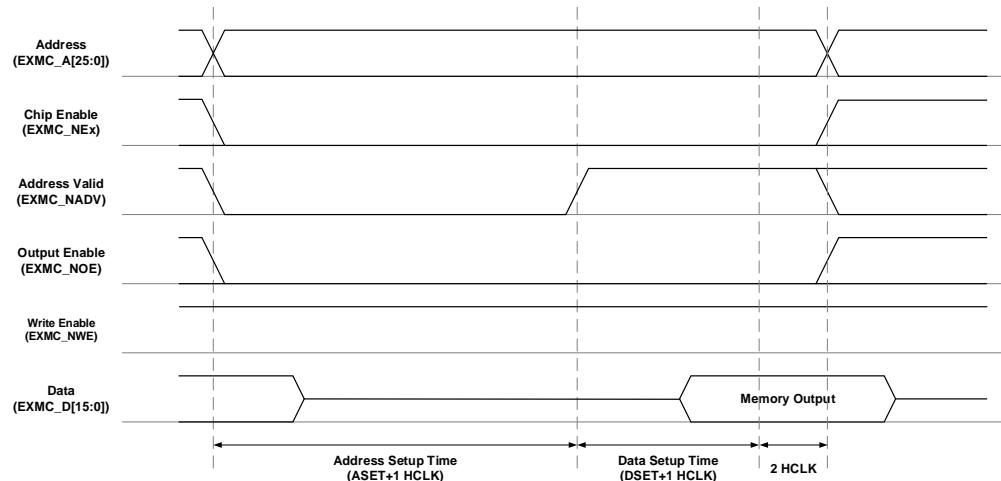
**Table 21-7. Mode A related registers configuration**

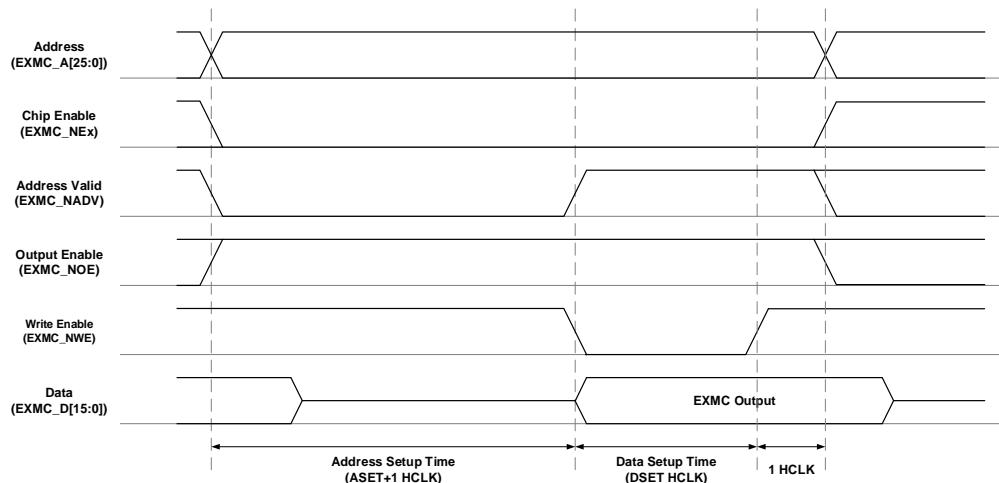
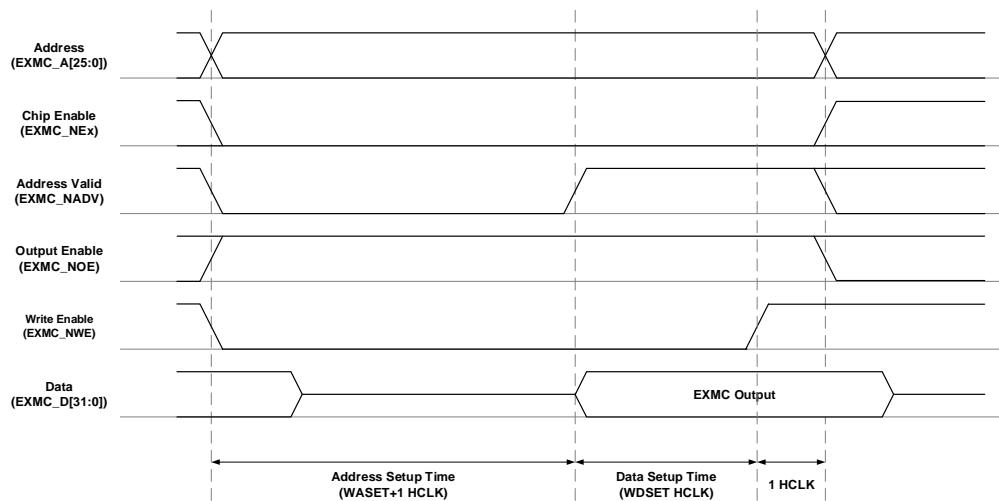
Bit Position	Bit Name	Reference Setting Value
<b>EXMC_SNCTLx</b>		
31-20	Reserved	0x000
19	SYNCWR	0x0
18-16	CPS	0x0
15	ASYNCWTEN	Depends on memory
14	EXMODEN	0x1
13	NRWTEN	0x0
12	WEN	Depends on user
11	NRWTCFG	No effect
10	WRAPEN	0x0
9	NRWTPOL	Meaningful only when the bit 15 is set to 1
8	SBRSTEN	0x0
7	Reserved	0x1
6	NREN	No effect
5-4	NRW	Depends on memory
3-2	NRTP	Depends on memory, except 2(Nor Flash)
1	NRMUX	0x0
0	NRBKEN	0x1
<b>EXMC_SNTCFGx(Read)</b>		
31-30	Reserved	0x0
29-28	ASYNCFMOD	0x0
27-24	DLAT	No effect
23-20	CKDIV	No effect

Bit Position	Bit Name	Reference Setting Value
19-16	BUSLAT	Time between EXMC_NE[x] rising edge to EXMC_NE[x] falling edge
15-8	DSET	Depends on memory and user (DSET+3 HCLK for read)
7-4	AHLD	No effect
3-0	ASET	Depends on memory and user
<b>EXMC_SNWTCFGx(Write)</b>		
31-30	Reserved	0x0
29-28	WASYNCMOD	0x0
27-20	Reserved	0x00
19-16	WBUSLAT	Time between EXMC_NE[x] rising edge to EXMC_NE[x] falling edge
15-8	WDSET	Depends on memory and user (WDSET+1 HCLK for write)
7-4	WAHLD	0x0
3-0	WASET	Depends on memory and user

Mode 2/B - NOR Flash

**Figure 21-10. Mode 2/B read access**



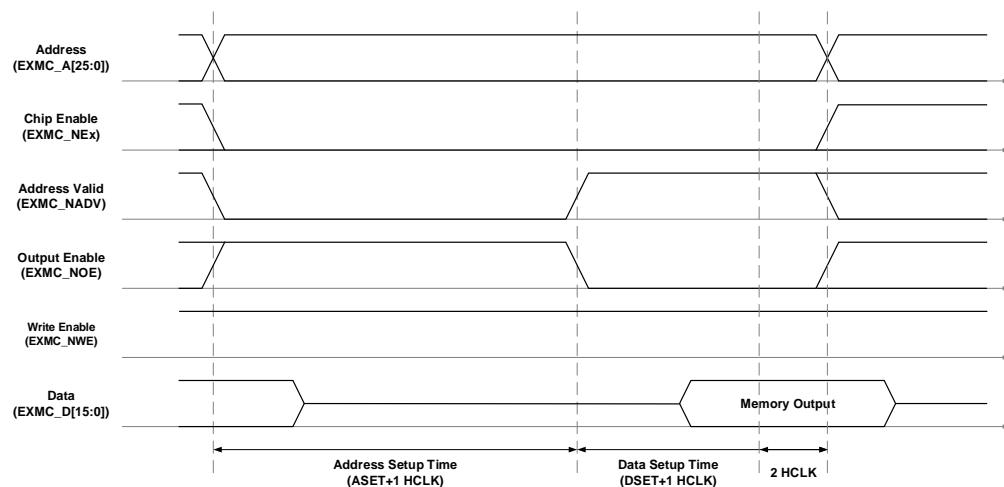
**Figure 21-11. Mode 2 write access**

**Figure 21-12. Mode B write access**

**Table 21-8. Mode 2/B related registers configuration**

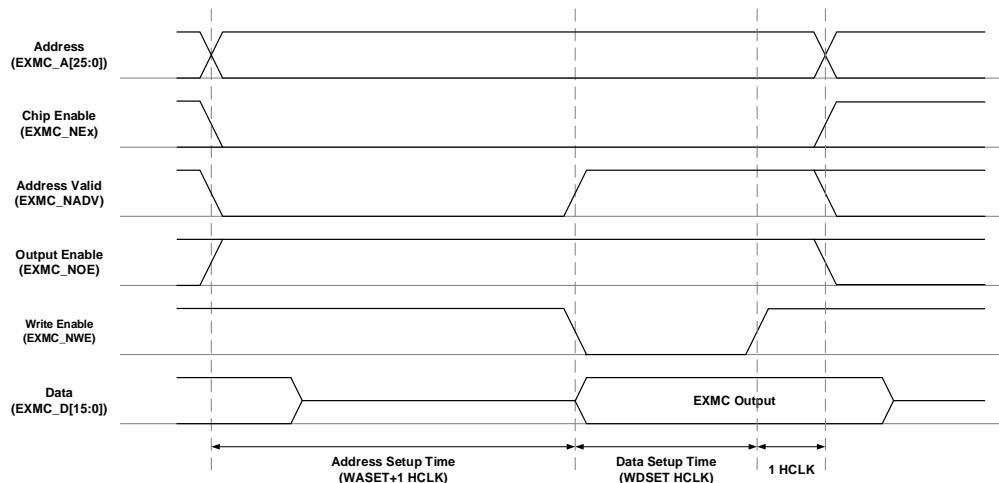
Bit Position	Bit Name	Reference Setting Value
<b>EXMC_SNCTLx(Mode 2, Mode B)</b>		
31-20	Reserved	0x000
19	SYNCWR	0x0
18-16	CPS	0x0
15	ASYNCWTEN	Depends on memory
14	EXMODEN	Mode 2:0x0, Mode B:0x1
13	NRWTEN	0x0
12	WEN	Depends on user
11	NRWTCFG	No effect
10	WRAPEN	0x0
9	NRWTPOL	Meaningful only when the bit 15 is set to 1
8	SBRSTEN	0x0
7	Reserved	0x1

Bit Position	Bit Name	Reference Setting Value
6	NREN	0x1
5-4	NRW	Depends on memory
3-2	NRTP	0x2, NOR Flash
1	NRMUX	0x0
0	NRBKEN	0x1
<b>EXMC_SNTCFGx(Read and write in mode 2,read in mode B)</b>		
31-30	Reserved	0x0000
29-28	ASYNCMOD	Mode B:0x1
27-24	DLAT	No effect
23-20	CKDIV	No effect
19-16	BUSLAT	Time between EXMC_NE[x] rising edge to EXMC_NE[x] falling edge
15-8	DSET	Depends on memory and user (DSET+3 HCLK for read)
7-4	AHLD	0x0
3-0	ASET	Depends on memory and user
<b>EXMC_SNWTCFGx(Write in mode B)</b>		
31-30	Reserved	0x0000
29-28	WASYNCMOD	Mode B:0x1
27-20	Reserved	0x00
19-16	WBUSLAT	Time between EXMC_NE[x] rising edge to EXMC_NE[x] falling edge
15-8	WDSET	Depends on memory and user (WDSET+1 HCLK for write)
7-4	WAHLD	0x0
3-0	WASET	Depends on memory and user

Mode C - NOR Flash OE toggling

**Figure 21-13. Mode C read access**



**Figure 21-14. Mode C write access**


The different between mode C and mode 1 write timing is that read/write timing is specified by the same set of timing configuration, while mode C write timing configuration is independent of its read configuration.

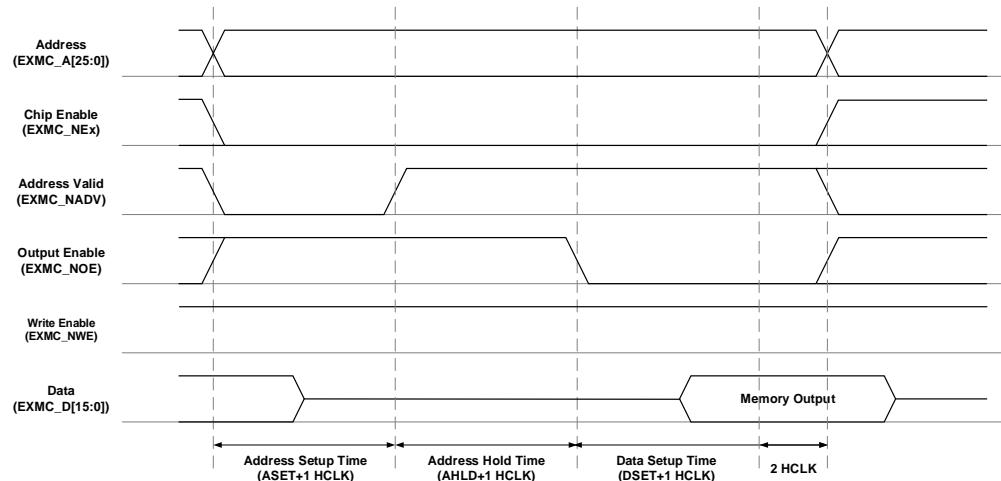
**Table 21-9. Mode C related registers configuration**

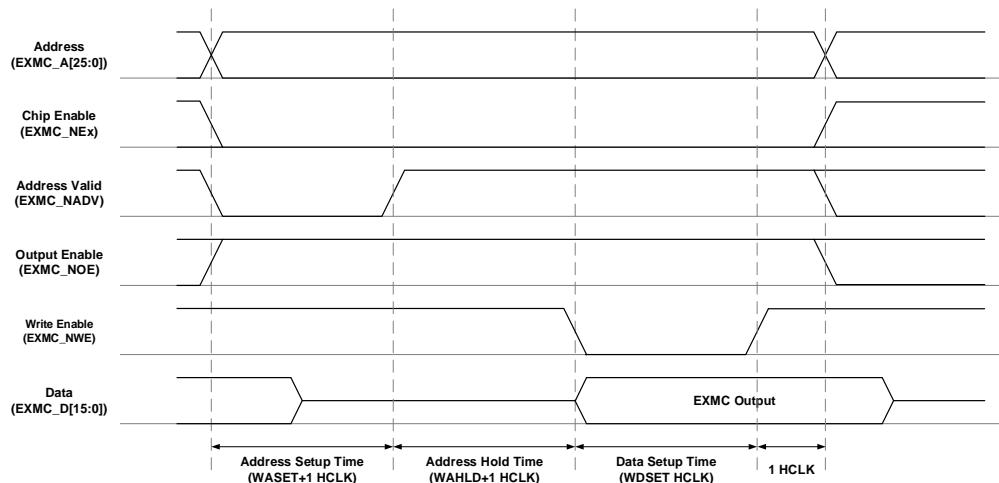
Bit Position	Bit Name	Reference Setting Value
<b>EXMC_SNCTLx</b>		
31-20	Reserved	0x000
19	SYNCWR	0x0
18-16	CPS	0x0
15	ASYNCWTEN	Depends on memory
14	EXMODEN	0x1
13	NRWTEN	0x0
12	WEN	Depends on user
11	NRWTCFG	No effect
10	WRAPEN	0x0
9	NRWTPOL	Meaningful only when the bit 15 is set to 1
8	SBRSTEN	0x0
7	Reserved	0x1
6	NREN	0x1
5-4	NRW	Depends on memory
3-2	NRTP	0x2, NOR Flash
1	NRMUX	0x0
0	NRBKEN	0x1
<b>EXMC_SNTCFGx</b>		
31-30	Reserved	0x0000
29-28	ASYNCFMOD	Mode C:0x2
27-24	DLAT	No effect
23-20	CKDIV	No effect

Bit Position	Bit Name	Reference Setting Value
19-16	BUSLAT	Time between EXMC_NE[x] rising edge to EXMC_NE[x] falling edge
15-8	DSET	Depends on memory and user (DSET+3 HCLK for read)
7-4	AHLD	0x0
3-0	ASET	Depends on memory and user
<b>EXMC_SNWT CFGx</b>		
31-30	Reserved	0x0
29-28	WASYNCMOD	Mode C:0x2
27-20	Reserved	0x00
19-16	WBUSLAT	Time between EXMC_NE[x] rising edge to EXMC_NE[x] falling edge
15-8	WDSET	Depends on memory and user (WDSET+1 HCLK for write)
7-4	WAHLD	0x0
3-0	WASET	Depends on memory and user

Mode D - Asynchronous access with extended address

**Figure 21-15. Mode D read access**



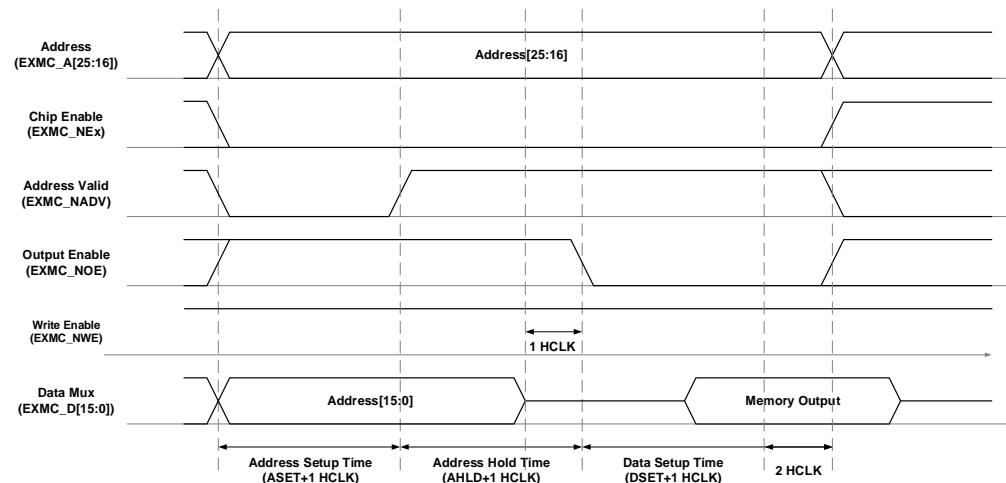
**Figure 21-16. Mode D write access**

**Table 21-10. Mode D related registers configuration**

Bit Position	Bit Name	Reference Setting Value
<b>EXMC_SNCTLx</b>		
31-20	Reserved	0x000
19	SYNCWR	0x0
18-16	CPS	0x0
15	ASYNCWTEN	Depends on memory
14	EXMODEN	0x1
13	NRWTEN	0x0
12	WEN	Depends on user
11	NRWTCFG	No effect
10	WRAPEN	0x0
9	NRWTPOL	Meaningful only when the bit 15 is set to 1
8	SBRSTEN	0x0
7	Reserved	0x1
6	NREN	Depends on memory
5-4	NRW	Depends on memory
3-2	NRTP	Depends on memory
1	NRMUX	0x0
0	NRBKEN	0x1
<b>EXMC_SNTCFGx</b>		
31-30	Reserved	0x0
29-28	ASYNCMOD	Mode D:0x3
27-24	DLAT	Don't care
23-20	CKDIV	No effect
19-16	BUSLAT	Time between EXMC_NE[x] rising edge to EXMC_NE[x] falling edge
15-8	DSET	Depends on memory and user (DSET+3 HCLK for read)

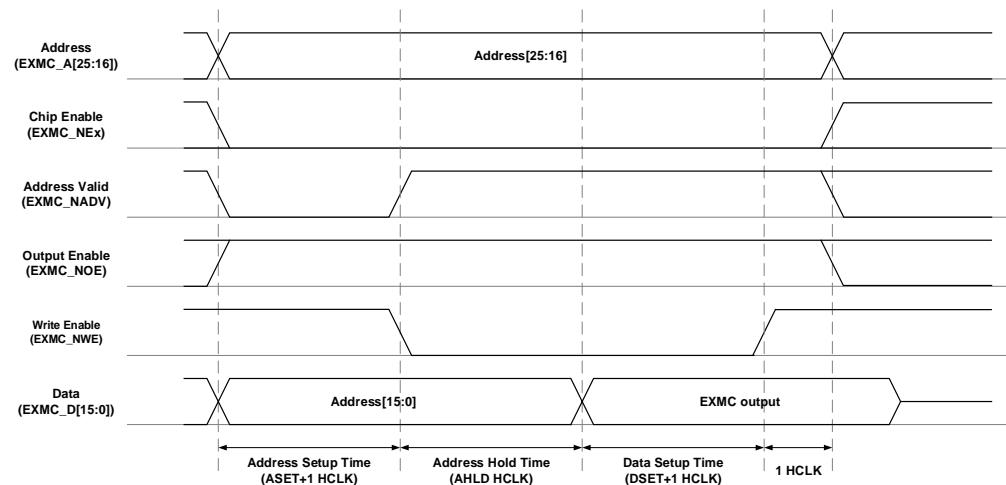
Bit Position	Bit Name	Reference Setting Value
7-4	AHLD	Depends on memory and user
3-0	ASET	Depends on memory and user
<b>EXMC_SNWT CFGx</b>		
31-30	Reserved	0x0
29-28	WASYNCMOD	Mode D:0x3
27-20	Reserved	0x00
19-16	WBUSLAT	Time between EXMC_NE[x] rising edge to EXMC_NE[x] falling edge
15-8	WDSET	Depends on memory and user (WDSET+1HCLK for write)
7-4	WAHLD	Depends on memory and user
3-0	WASET	Depends on memory and user

Mode AM - NOR Flash address / data bus multiplexing

**Figure 21-17. Multiplex mode read access**



**Figure 21-18. Multiplex mode write access**



**Table 21-11. Multiplex mode related registers configuration**

Bit Position	Bit Name	Reference Setting Value
<b>EXMC_SNCTLx</b>		
31-20	Reserved	0x000
19	SYNCWR	0x0
18-16	CPS	0x0
15	ASYNCWTEN	Depends on memory
14	EXMODEN	0x0
13	NRWTEN	0x0
12	WEN	Depends on memory
11	NRWTCFG	No effect
10	WRAPEN	0x0
9	NRWTPOL	Meaningful only when the bit 15 is set to 1
8	SBRSTEN	0x0
7	Reserved	0x1
6	NREN	0x1
5-4	NRW	Depends on memory
3-2	NRTP	0x2:NOR Flash
1	NRMUX	0x1
0	NRBKEN	0x1
<b>EXMC_SNTCFGx</b>		
31-30	Reserved	0x0
29-28	ASYNCMOD	0x0
27-24	DLAT	No effect
23-20	CKDIV	No effect
19-16	BUSLAT	Minimum time between EXMC_NE[x] rising edge to EXMC_NE[x] falling edge
15-8	DSET	Depends on memory and user (DSET+2 HCLK for write, DSET+3 HCLK for read)
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:

$$T_{DATA\_SETUP} \geq \max T_{WAIT\_ASSERTION} + 4HCLK \quad (21-1)$$

If memory wait signal is aligned to EXMC\_NE:

If

$$\max T_{\text{WAIT\_ASSERTION}} \geq T_{\text{ADDRES\_PHASE}} + T_{\text{HOLD\_PHASE}} \quad (21-2)$$

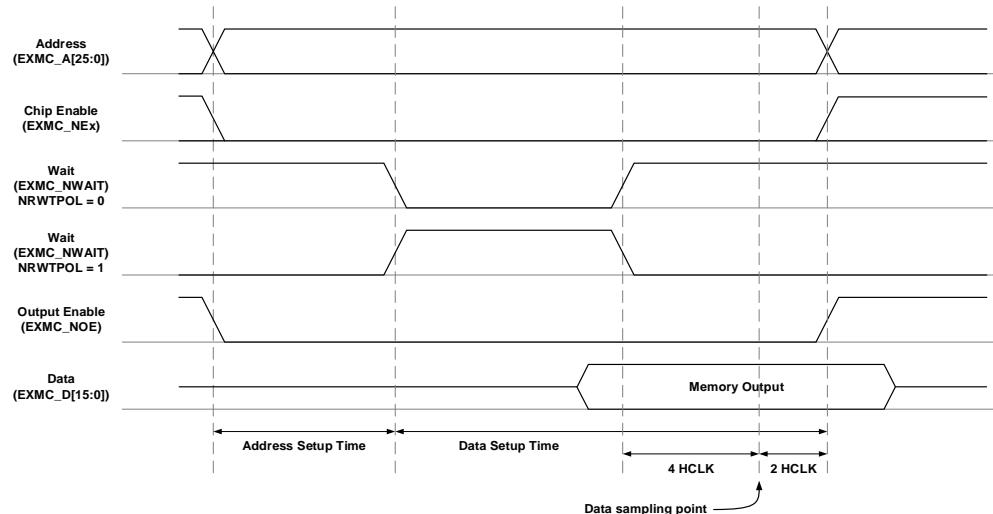
be

$$T_{\text{DATA\_SETUP}} \geq (\max T_{\text{WAIT\_ASSERTION}} - T_{\text{ADDRES\_PHASE}} - T_{\text{HOLD\_PHASE}}) + 4\text{HCLK} \quad (21-3)$$

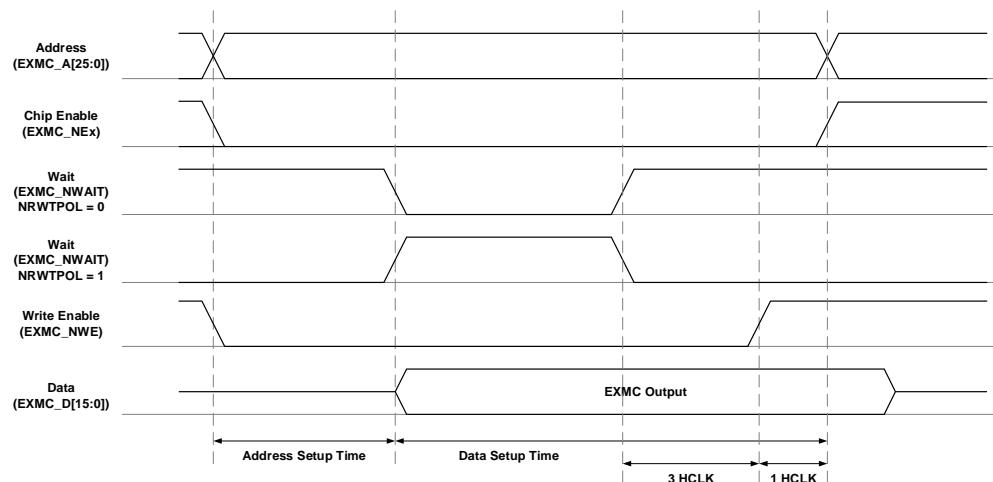
Otherwise

$$T_{\text{DATA\_SETUP}} \geq 4\text{HCLK} \quad (21-4)$$

**Figure 21-19. Read access timing diagram under async-wait signal assertion**



**Figure 21-20. Write access timing diagram under async-wait signal assertion**



### Synchronous access timing diagram

The relations between memory clock (EXMC\_CLK) and system clock (HCLK) clock are as follows:

$$\text{EXMC\_CLK} = \frac{\text{HCLK}}{\text{CKDIV}+1} \quad (21-5)$$

CKDIV is the synchronous clock divider ratio, it is configured through the CKDIV control field

in the EXMC\_SNTCFGx register.

#### 1. Data latency and NOR Flash latency

Data latency is the number of EXMC\_CLK cycles to wait before sampling the data. The relationship between data latency and NOR Flash specification's latency parameter is as follows:

For NOR Flash's specification excluding the EXMC\_NADV cycle, their relationship should be:

$$\text{NOR Flash latency} = \text{DLAT} + 2 \quad (21-6)$$

For NOR Flash's specification including the EXMC\_NADV cycle, their relationship should be:

$$\text{NOR Flash latency} = \text{DLAT} + 3 \quad (21-7)$$

#### 2. Data wait

Users should guarantee that EXMC\_NWAIT signal matches that of the external device. This signal is configured through the EXMC\_SNCTLx registers, it is enabled by the NRWTEN bit, and the active timing could be one data cycle before the wait state or active during the active state by the configuration NRWTCFG bit, while the wait signal's polarity is set by the NRWTPOL bit.

In NOR Flash synchronous burst access mode, when NRWTEN bit in EXMC\_SNCTLx register is set, EXMC\_NWAIT signal will be detected after a period of data latency. If EXMC\_NWAIT signal detected is valid, wait cycles will be inserted until EXMC\_NWAIT becomes invalid.

- The valid polarity of EXMC\_NWAIT:

NRWTPOL= 1: valid level of EXMC\_NWAIT signal is high.

NRWTPOL= 0: valid level of EXMC\_NWAIT signal is low.

- In synchronous burst mode, EXMC\_NWAIT signal has two kinds of configurations:

NRWTCFG = 1: When EXMC\_NWAIT signal is active, current cycle data is not valid.

NRWTCFG = 0: When EXMC\_NWAIT signal is active, the next cycle data is not valid. It is the default state after reset.

During wait-state inserted via the EXMC\_NWAIT signal, the controller continues to send clock pulses to the memory, keep the chip select and output signals available, and ignore the invalid data signal.

#### 3. Automatic burst split at CRAM page boundary

Crossing page boundary burst access is prohibited in CRAM 1.5, an automatic burst split functionality is implemented by the EXMC. To guarantee correct burst split operation, users should specify CRAM page size by configuring the CPS bit in EXMC\_SNCTLx register to inform the EXMC when this functionality should be performed.

#### 4. Mode SM - Single burst transmission

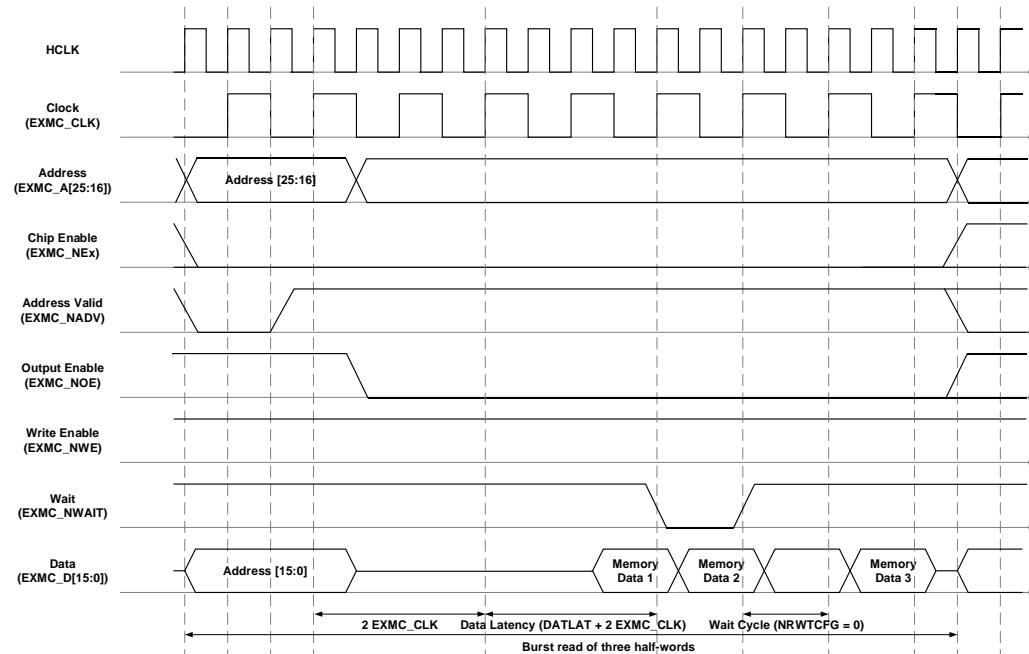
For synchronous burst transmission, if the needed data of AHB is 16-bit, EXMC will perform

a burst transmission whose length is 1. If the needed data of AHB is 32-bit, EXMC will make the transmission divided into two 16-bit transmissions, that is, EXMC performs a burst transmission whose length is 2.

For other configurations please refers to [\*\*Table 21-3. EXMC bank 0 supports all transactions.\*\*](#)

Synchronous mux burst read timing - NOR, PSRAM (CRAM)

**Figure 21-21. Read timing of synchronous multiplexed burst mode**



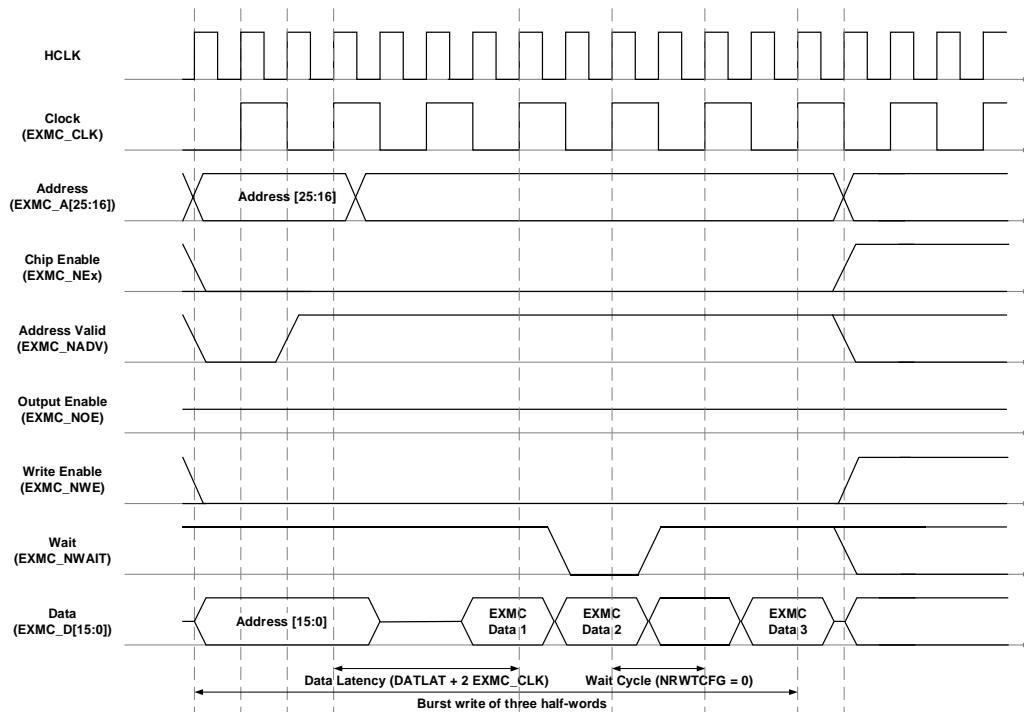
**Table 21-12. Timing configurations of synchronous multiplexed read mode**

Bit Position	Bit Name	Reference Setting Value
<b>EXMC_SNCTLx</b>		
Bit Position	Bit Name	Reference Setting Value
31-20	Reserved	0x000
19	SYNCWR	No effect
18-16	CPS	0x0
15	ASYNCTEN	0x0
14	EXMODEN	0x0
13	NRWTEN	Depends on memory
12	WEN	No effect
11	NRWTCFG	Depends on memory
10	WRAPEN	0x0
9	NRWTPOL	Depends on memory
8	SBRSTEN	0x1, burst read enable
7	Reserved	0x1
6	NREN	Depends on memory
5-4	NRW	0x1

Bit Position	Bit Name	Reference Setting Value
<b>EXMC_SNCTLx</b>		
3-2	N RTP	Depends on memory, 0x1/0x2
1	N R MUX	0x1, Depends on memory and users
0	N R BKEN	0x1
<b>EXMC_SNTCFGx(Read)</b>		
31-30	Reserved	0x0
29-28	A SYNC MOD	0x0
27-24	D LAT	Data latency
23-20	C K DIV	The figure above: 0x1, EXMC_CLK=2HCLK
19-16	B US LAT	Time between EXMC_NE[x] rising edge to EXMC_NE[x] falling edge
15-8	D SET	No effect
7-4	A HLD	No effect
3-0	A SET	No effect

Mode SM –Synchronous mux burst write timing – PSRAM (CRAM)

**Figure 21-22. Write timing of synchronous multiplexed burst mode**



**Table 21-13. Timing configurations of synchronous multiplexed write mode**

Bit Position	Bit Name	Reference Setting Value
<b>EXMC_SNCTLx</b>		
31-20	Reserved	0x000
19	S YNC WR	0x1, synchronous write enable
18-16	C P S	0x0

Bit Position	Bit Name	Reference Setting Value
<b>EXMC_SNCTLx</b>		
15	AYSNCWAIT	0x0
14	EXMODEN	0x0
13	NRWTEN	Depends on memory
12	WREN	0x1
11	NRWTCFG	0x0(Here must be zero)
10	WRAPEN	0x0
9	NTWTPOL	Depends on memory
8	SBRSTEN	No effect
7	Reserved	0x1
6	NREN	Depends on memory
5-4	NRW	0x1
3-2	NRTP	0x1
1	NRMUX	0x1, Depends on users
0	NRBKEN	0x1
<b>EXMC_SNTCFGx(Write)</b>		
31-30	Reserved	0x0
29-28	ASYNCMOD	0x0
27-24	DLAT	Data latency
23-20	CKDIV	The figure above: 0x1, EXMC_CLK=2HCLK
19-16	BUSLAT	Time between EXMC_NE[x] rising edge to EXMC_NE[x] falling edge
15-8	DSET	No effect
7-4	AHLD	No effect
3-0	ASET	No effect

### 21.3.5. NAND Flash or PC Card controller

EXMC has partitioned Bank1 and Bank2 as NAND Flash access field, bank3 as PC Card access field. Each bank has its own set of control register for access timing configuration. 8- and 16-bit NAND Flash and 16-bit PC Card are supported. An ECC hardware is provided for the NAND Flash controller to ensure the robustness of data transfer and storage.

#### NAND Flash or PC Card interface function

**Table 21-14. 8-bit or 16-bit NAND interface signal**

EXMC Pin	Direction	Functional description
EXMC_A[17]	Output	NAND Flash address latch (ALE)
EXMC_A[16]	Output	NAND Flash command latch (CLE)
EXMC_D[7:0]/ EXMC_D[15:0]	Input /Output	8-bit multiplexed, bidirectional address/data bus
		16-bit multiplexed, bidirectional address/data bus
EXMC_NCE[x]	Output	Chip select, x = 1, 2

<b>EXMC Pin</b>	<b>Direction</b>	<b>Functional description</b>
EXMC_NOE(NR E)	Output	Output enable
EXMC_NWE	Output	Write enable
EXMC_NWAIT/ EXMC_INT[x]	Input	NAND Flash ready/busy input signal to the EXMC, x=1, 2

**Table 21-15. 16-bit PC Card interface signal**

<b>EXMC Pin</b>	<b>Direction</b>	<b>Functional description</b>
EXMC_A[10:0]	Output	Address bus of PC Card
EXMC_NIOS16	Input	Only for 16-bit I/O space data transmission width (Must be shorted to GND)
EXMC_NIORD	Output	I/O space read enable
EXMC_NIOWR	Output	I/O space write enable
EXMC_NREG	Output	Register signal indicating if access is in Common space or Attribute space
EXMC_D[15:0]	Input /Output	Bidirectional data bus
EXMC_NCE3_x	Output	Chip select(x=0,1)
EXMC_NOE	Output	Output enable
EXMC_NWE	Output	Write enable
EXMC_NWAIT	Input	PC Card wait input signal to the EXMC
EXMC_INTR	Input	PC Card interrupt input signal
EXMC_CD	Input	PC Card presence detection. Active high.

### Supported memory access mode

**Table 21-16. Bank1/2/3 of EXMC support the memory and access mode**

<b>Memory</b>	<b>Mode</b>	<b>R/W</b>	<b>AHB transaction size</b>	<b>Comments</b>
8-bit NAND	Async	R	8	
	Async	W	8	
	Async	R	16	Automatically split into 2 EXMC accesses
	Async	W	16	
	Async	R	32	Automatically split into 4 EXMC accesses
	Async	W	32	
16-bit NAND/PC Card	Async	R	8	
	Async	W	8	
	Async	R	16	Not support this operation
	Async	W	16	
	Async	R	32	Automatically split into 2 EXMC accesses
	Async	W	32	

### NAND Flash or PC Card controller timing

EXMC can generate the appropriate signal timing for NAND Flash, PC Cards and other

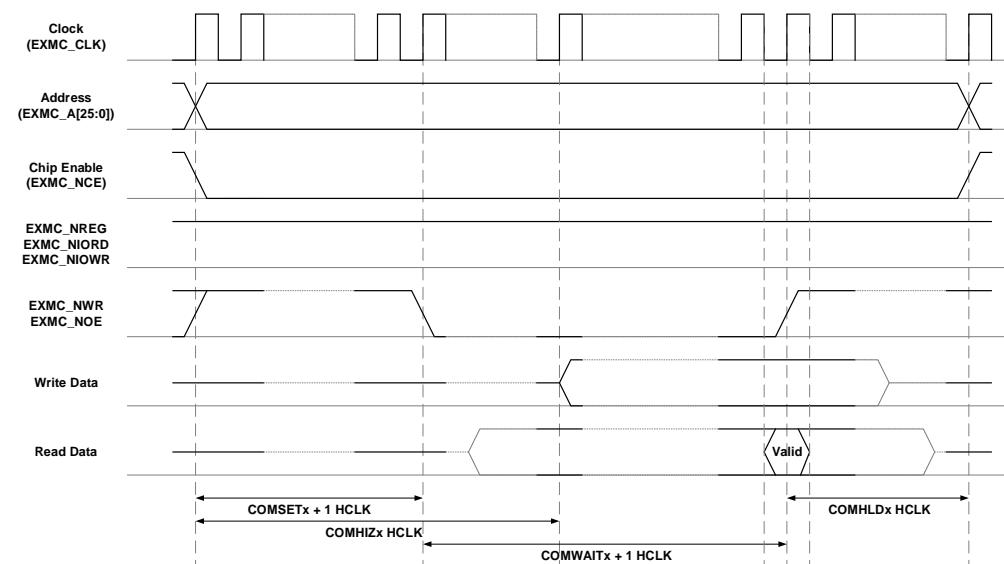
devices. Each bank has a corresponding register to manage and control the external memory, such as EXMC\_NPCTLx, EXMC\_NPINTENx, EXMC\_NPCTCFGx, EXMC\_NPATCFGx, EXMC\_PIOTCFG3 and EXMC\_NECCx. Among these registers, EXMC\_NPCTCFGx, EXMC\_NPATCFGx, EXMC\_PIOTCFG3 registers contain four timing parameters individually which are configured according to user specification and features of the external memory.

**Table 21-17. NAND Flash or PC Card programmable parameters**

Programmable parameter	W/R	Unit	Functional description	NAND Flash/ PC Card	
				Min	Max
High impedance time of the memory data bus (HIZ)	W/R	HCLK	Time to keep the data bus high impedance after starting write operation	0	255
Memory hold time (HLD)	W/R	HCLK	The number of HCLK clock cycles to keep address valid after sending the command. In write mode, it is also data hold time.	1	254
Memory wait time (WAIT)	W/R	HCLK	Minimum duration of sending command	2	256
Memory setup time (SET)	W/R	HCLK	The number of HCLK clock cycles to build address before sending command	1	255

The figure below shows the programmable parameters which are defined in the common memory space operations. The programmable parameters of Attribute memory space or I/O memory space (only for PC Card) are defined as well.

**Figure 21-23. Access timing of common memory space of PC Card Controller**



### NAND Flash operation

When EXMC sends command or address to NAND Flash, it needs to use the command latch signal (A [16]) or address latch signal (EXMC\_A [17]), namely, the CPU needs to perform write operation in particular address.

Example: NAND Flash read operation steps:

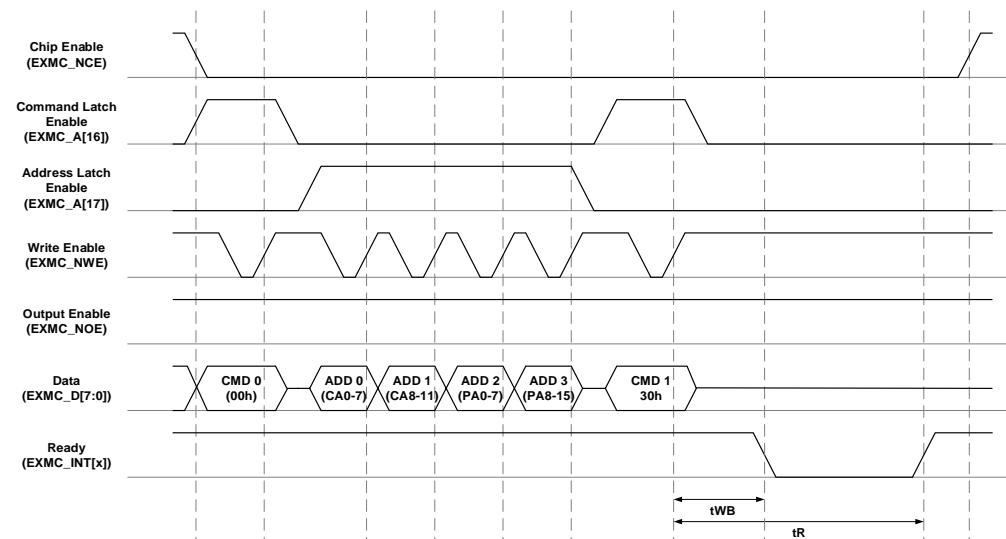
1. Configure EXMC\_NPCTLx and EXMC\_NPCTCFGx register. When pre-waiting is needed, EXMC\_NPATCFGx has to be configured.
2. Send the command of NAND Flash read operation to the common space. Namely, during the valid period of EXMC\_NCE and EXMC\_NWE, when EXMC\_CLE (EXMC\_A [16]) becomes valid (high level), data on the I/O pins is regarded as a command by NAND Flash.
3. Send the start address of read operation to the common space. During the valid period of EXMC\_NCE and EXMC\_NWE, when EXMC\_ALE (EXMC\_A [17]) becomes valid (high level), the data on the I/O pins is regarded as an address by NAND Flash.
4. Waiting for NAND ready signal. In this period, NAND controller will maintain EXMC\_NCE valid.
5. Read data byte by byte from the data area of the common space.
6. If new commands or address haven't been written, data of the next page can be read out automatically. You can also read the data of the next page by going to step 3 and then writing a new address or writing a new command and address in step 2.

## NAND Flash pre-wait functionality

Some NAND Flash requires that the controller should wait for NAND Flash to be busy after the first command byte following the address bytes is send, and some EXMC\_NCE-sensitive NAND Flash also requires that the EXMC\_NCE must remain valid before it is ready.

Taking TOSHIBA128 M x 8 bit NAND Flash as an example:

**Figure 21-24. Access to none "NCE don't care" NAND Flash**



1. Write CMD0 into NAND Flash bank common space command area.

- 
2. Write ADD0 into NAND Flash bank common space address area.
  3. Write ADD1 into NAND Flash bank common space address area.
  4. Write ADD2 into NAND Flash bank common space address area.
  5. Write ADD3 into NAND Flash bank common space address area.
  6. Write CMD1 into NAND Flash bank attribute space command area.

In step 6, EXMC uses the operation timing defined in EXMC\_NPATCFGx register. After a period of ATTHLD, NAND Flash waits for EXMC\_INTx signal to be busy, and the time period of ATTHLD should be greater than tWB (tWB is defined as the time from EXMC\_NWE high to EXMC\_INTx low). For NCE-sensitive NAND Flash, after the first command byte following address bytes has been entered, EXMC\_NCE must remain low until EXMC\_INTx goes from low to high. The ATTHLD value of attribute space can be set in EXMC\_NPATCFGx register to meet the timing requirements of tWB. CPU can use the attribute space timing when writing the first command byte following address bytes to the NAND Flash device. In other times, the MCU must use the common space timing.

### NAND Flash ECC calculation module

An ECC calculation hardware is implemented in bank1 and bank2 respectively. Users can choose page size according to the ECCSZ control field in the EXMC\_NPCTLx register. ECC offers one bit error correction and two bits errors detection.

When NAND memory block is enabled, ECC module will detect EXMC\_D[15:0], EXMC\_NCE and EXMC\_NWE signals. When a data size of ECCSZ has been read or written, software must read the calculated ECC in the EXMC\_NECCx register. When a recalculation of ECC is needed, software must clear the EXMC\_NECCx register value by resetting ECCEN bit of EXMC\_NPCTLx register to zero, and then restart ECC calculation by setting the ECCEN bit of EXMC\_NPCTLx to 1.

### PC/CF Card access

EXMC Bank3 is used exclusively for PC/CF Card, both memory and IO mode access are supported. This bank is divided further into three sub spaces, memory, attribute and IO space.

EXMC\_NCE3\_0 and EXMC\_NCE3\_1 are the byte select signals, when only EXMC\_NCE3\_0 is active (Low), the lower byte or upper byte is selected depending on the EXMC\_A[0], while only EXMC\_NCE3\_1 is active (Low), the upper byte is selected which is not supported, when both of these signals are active, 16-bit operation is performed. When NDTP is reset to select PC/CF Card as external memory device, NDW must be set to 01 in EXMC\_NPCTLx register to guarantee correct EXMC operation.

EXMC PC/CF card access behavior for different spaces:

1. Common space: EXMC\_NCE3\_x ( $x = 0, 1$ ) is the chip enable signal, it indicates whether 8- or 16-bit access operation is being performed. EXMC\_NWE and EXMC\_NOE dictates whether the on-going operation is a write or read operation, and EXMC\_NREG is high during common space access.

2. Attribute space: EXMC\_NCE3\_x (x = 0, 1) is the chip enable signal, it indicates whether 8- or 16-bit access operation is being performed. EXMC\_NWE and EXMC\_NOE dictates whether the on-going operation is a write or read operation, and EXMC\_NREG is low during attribute space access.
3. IO space: EXMC\_NCE3\_x (x = 0, 1) is the chip enable signal, it indicates whether 8- or 16-bit access operation is being performed. EXMC\_NIOWR and EXMC\_NIORD dictate whether the on-going operation is a write or read operation, and EXMC\_NREG is low during IO space access.

AHB access on 16-bit PC/CF card:

1. Common space: It is usually where data are stored, it could be accessible either in byte or in half-word mode, and odd address access is not supported in byte mode. When AHB word access is selected, EXMC automatically splits it into 2 consecutive half-word access. EXMC\_NREG is high when common memory is targeted. EXMC\_NOE and EXMC\_NWE are the read and write enable signal for this type of access.
2. Attribute space: It is usually where configuration information are stored, for byte AHB access, only even address is possible. Half-word access converts into a single byte access automatically, and word access is converted into two consecutive byte access where only the even bytes are operational. In both half-word and word access, only EXMC\_NCE3\_0 will be active. EXMC\_NREG is low when attribute memory is targeted. EXMC\_NOE and EXMC\_NWE are the read and write enable signal for this type of access.
3. IO space: Both byte and half-word AHB access are supported, in IO space memory access, EXMC\_NIORD and EXMC\_NIOWR act as the read and write enable signal respectively.

## 21.4. Registers definition

### 21.4.1. NOR/PSRAM controller registers

#### SRAM/NOR Flash control registers (EXMC\_SNCTLx) (x=0, 1, 2, 3)

Address offset: 0x00 + 8 \* x, (x = 0, 1, 2, and 3)

Reset value: 0x0000 30DB for region0, and 0x0000 30D2 for region1, region2, and region3.

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												SYNC WR	CPS[2:0]		
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ASYNC WAIT	EXMO DEN	NRWT EN	WREN	NRWT CFG	WRAPEN	NRWT POL	SBR STEN	Reserved	NR EN	NRW[1:0]	NRTP[1:0]	NR MUX	NRB EN		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31:20	Reserved	Must be kept at reset value.
19	SYNCWR	Synchronous write 0: Asynchronous write 1: Synchronous write
18:16	CPS[2:0]	CRAM page size 000: Automatic burst split on page boundary crossing 001: 128 bytes 010: 256 bytes 011: 512 bytes 100: 1024 bytes Others: Reserved
15	ASYNCWAIT	Asynchronous wait 0: Disable the asynchronous wait function 1: Enable the asynchronous wait function
14	EXMODEN	Extended mode enable 0: Disable extended mode 1: Enable extended mode
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 NWAI 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	NRWTCFG	NWAIT signal configuration, only work in synchronous mode  0: NWAIT signal is active one data cycle before wait state 1: NWAIT signal is active during wait state
10	WRAPEN	Wrapped burst mode enable  0: Disable wrap burst mode support 1: Enable wrap burst mode support
9	NRWTPOL	NWAIT signal polarity  0: Low level is active of NWAIT 1: High level is active of NWAIT
8	SBRSTEN	Synchronous burst enable  0: Disable burst access mode 1: Enable burst access mode
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(default after reset for region1-region3) 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, 1, 2,**

**3)**

Address offset: 0x04 + 8 \* x, (x = 0, 1, 2, and 3)

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		ASYNCFMOD[1:0]			DLAT[3:0]			CKDIV[3:0]			BUSLAT[3:0]				
rw					rw				rw						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
31:30	Reserved	Must be kept at reset value.
29:28	ASYNCFMOD[1:0]	Asynchronous access mode The bits are valid only when the EXMODEN bit in the EXMC_SNCTLx register is 1. 00: Mode A access 01: Mode B access 10: Mode C access 11: Mode D access
27:24	DLAT[3:0]	Data latency for NOR Flash. Only valid in synchronous access 0x0: Data latency of first burst access is 2 EXMC_CLK 0x1: Data latency of first burst access is 3 EXMC_CLK ..... 0xF: Data latency of first burst access is 17 EXMC_CLK
23:20	CKDIV[3:0]	Synchronous clock divide ratio. This field is only effect in synchronous mode. 0x0: Reserved 0x1: EXMC_CLK period = 2 * HCLK period ..... 0xF: EXMC_CLK period = 16 * HCLK period
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, which only used in mode D and multiplexed mode.
		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

### **SRAM/NOR Flash write timing configuration registers (EXMC\_SNWTCFGx) (x=0, 1, 2, 3)**

Address offset: 0x104 + 8 \* x, (X = 0, 1, 2, and 3)

Reset value: 0xFFFF FFFF

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

This register is meaningful only when the EXMODEN bit in EXMC\_SNCTLx is set to 1.

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

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:30	Reserved	Must be kept at reset value.
29:28	Wasyncmod[1:0]	Asynchronous access mode The bits are valid only when the EXMODEN bit in the EXMC_SNCTLx register is 1. 00: Mode A access 01: Mode B access 10: Mode C access

## 11: Mode D access

27:20	Reserved	Must be kept at reset value.
19:16	WBUSLAT[3:0]	<p>Bus latency</p> <p>Bus latency added at the end of each write transaction to match with the minimum time between consecutive transactions.</p> <p>0x0: Bus latency = 1 * HCLK period</p> <p>0x1: Bus latency = 2 * HCLK period</p> <p>.....</p> <p>0xF: Bus latency = 16 * HCLK period</p>
15:8	WDSET[7:0]	<p>Data setup time</p> <p>This field is meaningful only in asynchronous access.</p> <p>0x00: Reserved</p> <p>0x01: Data setup time = 2 * HCLK period</p> <p>.....</p> <p>0xFF: Data setup time = 256 * HCLK period</p>
7:4	WAHLD[3:0]	<p>Address hold time</p> <p>This field is used to set the time of address hold phase, which only used in mode D and multiplexed mode.</p> <p>0x0: Reserved</p> <p>0x1: Address hold time = 2 * HCLK</p> <p>.....</p> <p>0xF: Address hold time = 16 * HCLK</p>
3:0	WASET[3:0]	<p>Address setup time</p> <p>This field is used to set the time of address setup phase.</p> <p><b>Note:</b> Meaningful only in asynchronous access of SRAM,ROM,NOR Flash</p> <p>0x0: Address setup time = 1 * HCLK</p> <p>0x1: Address setup time = 2 * HCLK</p> <p>.....</p> <p>0xF: Address setup time = 16 * HCLK</p>

## 21.4.2. NAND Flash/PC Card controller registers

### NAND Flash/PC Card control registers (EXMC\_NPCTLx) (x=1, 2, 3)

Address offset: 0x40 + 0x20 \* x, (x = 1, 2, and 3)

Reset value: 0x0000 0018

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										ECCSZ[2:0]	ATR[3]				

rw                    rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ATR[2:0]		CTR[3:0]		Reserved	ECCEN	NDW[1:0]	NDTP	NDBKEN	NDWTEN	Reserved					
rw		rw			rw	rw	rw	rw	rw	rw					

Bits	Fields	Description
31:20	Reserved	Must be kept at reset value.
19:17	ECCSZ[2:0]	ECC size 000: 256 bytes 001: 512 bytes 010: 1024 bytes 011: 2048 bytes 100: 4096 bytes 101: 8192 bytes
16:13	ATR[3:0]	ALE to RE delay 0x0: ALE to RE delay = 1 * HCLK ..... 0xF: ALE to RE delay = 16 * HCLK
12:9	CTR[3:0]	CLE to RE delay 0x0: CLE to RE delay = 1 * HCLK 0x1: CLE to RE delay = 2 * HCLK ..... 0xF: CLE to RE delay = 16 * HCLK
8:7	Reserved	Must be kept at reset value.
6	ECCEN	ECC enable 0: Disable ECC, and reset EXMC_NECCx 1: Enable ECC
5:4	NDW[1:0]	NAND bank memory data bus width 00: 8 bits 01: 16 bits Others: Reserved <b>Note:</b> for PC/CF card, 16-bit bus width must be selected.
3	NDTP	NAND bank memory type 0: PC Card, CF card, PCMCIA 1: NAND Flash
2	NDBKEN	NAND bank enable 0: Disable corresponding memory bank 1: Enable corresponding memory bank
1	NDWTEN	Wait function enable 0: Disable wait function

1: Enable wait function

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

### NAND Flash/PC Card interrupt enable registers (EXMC\_NPINTECx) (x=1, 2, 3)

Address offset: 0x44 + 0x20 \* x, (x = 1, 2, and 3)

Reset value: 0x0000 0042 (for bank1 and bank2), 0x0000 0043 (for bank3)

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

In addition to interrupt controlling bits, this register also contains a FIFO empty status bit, design specifically for ECC purpose. When external memory write is performed, the FIFO can hold up to 2 word from AHB access, freeing the bus temporarily for other peripherals. ECC calculation is based on the data passing through the FIFO, for correct ECC, users should read the ECC register only after the FIFO empty status flag is raised.

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				FFEPT	INTFEN	INTHEN	INTREN	INTFS	INTHS	INTRS

Bits	Fields	Description
31:7	Reserved	Must be kept at reset value.
6	FFEPT	FIFO empty flag 0: FIFO is not empty. 1: FIFO is empty.
5	INTFEN	Interrupt falling edge detection enable 0: Disable interrupt falling edge detection 1: Enable interrupt falling edge detection
4	INTHEN	Interrupt high-level detection enable 0: Disable interrupt high-level detection 1: Enable interrupt high-level detection
3	INTREN	Interrupt rising edge detection enable bit 0: Disable interrupt rising edge detection 1: Enable interrupt rising edge detection
2	INTFS	Interrupt falling edge status 0: Not detect interrupt falling edge 1: Detect interrupt falling edge
1	INTHS	Interrupt high-level status

0: Not detect interrupt high-level

1: Detect interrupt high-level

0	INTRS	Interrupt rising edge status
		0: Not detect interrupt rising edge
		1: Detect interrupt rising edge

### **NAND Flash/PC Card common space timing configuration registers (EXMC\_NPCTCFGx) (x=1, 2, 3)**

Address offset: 0x48 + 0x20 \* x, (x = 1, 2, and 3)

Reset value: 0xFCFC FCFC

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

These operations applicable to common memory space for 16-bit PC Card, CF card and NAND Flash.

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

Bits	Fields	Description
31:24	COMHIZ[7:0]	<p>Common memory data bus HiZ time</p> <p>The bits are defined as time of bus keep high impedance state after writing the data.</p> <p>0x00: COMHIZ = 1 * HCLK</p> <p>.....</p> <p>0xFE: COMHIZ = 255 * HCLK</p> <p>0xFF: Reserved</p>
23:16	COMHLD[7:0]	<p>Common memory hold time</p> <p>After sending the address, the bits are defined as the address hold time. In write operation, they are also defined as the data signal hold time.</p> <p>0x00: Reserved</p> <p>0x01: COMHLD = 1 * HCLK</p> <p>.....</p> <p>0xFE: COMHLD = 254 * HCLK</p> <p>0xFF: Reserved</p>
15:8	COMWAIT[7:0]	<p>Common memory wait time</p> <p>Define the minimum time to maintain command</p> <p>0x00: Reserved</p>

0x01: COMWAIT = 2 \* HCLK (+NWAIT active cycles)

.....

0xFE: COMWAIT = 255 \* HCLK (+NWAIT active cycles)

0xFF: Reserved

7:0            COMSET[7:0]            Common memory setup time

Define the time to build address before sending command

0x00: COMSET = 1 \* HCLK

.....

0xFE: COMSET = 255 \* HCLK

0xFF: Reserved

### **NAND Flash/PC Card attribute space timing configuration registers (EXMC\_NPATCFGx) (x=1, 2, 3)**

Address offset: 0x4C + 0x20 \* x, (x = 1, 2, and 3)

Reset value: 0xFCFC FCFC

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

It is used for 8-bit accesses to the attribute memory space of the PC Card or to access the NAND Flash for the last address or command write access if another timing must be applied.

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

Bits	Fields	Description
31:24	ATTHIZ[7:0]	<p>Attribute memory data bus HiZ time</p> <p>The bits are defined as time of bus keep high impedance state after writing the data.</p> <p>0x00: ATTHIZ = 1 * HCLK</p> <p>.....</p> <p>0xFE: ATTHIZ = 255 * HCLK</p> <p>0xFF: Reserved</p>
23:16	ATTHLD[7:0]	<p>Attribute memory hold time</p> <p>After sending the address, the bits are defined as the address hold time. In write operation, they are also defined as the data signal hold time.</p> <p>0x00: Reserved</p> <p>0x01: ATTHLD = 1 * HCLK</p> <p>.....</p> <p>0xFE: ATTHLD = 254 * HCLK</p>

		0xFF: Reserved
15:8	ATTWAIT[7:0]	Attribute memory wait time Define the minimum time to maintain command 0x00: Reserved 0x01: ATTWAIT = 2 * HCLK (+NWAIT active cycles) ..... 0xFE: ATTWAIT = 255 * HCLK (+NWAIT active cycles) 0xFF: ATTWAIT = Reserved
7:0	ATTSET[7:0]	Attribute memory setup time Define the time to build address before sending command 0x00: ATTSET = 1 * HCLK ..... 0xFE: ATTSET = 255 * HCLK 0xFF: Reserved

### PC Card I/O space timing configuration register (EXMC\_PIOTCFG3)

Address offset: 0xB0

Reset value: 0xFCFC FCFC

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



Bits	Fields	Description
31:24	IOHIZ[7:0]	IO space data bus HiZ time The bits are defined as time of bus keep high impedance state after writing the data. 0x00: IOHIZ = 0 *HCLK ..... 0x00: IOHIZ = 255 *HCLK
23:16	IOHLD[7:0]	IO space hold time After sending the address, the bits are defined as the address hold time. In write operation, they are also defined as the data signal hold time. 0x00: Reserved 0x01: IOHLD = 1 * HCLK .....

0xFF: IOHLD = 255 \* HCLK

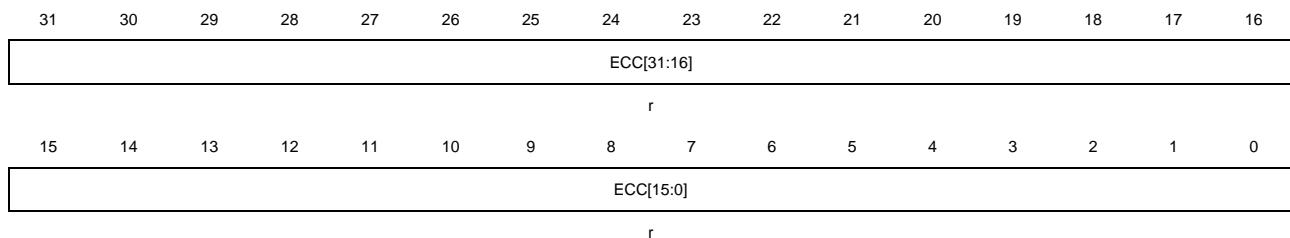
15:8	IOWAIT[7:0]	IO space wait time Define the minimum time to maintain command 0x00: Reserved 0x01: IOWAIT = 2 * HCLK (+NWAIT active cycles) ..... 0xFF: IOWAIT = 256 * HCLK (+NWAIT active cycles)
7:0	IOSET[7:0]	IO space setup time Define the time to build address before sending command 0x00: IOSET = 1 * HCLK ..... 0xFF: IOSET = 256 * HCLK

### NAND Flash ECC registers (EXMC\_NECCx) (x=1, 2)

Address offset: 0x54+0x20 \* x

Reset value: 0x0000 0000

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



Bits	Fields	Description
31:0	ECC[31:0]	ECC result

ECCSZ[2:0]	NAND Flash page size	ECC bits
0b000	256	ECC[21:0]
0b001	512	ECC[23:0]
0b010	1024	ECC[25:0]
0b011	2048	ECC[27:0]
0b100	4096	ECC[29:0]
0b101	8192	ECC[31:0]

## 22. Controller area network (CAN)

### 22.1. Overview

CAN bus (Controller Area Network) is a bus standard designed to allow microcontrollers and devices to communicate with each other without a host computer.

As CAN network interface, basic extended CAN supports the CAN protocols version 2.0A and B. The CAN interface automatically handles the transmission and the reception of CAN frames. The CAN provides 14 scalable/configurable identifier filter banks in GD32F30x XD/HD and 28 scalable/configurable identifier filter banks in GD32F30x CL. The filters are used for selecting the input message as software requirement and otherwise discarding the message. Three transmit mailboxes are provided to the software for transfer messages. The transmission scheduler decides which mailbox will be transmitted firstly. Three complete messages can be stored in every FIFO. The FIFOs are managed completely by hardware. Two receiving FIFOs are used by hardware to store the incoming messages. In addition, the CAN controller provides all hardware functions, which supports the time-triggered communication option, in safety-critical applications.

### 22.2. 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.
- Supports priority of transmission message.
- Supports time stamp at SOF transmission.

#### Reception

- Supports 2 Rx FIFOs and each has 3 messages depth
- 14 scalable/configurable identifier filter banks in GDF30x XD/HD
- 28 scalable/configurable identifier filter banks in GDF30x CL
- FIFO lock

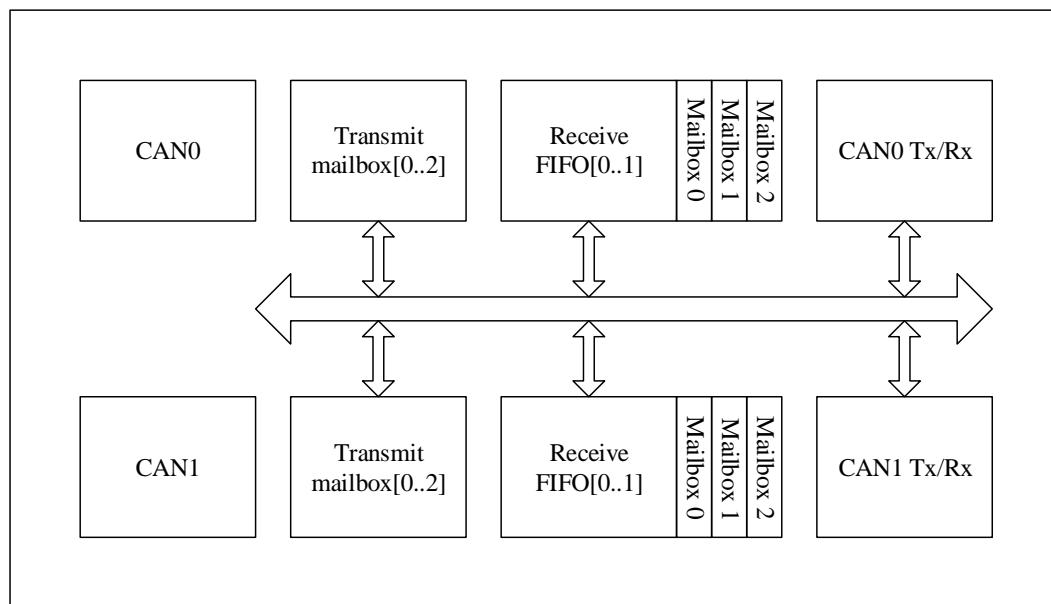
#### Time-triggered communication

- Disable retransmission automatically in time-triggered communication mode.
- 16-bit free timer
- Time stamp on SOF reception
- Time stamp sent in last two data bytes

## 22.3. Function overview

[Figure 22-1. CAN module block diagram](#) shows the CAN block diagram.

**Figure 22-1. CAN module block diagram**



### 22.3.1. 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 and 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 by hardware.

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 configuration 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 could communicate with other CAN communication nodes in normal working mode.

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.

## 22.3.2. 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 could detect 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 for monitoring the network messages.

### Loopback communication mode

Loopback communication mode means the transmitted messages are transferred into the Rx

FIFOs, the RX pin is disconnected from the CAN network and the TX pin can still send messages to the CAN network.

Setting LCMOD bit in CAN\_BT register to enter loopback communication mode, while clearing it to leave. Loopback communication mode is useful for 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 transmitted messages are transferred into the Rx FIFOs.

Setting LCMOD and SCMOD bit in CAN\_BT register to enter loopback and silent communication mode, while clearing them to leave.

Loopback and silent communication mode is used for self-test. The TX pin holds in recessive state. The RX pin holds in high impedance state.

### **Normal communication mode**

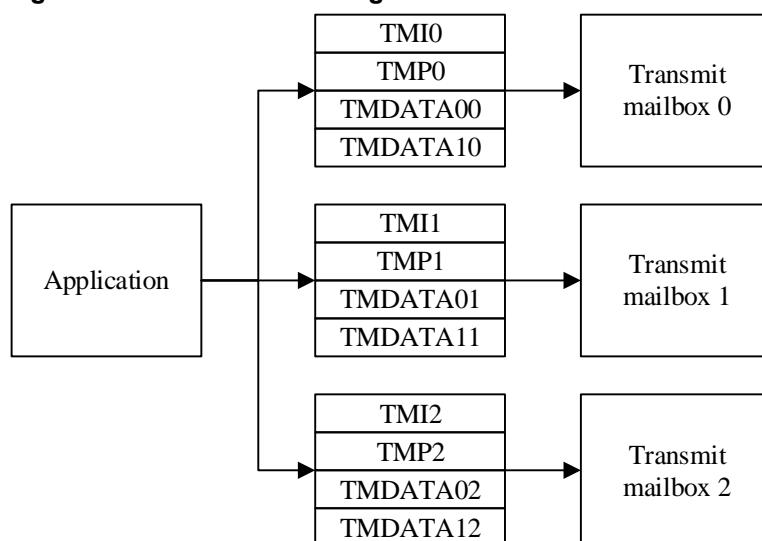
Normal communication mode is the default communication mode when the LCMOD and SCMOD bits in CAN\_BT register are cleared.

#### **22.3.3. Data transmission**

##### **Transmission register**

Three transmit mailboxes are used for the application. Transmit mailboxes are used by configuring four transmission registers: CAN\_TMIx, CAN\_TMPx, CAN\_TMDATA0x and CAN\_TMDATA1x. As is shown in [Figure 22-2. Transmission register](#).

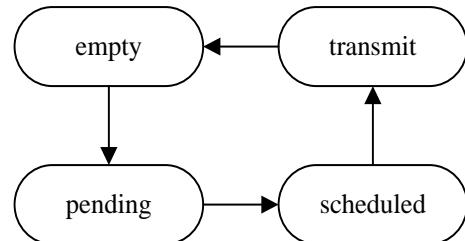
**Figure 22-2. Transmission register**



## Transmit mailbox state

A transmit mailbox can be used when it is free (**empty state**). If the mailbox is filled with data, set 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 scheduling the transmission (**Scheduled state**). A mailbox with highest priority enters into transmit state and starts transmitting the message (**transmit state**). After the message has been sent, the mailbox is free (**empty state**). As is shown in [Figure 22-3. State of transmit mailbox](#).

**Figure 22-3. State of transmit mailbox**



## Transmit status and error

The CAN\_TSTAT register includes the transmit status and error bits: MTF, MTFNERR, MAL, MTE.

- MTF: mailbox transmit finished. Typically, MTF is set when the frame in the transmit mailbox has been sent.
- MTFNERR: mailbox transmit finished with no error. MTFNERR is set when the frame in the transmit mailbox has been sent without any error.
- MAL: mailbox arbitration lost. MAL is set when the frame transmission is failed due to the arbitration lost.
- MTE: mailbox transmit error. MTE is set when the frame transmission is failed due to the error detected on the CAN bus.

## Steps of sending a frame

To send a frame through the CAN:

- Step 1: Select one free transmit mailbox.
- Step 2: Configure four transmission registers with the application's acquirement.
- Step 3: Set TEN bit in CAN\_TMIx register.
- Step 4: Check the transmit status. Typically, MTF and MTFNERR are set if transmission is successful.

## Transmission options

### Abort

MST bit in CAN\_TSTAT register can abort the transmission.

If the transmit mailbox's status is **Pending** or **Scheduled**, the abort of transmission can be done immediately.

In the **transmit** state, the abort of transmission does not take effect immediately until the transmission is finished. In case that the transmission is successful, the MTFNERR and MTF in CAN\_TSTAT are set and state changes to be **empty**. In case that the transmission is 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 that TFO is 1, the three transmit mailboxes work first-in first-out (FIFO).

In case that 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 firstly.

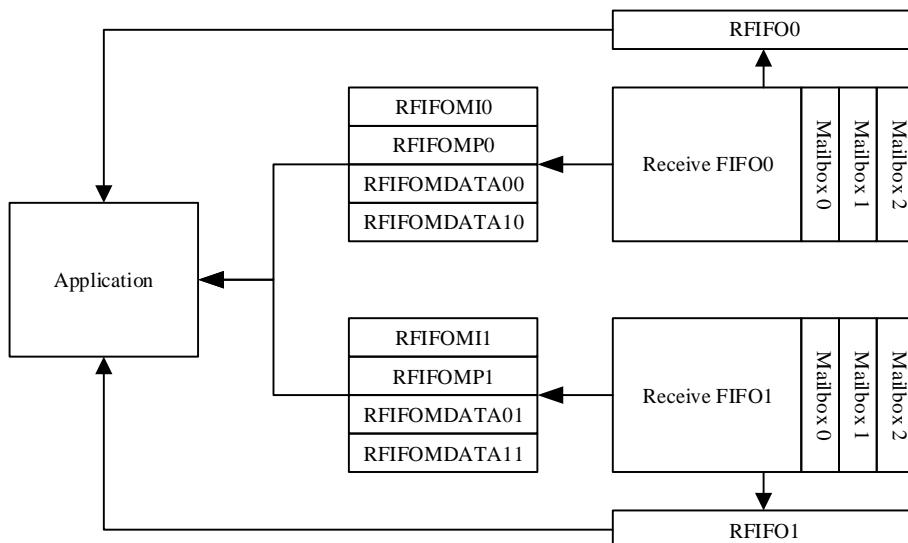
## 22.3.4. Data reception

### Reception register

Two Rx FIFOs are used for the application. Rx FIFOs are managed by 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 is shown in [Figure 22-4. Reception register](#).

**Figure 22-4. Reception register**



## Rx FIFO

Rx FIFO has three mailboxes. The reception frames are stored in the mailbox according to the arriving sequence. First arrived frame can be accessed by application firstly.

The number of frames in the Rx 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 Rx FIFO0, the frame data is stored in the CAN\_RFIFOMI0, CAN\_RFIFOMP0, CAN\_RFIFOMDATA00 and CAN\_RFIFOMDATA10 registers. After reading the current frame, set RFD bit in CAN\_RFIFO0 to release a frame in the Rx FIFO and the software can read the next frame.

## Rx FIFO status

RFL (Rx FIFO length) bits in CAN\_RFIFOx register is 0 when no frame is stored in the Rx FIFO and it is 3 when FIFOx is full.

When RFF bit in CAN\_RFIFOx register is set, it indicates FIFOx is full, at this time, RFL is 3.

When a new frame arrives after the FIFO has held three frames, the RFO bit in CAN\_RFIFOx register will be set, and it indicates FIFOx is overrun. 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 Rx FIFO and the last frame in the Rx FIFO is discarded.

## Steps of receiving a message

Step 1: Check the number of frames in the Rx FIFO.

Step 2: Read CAN\_RFIFOMIx, CAN\_RFIFOMPx, CAN\_RFIFOMDATA0x and CAN\_RFIFOMDATA1x.

Step 3: Set the RFD bit in CAN\_RFIFOx register.

### 22.3.5. Filtering function

The CAN receives frames from the CAN bus. If the frame passes the filter, it is stored in the Rx FIFOs. Otherwise, the frame will be discarded without intervention by the software.

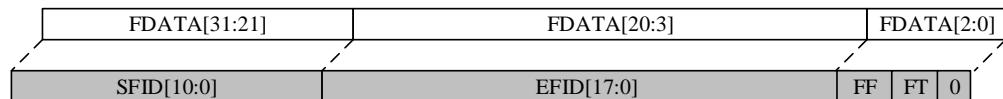
The identifier of frame is used for the matching of the filter.

#### Scale

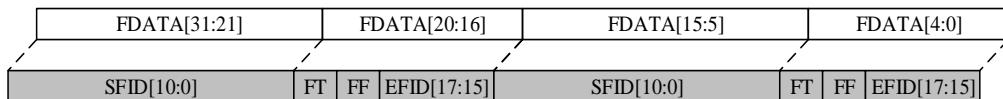
In GD32F30x XD/HD , the filter consists of 14 banks: bank0 to bank13. In GD32F30x CL, the filter consists of 28 banks: bank0 to bank27. Each bank has two 32-bit registers: CAN\_FxDATA0 and CAN\_FxDATA1.

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 is shown in [Figure 22-5. 32-bit filter](#).

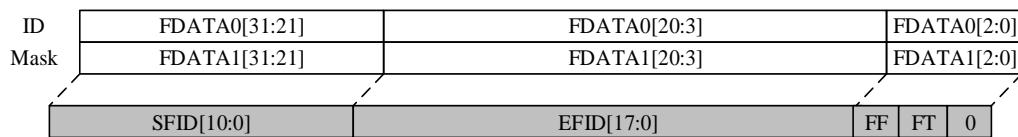
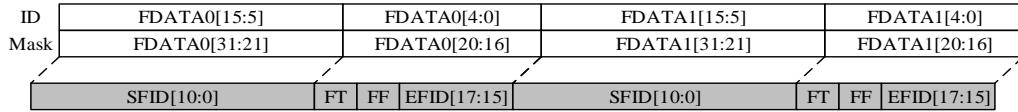
**Figure 22-5. 32-bit filter**


16-bit: SFID [10:0], FT, FF and EFID[17:15] bits. As is shown in [Figure 22-6. 16-bit filter](#).

**Figure 22-6. 16-bit filter**


### Mask mode

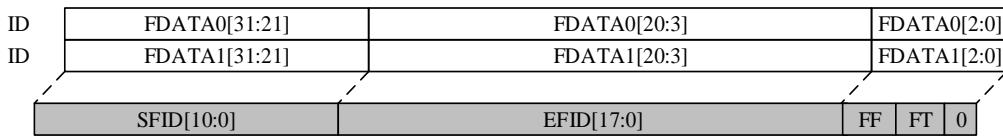
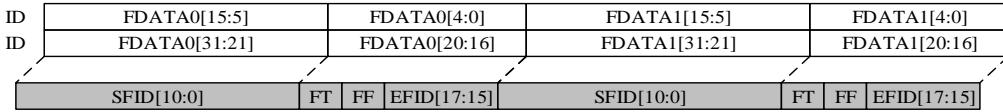
For the Identifier of a data frame to be filtered, the mask mode is used to specify which bits must be the same as the preset Identifier and which bits need not be judged. 32-bit mask mode example is shown in [Figure 22-7. 32-bit mask mode filter](#).

**Figure 22-7. 32-bit mask mode filter**

**Figure 22-8. 16-bit mask mode filter**


### List mode

The filter consists of frame identifiers. The filter can determine 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 22-9. 32-bit list mode filter](#).

**Figure 22-9. 32-bit list mode filter**

**Figure 22-10. 16-bit list mode filter**


## Filter number

Filter consists of some filter bank. According to the mode and the scale of each of the filter banks, filter has different effects.

For example, there are two filter banks. Bank0 is configured as 32-bit mask mode. Bank1 is configured as 32-bit list mode. The filter number is shown in [Table 22-1. 32-bit filter number](#).

**Table 22-1. 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 the bank will be stored in the FIFO0.

## Active

The filter bank needs to be activated if the bank is to be used, otherwise, the filter bank should be left deactivated.

## Filtering index

Each filter number corresponds to a filtering rule. When the frame which is associated with a filter number N passes the filters, the filter index is N. It stores in the FI bits in CAN\_RFIFOMPX.

Filter bank has filter index once it is associated with the FIFO no matter whether the bank is active or not.

The example about filtering index is shown in [Table 22-2. Filtering index](#).

**Table 22-2. Filtering index**

<b>Filter bank</b>	<b>FIFO0</b>	<b>Active</b>	<b>Filter number</b>	<b>Filter bank</b>	<b>FIFO1</b>	<b>Active</b>	<b>Filter number</b>	
0	F0DATA0-32bits-ID	Yes	0	2	F2DATA0[15:0]-16bits-ID	Yes	0	
	F0DATA1-32bits-Mask				F2DATA0[31:16]-16bits-Mask			
1	F1DATA0-32bits-ID	Yes	1	2	F2DATA1[15:0]-16bits-ID	Yes	1	
	F1DATA1-32bits-ID		2		F2DATA1[31:16]-16bits-Mask			
3	F3DATA0[15:0]-16bits-ID	No	3	4	F4DATA0-32bits-ID	No	2	
	F3DATA0[31:16]-16bits-Mask				F4DATA1-32bits-Mask			
	F3DATA1[15:0]-16bits-ID		4	5	F5DATA0-32bits-ID	No	3	
	F3DATA1[31:16]-16bits-Mask				F5DATA1-32bits-ID			
7	F7DATA0[15:0]-16bits-ID	No	5	6	F6DATA0[15:0]-16bits-ID	Yes	5	
	F7DATA0[31:16]-16bits-ID		6		F6DATA0[31:16]-16bits-ID		6	
	F7DATA1[15:0]-16bits-ID		7		F6DATA1[15:0]-16bits-ID		7	
	F7DATA1[31:16]-16bits-ID		8		F6DATA1[31:16]-16bits-ID		8	
8	F8DATA0[15:0]-16bits-ID	Yes	9	10	F10DATA0[15:0]-16bits-ID	No	9	
	F8DATA0[31:16]-16bits-ID		10		F10DATA0[31:16]-16bits-Mask			
	F8DATA1[15:0]-16bits-ID		11		F10DATA1[15:0]-16bits-ID		10	
	F8DATA1[31:16]-16bits-ID		12		F10DATA1[31:16]-16bits-Mask			
9	F9DATA0[15:0]-16bits-ID	Yes	13	11	F11DATA0[15:0]-16bits-ID	No	11	
	F9DATA0[31:16]-16bits-Mask				F11DATA0[31:16]-16bits-ID			
	F9DATA1[15:0]-16bits-ID		14		F11DATA1[15:0]-16bits-ID		13	
	F9DATA1[31:16]-16bits-Mask				F11DATA1[31:16]-16bits-ID			
12	F12DATA0-32bits-ID	Yes	15	13	F13DATA0-32bits-ID	Yes	15	
	F12DATA1-32bits-Mask				F13DATA1-32bits-ID			

### Priority

The filters have the priority rules:

1. 32-bits mode is higher than 16-bits mode.
2. List mode is higher than mask mode.
3. Smaller filter number has the higher priority.

### 22.3.6. 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 time points of message transmission are pre-defined.

In this mode, an internal 16-bit counter starts working, incrementing by 1 at each CAN bit time. This internal counter provides time stamps for sending and receiving data, stored in registers CAN\_RFIFOMPx and CAN\_TMPx.

The automatic retransmission is disabled in the time-triggered CAN communication.

### 22.3.7. Communication parameters

#### Automatic retransmission forbid mode

In time-triggered communication mode, the requirement for automatic retransmission must be disabled and can be met by setting ARD position 1 of the CAN\_CTL register.

In this mode, the data is sent only once, and if the transmission fails due to arbitration failure or bus error, the CAN bus controller does not automatically resend the data as usual.

At the end of sending, the MTF bit of register CAN\_TSTAT is hardware set to 1, and the sending status information can be obtained via MTFNERR, MAL, and MTE.

#### Bit time

On the bit-level, the CAN protocol uses synchronous bit transmission. This not only enhances the transmitting capacity but also requires a sophisticated method of bit synchronization. While bit synchronization in a character-oriented transmission (asynchronous) is performed upon the reception which the start bit is available with each character, the synchronous transmission protocol just need one start bit available at the beginning of a frame. To ensure that the receiver correctly reads the messages, resynchronization is required. Phase buffer segments' sample point of the front-end and back-end should be inserted a bit interval.

The CAN protocol regulates bus access by bit-wise arbitration. The signal propagated from sender to receiver and back to the sender must be completed within one bit-time. For synchronization, in addition to the phase buffer segments, a propagation delay segment is needed. The propagation delay segment is regarded as signal delays caused by transmitting and receiving nodes in the process of the signal propagation on the bus.

The normal bit time 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 ( $1 \times t_q$ ).

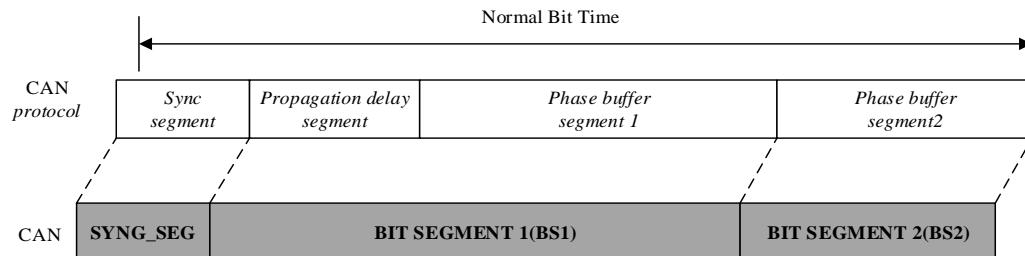
**Bit segment 1 (BS1):** It defines the location of the sample point. It includes the Propagation

delay segment and Phase buffer segment 1 in the CAN standard. Its duration is programmable from 1 to 16 time quanta but it may be automatically lengthened to compensate for positive phase drifts due to different frequency of the various nodes of the network.

**Bit segment 2 (BS2):** It defines the location of the transmit point. It represents the Phase buffer segment 2 in the CAN standard. Its duration is programmable from 1 to 8 time quanta but it may also be automatically shortened to compensate for negative phase drifts.

The bit time is shown as in the [\*\*Figure 22-11. The bit time.\*\*](#)

**Figure 22-11. The bit time**



The resynchronization Jump Width (SJW): it can be lengthened or shortened to compensate for the Synchronization error of the CAN network node. It is programmable from 1 to 4 time quanta.

A valid edge is defined as the first toggle in a bit time from dominant to recessive bus level before the controller sends a recessive bit.

If a valid edge is detected in BS1, not in SYNC\_SEG, BS1 is added up to SJW maximumly, so that the sample point is delayed.

Conversely, if a valid edge is detected in BS2, not in SYNC\_SEG, BS2 is cut down to SJW at most, so that the transmit point is moved earlier.

### Baud rate

The clock of the CAN derives from the APB1 bus. The CAN calculates its baud rate as follow:

$$\text{BaudRate} = \frac{1}{\text{Normal Bit Time}} \quad (22-1)$$

$$\text{Normal Bit Time} = t_{SYNC\_SEG} + t_{BS1} + t_{BS2} \quad (22-2)$$

with:

$$t_{SYNC\_SEG} = 1 \times t_q \quad (22-3)$$

$$t_{BS1} = (1 + BT.BS1) \times t_q \quad (22-4)$$

$$t_{BS2} = (1 + BT.BS2) \times t_q \quad (22-5)$$

$$t_q = (1 + BT.BAUDPSC) \times t_{PCLK1} \quad (22-6)$$

### 22.3.8. Error flags

The state of CAN bus can be reflected by Transmit Error Counter (TECNT) and Receive Error Counter (RECNT) of CAN\_ERR register. The value can be increased or decreased by the hardware according to the error, and the software can judge the stability of the CAN network by these values. For details on incorrect counting, refer to the CAN protocol section.

By using the CAN\_INTEN register (ERRIE bit, etc.), the software can control the interrupt generation when error is detected.

#### Bus-Off recovery

The CAN controller is in Bus-Off state when TECNT is over than 255. In This state, BOERR bit is set in CAN\_ERR register, and no longer able to transmit and receive messages.

According to the ABOR configuration in register CAN\_CTL, there are two ways to recover from Bus-Off (to an error active state). Both of these methods require the CAN bus controller in the Bus-Off state to detect the Bus-Off recovery sequence defined by CAN protocol (when CAN\_RX detects 128 consecutive 11-bit recessive bits) before automatic recovery.

If ABOR is set, it will be automatically recovered when a Bus-Off recovery sequence is detected.

If ABOR is cleared, CAN controller must be configured to enter initialization mode by setting IWMOD bit in CAN\_CTL register, then exit and enter nomal mode. After this operation, it will recover when the recovering sequence is detected.

### 22.3.9. CAN interrupts

The CAN bus controller occupies 4 interrupt vectors, which are controlled by the register CAN\_INTEN.

The interrupt sources can be classified as:

- 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 Rx FIFO0 interrupt can be generated by the following conditions:

- Rx FIFO0 not empty: RFL0 bits in the CAN\_RFIFO0 register are not '00' and RFNEIE0 in CAN\_INTEN register is set.
- Rx FIFO0 full: RFF0 bit in the CAN\_RFIFO0 register is set and RFFIE0 in CAN\_INTEN register is set.
- Rx FIFO0 overrun: RFO0 bit in the CAN\_RFIFO0 register is set and RFOIE0 in CAN\_INTEN register is set.

### Rx FIFO1 interrupt

The Rx FIFO1 interrupt can be generated by the following conditions:

- Rx FIFO1 not empty: RFL1 bits in the CAN\_RFIFO1 register are not '00' and RFNEIE1 in CAN\_INTEN register is set.
- Rx FIFO1 full: RFF1 bit in the CAN\_RFIFO1 register is set and RFFIE1 in CAN\_INTEN register is set.
- Rx 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.

## 22.4. CAN registers

CAN0 base address: 0x4000 6400

CAN1 base address: 0x4000 6800

### 22.4.1. 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
rw															
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	Debug freeze If the CANx_HOLD in DBG_CTL0 register is set, this bit defines the CAN controller is in debug freezing mode or normal working mode. If the CANx_HOLD in DBG_CTL0 register is cleared, this bit takes no effect. 0: CAN reception and transmission work normal even during debug 1: CAN reception and transmission stop working during debug
15	SWRST	Software reset 0: No effect 1: Reset CAN to enter sleep working mode. This bit is automatically reset to 0.
14:8	Reserved	Must be kept at reset value.
7	TTC	Time-triggered communication 0: Disable time-triggered communication 1: Enable time-triggered communication
6	ABOR	Automatic Bus-Off recovery 0: The Bus-Off state is left manually by software 1: The Bus-Off state is left automatically by hardware
5	AWU	Automatic wakeup

If this bit is set, the CAN leaves sleep working mode when CAN bus activity is detected, and SLPWMOD bit in CAN\_CTL register will be cleared automatically.

0: The sleeping working mode is left manually by software

1: The sleeping working mode is left automatically by hardware

4	ARD	Automatic retransmission disable
		0: Enable automatic retransmission
		1: Disable automatic retransmission
3	RFOD	Rx FIFO overwrite disable
		0: Enable Rx FIFO overwrite when Rx FIFO is full and overwrite the FIFO with the incoming frame
		1: Disable Rx FIFO overwrite when Rx FIFO is full and discard the incoming frame
2	TFO	Tx FIFO order
		0: Order with the identifier of the frame (the smaller identifier has higher priority)
		1: Order with first-in and first-out
1	SLPWMOD	Sleep working mode
		If this bit is set by software, the CAN enters sleep working mode after current transmission or reception is completed. 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 is 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

## 22.4.2. 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 entering sleep working mode</p> <p>This bit is set by hardware when entering sleep working mode, and cleared by hardware when the CAN is not in sleep working mode. This bit can also be cleared by software when writing 1 to this bit.</p> <p>0: CAN is not in the sleep working mode</p> <p>1: CAN is in the sleep working mode</p>
3	WUIF	<p>Status change interrupt flag of waking up from sleep working mode</p> <p>This bit is set when CAN bus activity event is detected in sleep working mode.</p> <p>This bit can be cleared by software when writing 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 the following events. 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 writing 1 to this bit.</p> <p>0: No error interrupt event</p> <p>1: Any error interrupt event has happened</p>
1	SLPWS	<p>Sleep working state</p> <p>This bit is set by hardware when the CAN enters sleep working mode after setting SLPWMOD bit in CAN_CTL register. If the CAN leaves normal working mode to sleep working mode, it must wait the current frame transmission or reception to be completed. This bit is cleared by hardware when the CAN leaves 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 leaving sleep working mode to normal working mode, this bit will be cleared after receiving 11 consecutive recessive bits from the CAN bus.</p> <p>0: CAN is not in the state of sleep working mode</p>

1: CAN is in the state of sleep working mode

0	IWS	Initial working state
		This bit is set by hardware when the CAN enters initial working mode after setting IWMOD bit in CAN_CTL register. If the CAN leaves normal working mode to initial working mode, it must wait the current frame transmission or reception to be completed. This bit is cleared by hardware when the CAN leaves initial working mode after clearing IWMOD bit in CAN_CTL register. If leaving initial working mode to normal working mode, this bit will be cleared after receiving 11 consecutive recessive bits from the CAN bus.
0:	CAN is not in the state of initial working mode	
1:	CAN is in the state of initial working mode	

### 22.4.3. 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	MTFO				
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 Tx FIFO  This bit is set by hardware when transmit mailbox 2 has the last sending order in the Tx FIFO with at least two frames pending.
30	TMLS1	Transmit mailbox 1 last sending in Tx FIFO  This bit is set by hardware when transmit mailbox 1 has the last sending order in the Tx FIFO with at least two frames pending.
29	TMLS0	Transmit mailbox 0 last sending in Tx FIFO  This bit is set by hardware when transmit mailbox 0 has the last sending order in the Tx FIFO with at least two frames pending.
28	TME2	Transmit mailbox 2 empty  0: Transmit mailbox 2 not empty 1: Transmit mailbox 2 empty

27	TME1	Transmit mailbox 1 empty 0: Transmit mailbox 1 not empty 1: Transmit mailbox 1 empty
26	TME0	Transmit mailbox 0 empty 0: Transmit mailbox 0 not empty 1: Transmit mailbox 0 empty
25:24	NUM[1:0]	These bits are the number of the Tx FIFO mailbox in which the frame will be transmitted if at least one mailbox is empty. These bits are the number of the Tx FIFO mailbox in which the frame will be transmitted at last if all mailboxes are full.
23	MST2	Mailbox 2 stop transmitting This bit is set by the software to stop mailbox 2 transmitting. This bit is reset by the hardware while the mailbox 2 is empty.
22:20	Reserved	Must be kept at reset value.
19	MTE2	Mailbox 2 transmit error This bit is set by hardware when the transmit error occurs. This bit is reset by writing 1 to this bit or MTF2 bit in CAN_TSTAT register. This bit is reset by hardware when next transmit starts.
18	MAL2	Mailbox 2 arbitration lost This bit is set when the arbitration lost occurs. This bit is reset by writing 1 to this bit or MTF2 bit in CAN_TSTAT register. This bit is reset by hardware when next transmit starts.
17	MTFNERR2	Mailbox 2 transmit finished with no error This bit is set when the transmission finishes and no error occurs. This bit is reset by writing 1 to this bit or MTF2 bit in CAN_TSTAT register. This bit is reset by hardware when the transmission finishes with error. 0: Mailbox 2 transmit finished with error 1: Mailbox 2 transmit finished with no error
16	MTF2	Mailbox 2 transmit finished This bit is set by hardware when the transmission finishes or aborts. This bit is reset by writing 1 to this bit or TEN bit in CAN_TMI2 is 1. 0: Mailbox 2 transmit is progressing 1: Mailbox 2 transmit finished
15	MST1	Mailbox 1 stop transmitting This bit is set by software to stop mailbox 1 transmitting. This bit is reset by hardware when the mailbox 1 is empty.
14:12	Reserved	Must be kept at reset value.
11	MTE1	Mailbox 1 transmit error

This bit is set by hardware when the transmit error occurs. This bit is reset by writting 1 to this bit or MTF1 bit in CAN\_TSTAT register. This bit is reset by hardware when next transmit starts.

10	MAL1	Mailbox 1 arbitration lost  This bit is set when the arbitration lost occurs. This bit is reset by writting 1 to this bit or MTF1 bit in CAN_TSTAT register. This bit is reset by hardware when next transmit starts.
9	MTFNERR1	Mailbox 1 transmit finished with no error  This bit is set when the transmission finishes and no error occurs. This bit is reset by writting 1 to this bit or MTF1 bit in CAN_TSTAT register. This bit is reset by hardware when the transmission finishes with error.  0: Mailbox 1 transmit finished with error 1: Mailbox 1 transmit finished with no error
8	MTF1	Mailbox 1 transmit finished  This bit is set by hardware when the transmission finishes or aborts. This bit is reset by writting 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 when 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 when the transmit error occurs. This bit is reset by writting 1 to this bit or MTF0 bit in CAN_TSTAT register. This bit is reset by hardware when next transmit starts.
2	MAL0	Mailbox 0 arbitration lost  This bit is set when the arbitration lost occurs. This bit is reset by writting 1 to this bit or MTF0 bit in CAN_TSTAT register. This bit is reset by hardware when next transmit starts.
1	MTFNERR0	Mailbox 0 transmit finished with no error  This bit is set when the transmission finishes and no error occurs. This bit is reset by writting 1 to this bit or MTF0 bit in CAN_TSTAT register. This bit is reset by hardware when the transmission finishes with error.  0: Mailbox 0 transmit finished with error 1: Mailbox 0 transmit finished with no error
0	MTF0	Mailbox 0 transmit finished  This bit is set by hardware when the transmission finishes or aborts. This bit is reset by writting 1 to this bit or TEN bit in CAN_TMI0 is 1.

0: Mailbox 0 transmit is progressing

1: Mailbox 0 transmit finished

#### **22.4.4. 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	Rx FIFO0 dequeue This bit is set by software to start dequeuing a frame from Rx FIFO0. This bit is reset by hardware when the dequeuing is done.
4	RFO0	Rx FIFO0 overfull This bit is set by hardware when Rx FIFO0 is overfull and reset by software when writing 1 to this bit. 0: The Rx FIFO0 is not overfull 1: The Rx FIFO0 is overfull
3	RFF0	Rx FIFO0 full This bit is set by hardware when Rx FIFO0 is full and reset by software when writing 1 to this bit. 0: The Rx FIFO0 is not full 1: The Rx FIFO0 is full
2	Reserved	Must be kept at reset value.
1:0	RFL0[1:0]	Rx FIFO0 length These bits are the length of the Rx FIFO0.

#### **22.4.5. 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															

rs rc\_w1 rc\_w1 r

Bits	Fields	Descriptions
31:6	Reserved	Must be kept at reset value.
5	RFD1	Rx FIFO1 dequeue This bit is set by software to start dequeuing a frame from Rx FIFO1. This bit is reset by hardware when the dequeuing is done.
4	RFO1	Rx FIFO1 overfull This bit is set by hardware when Rx FIFO1 is overfull and reset by writing 1 to this bit. 0: The Rx FIFO1 is not overfull 1: The Rx FIFO1 is overfull
3	RFF1	Rx FIFO1 full This bit is set by hardware when Rx FIFO1 is full and reset by writing 1 to this bit. 0: The Rx FIFO1 is not full 1: The Rx FIFO1 is full
2	Reserved	Must be kept at reset value.
1:0	RFL1[1:0]	These bits are the length of the Rx FIFO1.

#### 22.4.6. 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															
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 disabled 1: Sleep working interrupt enabled
16	WIE	Wakeup interrupt enable 0: Wakeup interrupt disabled 1: Wakeup interrupt enabled
15	ERRIE	Error interrupt enable 0: Error interrupt disabled 1: Error interrupt enabled
14:12	Reserved	Must be kept at reset value.
11	ERRNIE	Error number interrupt enable 0: Error number interrupt disabled 1: Error number interrupt enabled
10	BOIE	Bus-Off interrupt enable 0: Bus-Off interrupt disabled 1: Bus-Off interrupt enabled
9	PERRIE	Passive error interrupt enable 0: Passive error interrupt disabled 1: Passive error interrupt enabled
8	WERRIE	Warning error interrupt enable 0: Warning error interrupt disabled 1: Warning error interrupt enabled
7	Reserved	Must be kept at reset value.
6	RFOIE1	Rx FIFO1 overfull interrupt enable 0: Rx FIFO1 overfull interrupt disabled 1: Rx FIFO1 overfull interrupt enabled
5	RFFIE1	Rx FIFO1 full interrupt enable 0: Rx FIFO1 full interrupt disabled 1: Rx FIFO1 full interrupt enabled
4	RFNEIE1	Rx FIFO1 not empty interrupt enable 0: Rx FIFO1 not empty interrupt disabled 1: Rx FIFO1 not empty interrupt enabled
3	RFOIE0	Rx FIFO0 overfull interrupt enable 0: Rx FIFO0 overfull interrupt disabled 1: Rx FIFO0 overfull interrupt enabled

---

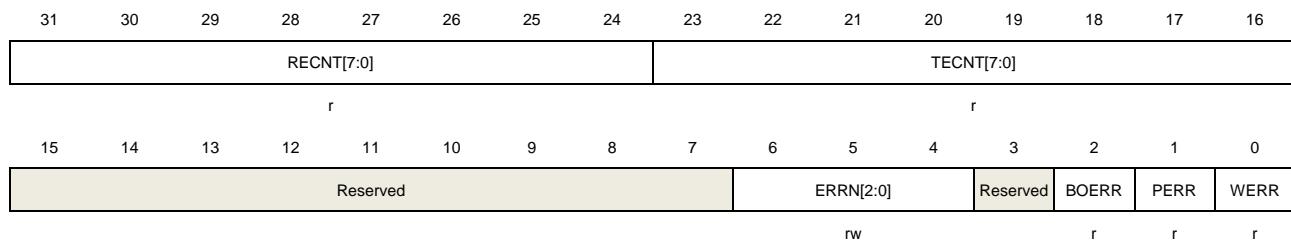
2	RFFIE0	Rx FIFO0 full interrupt enable 0: Rx FIFO0 full interrupt disabled 1: Rx FIFO0 full interrupt enabled
1	RFNEIE0	Rx FIFO0 not empty interrupt enable 0: Rx FIFO0 not empty interrupt disabled 1: Rx FIFO0 not empty interrupt enabled
0	TMEIE	Transmit mailbox empty interrupt enable 0: Transmit mailbox empty interrupt disabled 1: Transmit mailbox empty interrupt enabled

#### 22.4.7. Error register (CAN\_ERR)

Address offset: 0x18

Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
31:24	RECNT[7:0]	Receive error count defined by the CAN standard
23:16	TECNT[7:0]	Transmit error count defined by the CAN standard
15:7	Reserved	Must be kept at reset value.
6:4	ERRN[2:0]	Error number These bits indicate the error status of bit transformation. They are updated by hardware. When the bit transformation is successful, they are equal to 0. 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 hardware.
1	PERR	Passive error Whenever the TECNT or RECNT is greater than 127, the bit will be set by hardware.
0	WERR	Warning error Whenever the TECNT or RECNT is greater than or equal to 96, the bit will be set by hardware.

#### 22.4.8. 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 disabled 1: Silent communication enabled
30	LCMOD	Loopback communication mode 0: Loopback communication disabled 1: Loopback communication enabled
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
-----	--------------	--

### 22.4.9. Transmit mailbox identifier register (CAN\_TMIx) (x=0..2)

Address offset: 0x180, 0x190, 0x1A0

Reset value: 0xXXXX 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]					
rw										rw					
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		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 software when one frame will be transmitted and reset by hardware when the transmit mailbox is empty. 0: Transmit disabled 1: Transmit enabled

### 22.4.10. 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															

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 disabled 1: Time stamp enabled. The TS[15:0] will be transmitted in the DB6 and DB7 in DL. This bit is available when 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.

#### 22.4.11. 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															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DB1[7:0]								DB0[7:0]							
rw															

Bits	Fields	Descriptions
31:24	DB3[7:0]	Data byte 3

---

23:16	DB2[7:0]	Data byte 2
15:8	DB1[7:0]	Data byte 1
7:0	DB0[7:0]	Data byte 0

#### 22.4.12. Transmit mailbox data1 register (CAN\_TMDATA1x) (x=0..2)

Address offset: 0x18C, 0x19C, 0x1AC

Reset value: 0xXXXX XXXX

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DB7[7:0]								DB6[7:0]							
rw								rw							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DB5[7:0]								DB4[7:0]							
rw								rw							

Bits	Fields	Descriptions
31:24	DB7[7:0]	Data byte 7
23:16	DB6[7:0]	Data byte 6
15:8	DB5[7:0]	Data byte 5
7:0	DB4[7:0]	Data byte 4

#### 22.4.13. Receive FIFO mailbox identifier register (CAN\_RFIFOMIx) (x=0,1)

Address offset: 0x1B0, 0x1C0

Reset value: 0xXXXX XXXX

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

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]			
r												r			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EFID[12:0]												FF FT Reserved			
r												r r			

Bits	Fields	Descriptions
31:21	SFID[10:0]/EFID[28:1]	The frame identifier

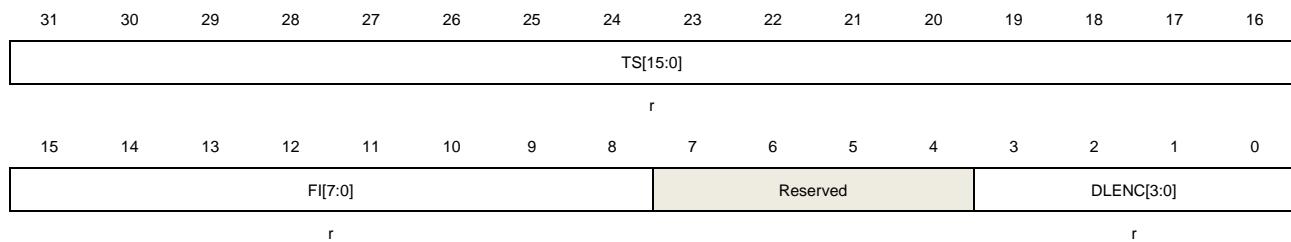
	8]	SFID[10:0]: Standard format frame identifier EFID[28:18]: Extended format frame identifier
20:16	EFID[17:13]	The frame identifier EFID[17:13]: Extended format frame identifier
15:3	EFID[12:0]	The frame identifier EFID[12:0]: Extended format frame identifier
2	FF	Frame format 0: Standard format frame 1: Extended format frame
1	FT	Frame type 0: Data frame 1: Remote frame
0	Reserved	Must be kept at reset value.

#### 22.4.14. Receive FIFO mailbox property register (CAN\_RFIFOMP $x$ ) ( $x=0,1$ )

Address offset: 0x1B4, 0x1C4

Reset value: 0xFFFF XXXX

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



Bits	Fields	Descriptions
31:16	TS[15:0]	Time stamp The time stamp of frame in transmit mailbox.
15:8	FI[7:0]	Filtering index The index of the filter which the frame passes.
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.

### **22.4.15. Receive FIFO mailbox data0 register (CAN\_RFIFOMDATA0x) (x=0,1)**

Address offset: 0x1B8, 0x1C8

Reset value: 0xFFFF XXXX

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

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

### **22.4.16. Receive FIFO mailbox data1 register (CAN\_RFIFOMDATA1x) (x=0,1)**

Address offset: 0x1BC, 0x1CC

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
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

### **22.4.17. Filter control register (CAN\_FCTL) (Just for CAN0)**

Address offset: 0x200

Reset value: 0x2A1C 0E01

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

The filter control register with GD32F30x XD/HD :

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															

The filter control register with GD32F30x CL:

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

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:14	Reserved	Must be kept at reset value.
13:8	HBC1F[5:0]	Header bank of CAN1 filter These bits are set and cleared by software to define the first bank for CAN1 filter. Bank0 ~ Bank HBC1F-1 is used for CAN0. Bank HBC1F ~ Bank27 is used for CAN1. When set 0, no bank used for CAN0. When set 28, no bank used for CAN1.
7:1	Reserved	Must be kept at reset value.
0	FLD	Filter lock disable 0: Filter lock enabled 1: Filter lock disabled

### **22.4.18. Filter mode configuration register (CAN\_FMCFG) (Just for CAN0)**

Address offset: 0x204

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit). This register can be modified only when FLD bit in CAN\_FCTL register is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
			Reserved		FMOD27	FMOD26	FMOD25	FMOD24	FMOD23	FMOD22	FMOD21	FMOD20	FMOD19	FMOD18	FMOD17	FMOD16
					rw											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
FMOD15	FMOD14	FMOD13	FMOD12	FMOD11	FMOD10	FMOD9	FMOD8	FMOD7	FMOD6	FMOD5	FMOD4	FMOD3	FMOD2	FMOD1	FMOD0	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bits	Fields	Descriptions
31:28	Reserved	Must be kept at reset value.
27:0	FMODx	Filter mode 0: Filter x with mask mode 1: Filter x with list mode

#### 22.4.19. Filter scale configuration register (CAN\_FSCFG) (Just for CAN0)

Address offset: 0x20C

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit). This register can be modified only when FLD bit in CAN\_FCTL register is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
			Reserved		FS27	FS26	FS25	FS24	FS23	FS22	FS21	FS20	FS19	FS18	FS17	FS16
					rw											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
FS15	FS14	FS13	FS12	FS11	FS10	FS9	FS8	FS7	FS6	FS5	FS4	FS3	FS2	FS1	FS0	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bits	Fields	Descriptions
31:28	Reserved	Must be kept at reset value.
27:0	FSx	Filter scale 0: Filter x with 16-bit scale 1: Filter x with 32-bit scale

#### 22.4.20. Filter associated FIFO register (CAN\_FAFIFO) (Just for CAN0)

Address offset: 0x214

Reset value: 0x0000 0000

This register has to be accessed by word(32-bit). This register can be modified only when FLD bit in CAN\_FCTL register is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
			Reserved		FAF27	FAF26	FAF25	FAF24	FAF23	FAF22	FAF21	FAF20	FAF19	FAF18	FAF17	FAF16
					rw											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
FAF15	FAF14	FAF13	FAF12	FAF11	FAF10	FAF9	FAF8	FAF7	FAF6	FAF5	FAF4	FAF3	FAF2	FAF1	FAF0	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bits	Fields	Descriptions
31:28	Reserved	Must be kept at reset value.
27:0	FAFx	Filter associated FIFO 0: Filter x associated with FIFO0 1: Filter x associated with FIFO1

#### 22.4.21. Filter working register (CAN\_FW) (Just for CAN0)

Address offset: 0x21C

Reset value: 0x0000 0000

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
			Reserved		FW27	FW26	FW25	FW24	FW23	FW22	FW21	FW20	FW19	FW18	FW17	FW16
					rw											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
FW15	FW14	FW13	FW12	FW11	FW10	FW9	FW8	FW7	FW6	FW5	FW4	FW3	FW2	FW1	FW0	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bits	Fields	Descriptions
31:28	Reserved	Must be kept at reset value.
27:0	FWx	Filter working 0: Filter x working disabled 1: Filter x working enabled

#### 22.4.22. Filter x data y register (CAN\_FxDATAY) (x=0..27, y=0,1) (Just for CAN0)

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	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
FD15	FD14	FD13	FD12	FD11	FD10	FD9	FD8	FD7	FD6	FD5	FD4	FD3	FD2	FD1	FD0		

rw																	
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:0	FDx	Filter data Mask mode 0: Mask match disable 1: Mask match enable List mode 0: List identifier bit is 0 1: List identifier bit is 1

## 23. Ethernet (ENET)

### 23.1. Overview

This chapter describes the Ethernet peripheral module. There is a media access controller (MAC) designed in Ethernet module to support 10/100Mbps interface speed. For more efficient data transfer between Ethernet and memory, a DMA controller is designed in this module. The support interface protocol for Ethernet is media independent interface (MII) and reduced media independent interface (RMII). This module is mainly compliant with the following two standards: IEEE 802.3-2002 and IEEE 1588-2008. Ethernet module is compatible with GD32F10x series.

### 23.2. Characteristics

#### MAC feature

- 10Mbit/s and 100Mbit/s data transfer rates support.
- MII and RMII interface support
- Loopback mode support for diagnosis
- CSMA/CD Protocol for Half-duplex back-pressure operation support.
- IEEE 802.3x flow control protocol support. Automatic delay a pause time which is decoded from a receive pause frame after current transmitting frame complete. MAC automatically transmits pause frame or back pressure feature depending on fill level of RxFIFO in Full-duplex mode or in Half-duplex mode.
- Automatic transmission of pause frame on assertion and de-assertion of flow control input frame. Zero-quanta pause time length frame for Full-duplex operation. IEEE 802.3x flow control for Full-duplex operation support. Back pressure feature to the MAC core based on RxFIFO fill level (Cut-Through mode) support. IEEE 802.3x flow control for Half-duplex operation support.
- Software configurable for automatic PAD/CRC generation in transmits operation.
- Software configurable for automatic PAD/CRC stripping in receives operation.
- Software configurable for frame length to support standard frames with sizes up to 16 KB.
- Software configurable for inter-frame gap (40-96 bit times in steps of 8).
- Support different receiving filter mode.
- IEEE 802.1Q VLAN tag detection function support for reception frames.
- Support mandatory network statistics standard (RFC2819/RFC2665).
- Two types of wakeup frame detection: LAN remote wakeup frame and AMD Magic Packet™ frames.
- Support checking IPv4 header checksum and TCP, UDP, or ICMP checksum encapsulated in IPv4 or IPv6 datagram.

- Support Ethernet frame time stamping for both transmit and receive operation, which describes in IEEE 1588-2008, and 64 bit time stamps are given in each frame's status.
- Two independent FIFO of 2K Byte for transmitting and receiving.
- Support special condition frame discards handling, e.g. late collision, excessive collisions, excessive deferral or underrun.
- Calculate and insert IPv4 header checksum and TCP, UDP, or ICMP checksum in frame transmit under Store-and-Forward mode.

### DMA Feature

- Two types of descriptor addressing: Ring and Chain.
- Each descriptor can transfer up to 8 KB of data.
- Programmable normal and abnormal interrupt for many status conditions
- Round-robin or fixed-priority arbitration between reception and transmission controller.

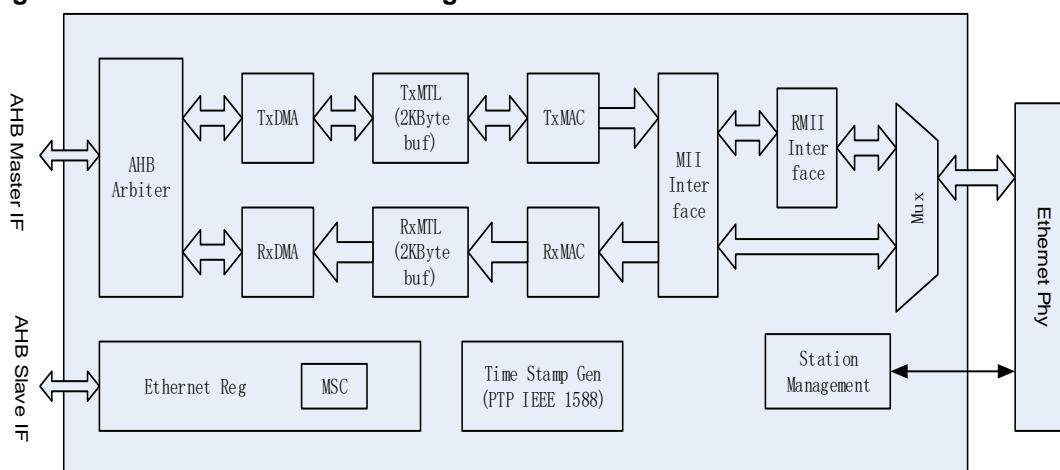
### PTP Feature

- Support IEEE 1588 time synchronization function.
- Support two correction methods: Coarse or Fine.
- Pulse per second output.
- Preset target time reaching trigger and interrupt

#### 23.2.1. Block diagram

The Ethernet module is composed of a MAC module, MII/RMII module and a DMA module by descriptor control.

**Figure 23-1. ENET module block diagram**



The MAC module is connected to the external PHY by MII or RMII through one selection bit (refer to AFIO\_PCF0 register). The SMI (Station Management Interface) is used to configure and manage external PHY.

Transmitting data module includes:

- TxDMA controller, used to read descriptors and data from memory and writes status to memory.
- TxMTL, used to control, management and store the transmit data. TxFIFO is implemented in this module and used to cache transmitting data from memory for MAC transmission.
- The MAC transmission relative control registers, used to control frame transmit.

Receiving data module includes:

- RxDMA controller, used to read descriptors from memory and writes received frame data and status to memory.
- RxMTL, used to control, management and store reception data. RxFIFO is implemented in this module and used to temporarily store received frame data before forwarding them into the system physical memory.
- The MAC reception relative control registers, used to control frame receive and marked the receiving state. Also a receiving filter with a variety of filtering mode is implemented in MAC, used to filter out specific Ethernet frame

**Note:** The AHB clock frequency must be at least 25 MHz when the Ethernet is used.

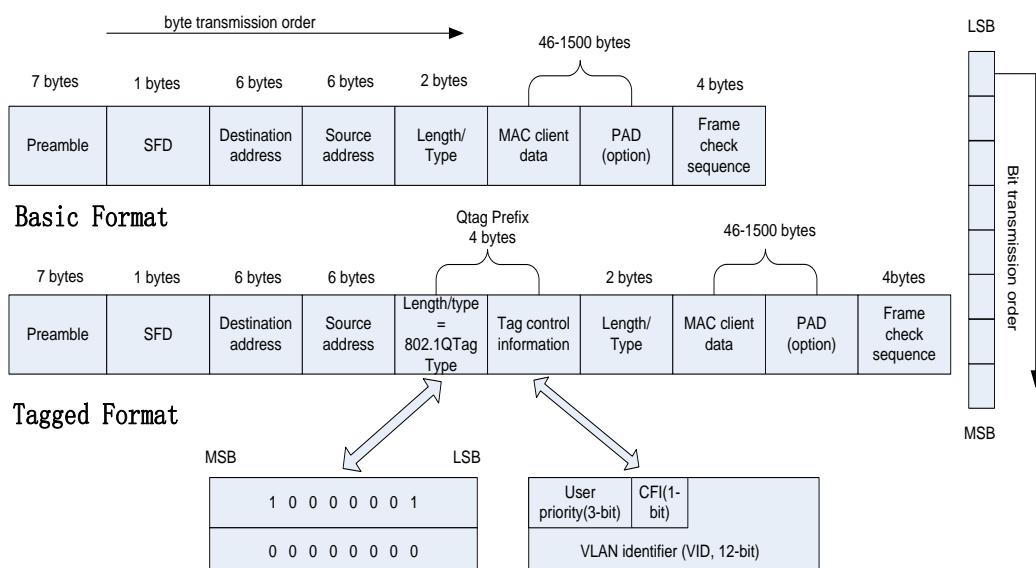
### 23.2.2. MAC 802.3 Ethernet packet description

Data communication of MAC can use two frame formats:

- Basic MAC frame format.
- Tagged MAC frame format (extension of the basic MAC frame format).

**Figure 23-2. MAC/Tagged MAC frame format** describes the structure of the frame (Basic and Tagged) that includes the following fields:

**Figure 23-2. MAC/Tagged MAC frame format**



**Note:** The Ethernet controller transmits each byte at LSB first except FCS field.

CRC calculation data comes from all bytes in the frame except the Preamble and SFD domain. The Ethernet frame's 32-bit CRC calculation value generating polynomial is fixed 0x04C11DB7 and this polynomial is used in all 32-bit CRC calculation places in Ethernet module, as follows:

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

### 23.2.3. Ethernet signal description

**Table 23-1. Ethernet pin configuration** below shows the MAC module that pin is used default and remapping functions and specific configuration in MII/RMII mode.

**Table 23-1. Ethernet pin configuration**

MAC signals	Pin	Pin configuration	MII default	MII remap	RMII default	RMII remap
ETH_MDC	PC1	AF output push-pull highspeed (50 MHz)	MDC		MDC	
ETH_MII_TXD2	PC2	AF output push-pull highspeed (50 MHz)	TXD2			
ETH_MII_TX_CLK	PC3	Floating input (reset state)	TX_CLK			
ETH_MII_CRS	PA0	Floating input (reset state)	CRS			
ETH_RX_CLK ETH_RMII_REF_CLK	PA1	Floating input (reset state)	RX_CLK		REF_CLK	
ETH_MDIO	PA2	AF output push-pull highspeed (50 MHz)	MDIO		MDIO	
ETH_MII_COL	PA3	Floating input (reset state)	COL			
ETH_MII_RX_DV ETH_RMII_CRS_DV	PA7	Floating input (reset state)	RX_DV		CRS_DV	
ETH_MII_RXD0 ETH_RMII_RXD0	PC4	Floating input (reset state)	RXD0		RXD0	
ETH_MII_RXD1 ETH_RMII_RXD1	PC5	Floating input (reset state)	RXD1		RXD1	
ETH_MII_RXD2	PB0	Floating input (reset state)	RXD2			
ETH_MII_RXD3	PB1	Floating input (reset state)	RXD3			
ETH_PPS_OUT	PB5	AF output push-pull highspeed (50 MHz)	PPS_OUT		PPS_OUT	
ETH_MII_TXD3	PB8	AF output push-pull highspeed (50 MHz)	TXD3			
ETH_MII_RX_ER	PB10	Floating input	RX_ER			

		(reset state)				
ETH_MII_TX_EN	PB11	AF output push-pull highspeed (50 MHz)	TX_EN		TX_EN	
ETH_RMII_TX_EN						
ETH_MII_TXD0	PB12	AF output push-pull highspeed (50 MHz)	TXD0		TXD0	
ETH_RMII_TXD0						
ETH_MII_TXD1	PB13	AF output push-pull highspeed (50 MHz)	TXD1		TXD1	
ETH_RMII_CRS_DV	PD8	Floating input (reset state)		RX_DV		CRS_DV
ETH_MII_RXD0	PD9	Floating input (reset state)		RXD0		RXD0
ETH_RMII_RXD0						
ETH_MII_RXD1	PD10	Floating input (reset state)		RXD1		RXD1
ETH_RMII_RXD1						
ETH_MII_RXD2	PD11	Floating input (reset state)		RXD2		
ETH_MII_RXD3	PD12	Floating input (reset state)		RXD3		

## 23.3. Function overview

### 23.3.1. Interface configuration

The Ethernet block can transmit and receive Ethernet packets from an off-chip Ethernet PHY connected through the MII/RMII interface. MII or RMII mode is selected by software and carry on the PHY management through the SMI interface.

#### SMI: Station management interface

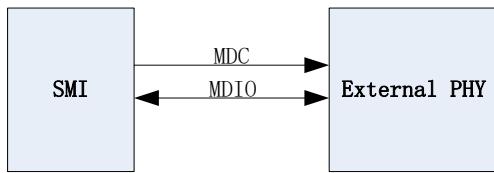
SMI is designed to access and configure PHY's configuration.

Station management interface (SMI) is performed through two wires to communicate with the external PHY: one clock line (MDC) and one data line (MDIO), it can access to the any PHY register. The interface supports accessing up to 32 PHYs, but only one register in one PHY can be addressed at the same time.

MDC and MDIO specific functions as follows:

- **MDC:** A clock of maximum frequency is 2.5 MHz. The pin remains low level when it is in idle state. The minimum high or low level lasts time of MDC must be 160ns, and the minimum period of MDC must be 400ns when it is in data transmission state.
- **MDIO:** Used to transfer data in conjunction with the MDC clock line, receiving data from external PHY or sending data to external PHY.

**Figure 23-3. Station management interface signals**



### **SMI write operation**

Applications need to write transmission data to the ENET\_MAC\_PHY\_DATA register and operate the ENET\_MAC\_PHY\_CTL register as follows:

- 1) Set the PHY device address and PHY register address, and set PW to 1, so that can select write mode.
- 2) Set PB bit to start transmission. In the process of transaction PB is always high until the transfer is complete. Hardware will clear PB bit automatically.

The application can be aware of whether a transaction is complete or not through checking PB bit. When PB is 1, it means the application should not change the PHY address register contents and the PHY data register contents because of operation is running. Before writing PB bit to 1, application must poll the PB bit until it is 0.

### **SMI read operation**

Applications need to operate the ENET\_MAC\_PHY\_CTL register as follows:

- 1) Set the PHY device address and PHY register address and set PW to 0, so that can select read mode.
- 2) Set PB bit to start reception. In the process of transaction PB is always high until the transfer is complete. Hardware will clear PB bit automatically.

The application can be aware of whether a transaction is complete or not through checking PB bit. When PB is 1, it means the application should not change the PHY address register contents and the PHY data register contents because of operation is running. Before writing PB bit to 1, application must poll the PB bit until it is 0.

**Note:** Because the PHY register address 16-31 register function is defined by each manufacturer, access different PHY device's this part should see according to the manufacturer's manual to adjust the parameters of applications. Details of catalog that firmware library currently supports the PHY device can refer to firmware library related instructions.

### **SMI clock selection**

The SMI clock is generated by dividing application clock (AHB clock). In order to guarantee the MDC clock frequency is no more than 2.5MHz, application should set appropriate division factor according to the different AHB clock frequency. The following table lists the frequency factor corresponding AHB clock selection.

**Table 23-2. Clock range**

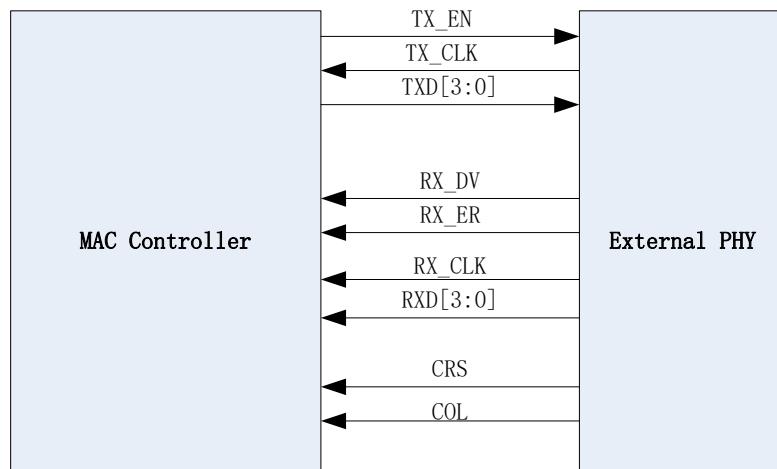
AHB clock	MDC clock	Selection
35~60MHz	AHB clock/26	0x3
20~35MHz	AHB clock/16	0x2
100~120 MHz	AHB clock/62	0x1
60~100MHz	AHB clock/42	0x0

### MII/RMII selection

The application can select the MII or RMII mode through the configuration bit 23 of the AFIO\_PCF0 register ENET\_PHY\_SEL while the Ethernet controller is under reset state or before enabling the clocks. The MII mode is set by default.

### MII: Media independent interface

The media-independent interface (MII) defines the interconnection between the MAC sub-layer and the PHY for data transfer at 10 Mbit/s or 100 Mbit/s.

**Figure 23-4. Media independent interface signals**


- **MII\_TX\_CLK**: clock signal for transmitting data. For the data transmission speed of 10Mbit/s, the clock is 2.5MHz, for the data transmission speed of 100Mbit/s, the clock is 25MHz.

- **MII\_RX\_CLK**: Clock signal for receiving data. For the data transmission speed of 10Mbit/s, the clock is 2.5MHz, for the data transmission speed of 100Mbit/s, the clock is 25MHz.

- **MII\_TX\_EN**: Transmission enable signal. It must be asserted synchronously with the first bit of the preamble and must remain asserted while all bits to be transmitted are presented to the MII.

- **MII\_TXD[3:0]**: Transmit data line, each 4 bit data transfer, data is valid when the MII\_TX\_EN signal is effective. MII\_TXD[0] is the least significant bit and MII\_TXD[3] is the most significant bit. While MII\_TX\_EN is de-asserted the transmit data must have no effect upon the PHY.

- **MII\_CRS**: Carrier sense signal, only working in Half-duplex mode and controlled by the PHY.

It is active when either transmit or receive medium is in non idle state. The PHY must ensure that the MII\_CRS signal remains asserted throughout the duration of a collision condition. This signal is not required to transition synchronously with respect to the TX and RX clock.

- **MII\_COL:** Collision detection signal, only working in Half-duplex mode, controlled by the PHY. It is active when a collision on the medium is detected and must remain active while the collision condition continues. This signal is not required to transition synchronously with respect to the TX and RX clock.
- **MII\_RXD[3:0]:** Receive data line, each 4 bit data transfer; data are valid when the MII\_RX\_DV signal is effective. MII\_RXD[0] is the least significant bit and MII\_RXD[3] is the most significant bit. While MII\_RX\_DV is de-asserted and MII\_RX\_ER is asserted, a specific MII\_RXD[3:0] value is used to indicate specific information (see [Table 23-3. Rx interface signal encoding](#)).
- **MII\_RX\_DV:** Receive data valid signal, controlled by the PHY. It is asserted when PHY is presenting data on the MII for reception. It must be asserted synchronously with the first 4-bit of the frame and must remain asserted while all bits to be transmitted are presented on the MII. It must be de-asserted prior to the first clock cycle that follows the final 4-bit. In order to receive the frame correctly, the effective signal starting no later than the SFD field.
- **MII\_RX\_ER:** Receive error signal. It must be asserted for one or more RX clock to indicate MAC detected an error in the receiving process. The specific error reason needs to cooperate with the state of the MII\_RX\_DV and the MII\_RXD[3:0] data value (see [Table 23-3. Rx interface signal encoding](#)).

**Table 23-3. Rx interface signal encoding**

MII_RX_ER	MII_RX_DV	MII_RXD[3:0]	Description
0	0	0000 to 1111	Normal inter-frame
0	1	0000 to 1111	Normal reception frame data
1	0	0000	Normal inter-frame
1	0	0001 to 1101	Reserved
1	0	1110	False carrier indication
1	0	1111	Reserved
1	1	0000 to 1111	Data reception with errors

### MII clock sources

To generate both TX\_CLK and RX\_CLK clock signals, the external PHY needs an external 25MHz clock. This 25MHz clock does not require the same one with MAC clock. It can use the external 25MHz crystal or the output clock of microcontroller's CK\_OUT0 pin. If the clock source is selected from CK\_OUT0 pin, the MCU needs to configure the appropriate PLL to ensure the output frequency of CK\_OUT0 pin is 25MHz.

### RMII: Reduced media independent interface

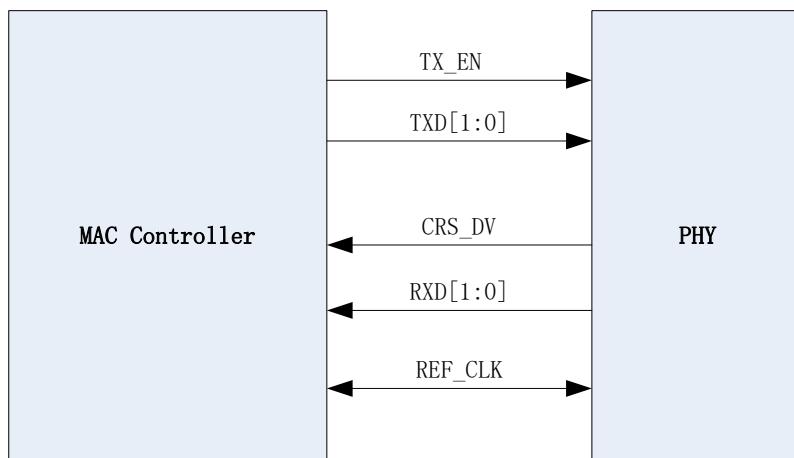
The reduced media-independent interface (RMII) specification reduces the pin count during

Ethernet communication. According to the IEEE 802.3 standard, an MII contains 16 pins for data and control. The RMII specification is dedicated to reduce the pin count to 7.

The RMII block has the following characteristics:

- The clock signal needs to be increased to 50MHz and only one clock signal.
- MAC and external PHY use the same clock source
- Using the 2-bit wide data transceiver

**Figure 23-5. Reduced media-independent interface signals**



#### MII/RMII bit transmission order

No matter which interface (MII or RMII) is selected, the bit order of transmit/receive is LSB first.

The deference between MII and RMII is bit number and sending times. MII is low 4bits first and then high 4bits, but RMII is the lowest 2bits, low 2bits, high 2bits and the highest 2bits.

For example: a byte value is: 10011101b (left to right order: high to low)

Transmission order for MII use 2 cycles: 1101 -> 1001 (left to right order: high to low)

Transmission order for RMII use 4 cycles: 01 -> 11 -> 01 -> 10 (left to right order: high to low)

#### RMII clock sources

To ensure the synchronization of the clock source, the same clock source is selected for the MAC and external PHY which is called REF\_CLK. The REF\_CLK input clock can be connected to the external 50MHz crystal or microcontroller CK\_OUT0 pin. If the clock source is from CK\_OUT0 pin, then the MCU needs to configure the appropriate PLL to ensure the output frequency of CK\_OUT0 pin is 50MHz.

### 23.3.2. MAC function overview

MAC module can achieve the following functions:

Data package (transmission and reception)

- Frame detecting/decoding and frame boundary delimitation.

- Addressing (handling source address and destination address).
- Error conditions detect.

Medium access management in Half-duplex mode

- Medium allocation (prevent conflicts).
- Conflict resolution (dealing with conflicts).

The MAC module can work in two modes

- Half-duplex mode: with the CSMA/CD algorithm to contend for using of the physical medium, at the same time only one transmission direction is active between two stations is active.
- Full-duplex mode: simultaneous transmission and reception without any conflict mode, if all of the following conditions are satisfied: 1) PHY supports the feature of transmission and reception operations at the same time. 2) Only two devices connect to the LAN and the two devices are both configured for Full-duplex mode.

### Transmission process of MAC

All transactions are controlled by the dedicated DMA controller and MAC in Ethernet. After received the sending instruction, the TxDMA fetches the transmit frames from system memory and pushes them into the Tx FIFO, then the data in Tx FIFO are popped to MAC for sending on MII/RMII interface. The method of popping is according to the selected Tx FIFO mode (Cut-Through mode or Store-and-Forward mode, the specific definition see the next paragraph). For convenient, application can configure automatically hardware calculated CRC and insert it to the FCS domain of Ethernet frame function. The entire transmission process complete when the MAC received the frame termination signal from transmit FIFO. When transmission completed, the transmission status information will be composed of MAC and write return to the DMA controller, the application can query through the DMA current transmit descriptor.

Operation for popping data from FIFO to the MAC has two modes:

- In Cut-Through mode, as soon as the number of bytes in the FIFO crosses or equals the configured threshold level or when the end-of-frame flag in descriptor is written, the data is ready to be popped out and forwarded to the MAC. The threshold level is configured using the TTCHC bits of ENET\_DMA\_CTL.
- In Store-and-Forward mode, only after an integrated frame is stored in the FIFO, the frame is popped towards the MAC. But there is another condition for FIFO popping out data but the frame is not integrated. This is when the transmit FIFO size is smaller than the Ethernet frame to be transmitted, the frame is popped towards the MAC when the transmit FIFO becomes almost full.

### Handle special cases

In the transmission process, due to the insufficient TxDMA descriptor or misuse of FTF bit in ENET\_DMA\_CTL register (when this bit is set, it will clear FIFO data and reset the FIFO pointer, after clear operation is completed, it will be reset), there will be a transmit data

underflow fault occurs because of insufficient data in FIFO. At the same time MAC will identify such data underflow state and write relevant status flag.

If one transmit frame uses two TxDMA descriptors for sending data, then the first segment (FSG) and the last segment (LSG) of the first descriptor should be 10b and the second ones should be 01b. If both the FSG bit of the first and the second descriptor are set and the LSG bit in the first descriptor is reset, then the FSB bit of the second descriptor will be ignored and these two descriptors are considered to sending the only one frame.

If the byte length of one transmission MAC frame's data field is less than 46 (for Tagged MAC frame is less than 42), application can configure the MAC for automatically adding a load of content of '0' bit to make the byte length of frame's data field in accordance with the relevant domain of definition of IEEE802.3 specification. At the same time, if automatically adding zeros function is performed, the MAC will certainly calculate CRC value of the frame and append it to the frame's FCS field domain no matter what configuration of DCRC bit in the descriptor is.

## Transmission management of MAC

### Jabber timer

In case of one station occupies the PHY for a long time, there is a jabber timer designed for cutting off the frame whose length is more than 2048 bytes. By default, jabber timer is enabled so when application is transmitting a frames whose byte length is more than 2048, the MAC will only transmit 2048 bytes and drop the last ones.

### Collision condition solve mechanism – Re-transmission

When the MAC is running under Half-duplex mode, collision may happen when MAC is transmitting frame data on interface. When no more than 96 bytes data popped from FIFO towards MAC and collision condition occurs, the re-transmission function is active. In this case, MAC will stop current transmitting and then read frame data from FIFO again and send them on interface again. When more than 96 bytes data popped from FIFO towards MAC and collision condition occurs, MAC will abort transmitting current frame data and not re-transmit it. Also MAC will set late collision flag in descriptor to inform application.

### Transmit status word

Transmit status word includes many transmit state flags for application and are updated after the complete the transmission of the frame. If timestamp function is enable, the timestamp value is also write back to transmit descriptor along with transmit status.

### Transmit FIFO flush operation

Application can clear TxFIFO and reset the FIFO data pointer by setting FTF bit (bit 20) of ENET\_DMA\_CTL register. The flush operation will be executed at once no matter whether TxFIFO is popping data to MAC. This results in an underflow event in the MAC transmitter, and the makes frame transmission abort. At the same time, MAC returns state information of

frame and transmit status words transferred to the application. The status of such a frame is marked with both underflow and frame flush events (TDES0 bits 1 and 13). When the transmit data in TxFIFO is flushed, the transmit status word will be written back to descriptor. After the status is written, the flush operation is complete. When a flush operation is received, all the following data which should be popped from TxFIFO into MAC will be dropped unless a new FSG bit of descriptor is received. After operation completed, the FTF bit of ENET\_DMA\_CTL register is then automatically cleared.

### Transmit flow control

The MAC manages transmission frame through back pressure (in Half-duplex mode) and the pause control frame (in Full-duplex mode) for flow control.

#### ■ Half-duplex mode flow control : Back Pressure

When MAC is configured in Half-duplex mode, there are two conditions to trigger the back pressure feature. Both of the two conditions are triggered to enable back pressure function which is implemented by sending a special pattern (called jam pattern) 0x5555 5555 once to notify conflict to all other sites. The first condition is triggered by application setting the FLCB/BKPA bit in ENET\_MAC\_FCTL register. The second condition occurs during receiving frame. When MAC receiver is receiving frame, the byte number of Rx FIFO is more and more great. When this number is greater than the high threshold (RFA bits in ENET\_MAC\_FCTH), MAC will set the back pressure pending flag. If this flag is set and a new frame presents on interface, MAC will send a jam pattern to delay receiving this new frame a back pressure time. After this back pressure time is end, external PHY will send this new frame again. If the number of the Rx FIFO is not less than low threshold (RFD bits in ENET\_MAC\_FCTH) during this back pressure time, a jam pattern is send again. If the number of the Rx FIFO is less than low threshold (RFD bits in ENET\_MAC\_FCTH) during this back pressure time, MAC resets the back pressure pending flag and is enable to receive the new frame instead of sending jam pattern.

#### ■ Full-duplex mode flow control : Pause Frame

The MAC uses a mechanism named "pause frame" for flow control in Full-duplex mode. Receiver can send a command to the sender for informing it to suspend transmission a period of time. If the application sets transmit flow control bit TFCEN in ENET\_MAC\_FCTL register, MAC will generate and transmit a pause frame when either of two conditions is satisfied in Full-duplex mode. There are two conditions to start transmit pause frames:

- 1) Application sets FLCB/BKPA bit in ENET\_MAC\_FCTL register to immediately send a pause frame. When doing this, MAC sends a pause frame right now with the pause time value PTM configured in ENET\_MAC\_FCTL register. If application considers the pause time is no need any more because the transmit frame can be transmitted without pause time, it can end the pause time by setting the pause time value PTM bits in ENET\_MAC\_FCTL register to 0 and set FLCB/BKPA bit to send this zero pause time frame.

- 2) MAC automatically sends pause time when the Rx FIFO is in some condition. When MAC is receiving frame, Rx FIFO will be fill in many receive data. At same time Rx FIFO pops out

data to RxDMA for forwarding to memory. If the popping frequency is lower than MAC pushing frequency, the number of bytes in RxFIFO is getting great. Once the data amount in RxFIFO is greater than the active threshold value (RFA bits in ENET\_MAC\_FCTH) of flow control, MAC will send a pause frame with PTM value in it. After sending pause frame, MAC will start a counter with configured reload value PLTS in ENET\_MAC\_FCTL register, when configured PLTS time has spent, the MAC will check RxFIFO again. If the byte number in RxFIFO is also greater than active threshold value, the MAC sends a pause time again. When the byte number of RxFIFO is lower than the de-active threshold value, MAC maybe send a pause frame with zero time value in frame's pause time field if DZQP bit in ENET\_MAC\_FCTL register is reset. This zero-pause time frame can inform send station that RxFIFO is almost empty and can receive new data again.

### Transmit inter-frame gap management

MAC can manage the interval time between two frames. This interval time is called frame gap time. For Full-duplex mode, after complete sending a frame or MAC entered idle state, the gap time counter starts counting. If another transmit frame presents before this counter has not reach the configured IGBS bit time in ENET\_MAC\_CFG register, this transmit frame will be pended unless the counter reach the gap time. But if the second transmit frame presents after the gap time counter has reached the configured gap time, this frame will send immediately. For Half-duplex mode, the gap time counter follows the Truncated Binary Exponential Backoff algorithm. Briefly speaking, the gap time counter starts after the previous frame has completed transmitting on interface or the MAC entered idle state, and there are three conditions may occur during the gap time:

- 1) The carrier sense signal active in the first 2/3 gap period. In this case, the counter will reload and restart.
- 2) The carrier sense signal active in the last 1/3 gap period. In this case, the counter will not reload but continue counting, and when reaches gap time, the MAC sends the second frame
- 3) The carrier sense signal not active during the whole gap period. In this case, the counter stops after reaches the configured gap time and sends frame if the second frame has pended.

### Transmit checksum offload

The MAC supports transmit checksum offload. This feature can calculate checksum and insert it in the transmit frame, and detect error in the receive frame. This section describes the operation of the transmit checksum offload.

**Note:** This function is enabled only when the TSFD bit in the ENET\_DMA\_CTL register is set (TxFIFO is configured to Store-and-Forward mode) and application must ensure the TxFIFO deep enough to store the whole transmit frame. If the depth of the TxFIFO is less than the frame length, the MAC only does calculation and insertion for IPv4 header checksum field.

See IETF specifications RFC 791, RFC 793, RFC 768, RFC 792, RFC 2460 and RFC 4443 for IPv4, TCP, UDP, ICMP, IPv6 and ICMPv6 packet header specifications, respectively.

**■ IP header checksum**

If the value is 0x0800 in type field of Ethernet frame and the value is 0x4 in the IP datagram's version field, checksum offload module marks the frame as IPv4 package and calculated value replace the checksum field in frame. Because of IPv6 frame header does not contain checksum field, the module will not change any value of the IPv6's header field. After IP header checksum calculation end, the result is stored in IPHE bit (bit 16 in TDES0). The following shows the conditions under which the IPHE bit can be set:

**1) For IPv4 frame type:**

- A). type field is 0x0800 but version filed in IP header is not 0x4.
- B). IPv4 header length field value is greater than total frame byte length
- C). the value of IPv4 header length field is less than 0x5 (20 bytes)

**2) For IPv6 frame type:**

- A). type field is 0x86dd but version field in IP header is not 0x6
- B). the frame ends before the IPv6 standard header or extension header (as given in the corresponding header length field in an extension header) has been completely received.

**■ TCP/UDP/ICMP payload checksum**

The checksum offload module processes the IPv4 or IPv6 header (including extension headers) and marks the type of frame (TCP, UDP or ICMP).

But when the following frame cases are detected, the checksum offload function will be bypassed and these frames will not be processed by the checksum offload module:

**1) Incomplete IPv4 or IPv6 frames****2) IP frames with security features (e.g. authentication header, security payload)****3) IP frames without TCP/UDP/ICMPv4/ICMPv6 payload****4) IPv6 frames with routing headers**

The checksum offload module calculates the TCP, UDP, or ICMP payload and inserts the result into its corresponding field in the header. It has two modes when working, as follows:

1) TCP, UDP, or ICMPv6 pseudo-header is not included in the checksum calculation and is assumed to be present in the input frame's checksum field. The checksum field is included in the checksum calculation, and then replaced by the final calculated checksum.

2) Checksum offload module clears the contents of the checksum field in the transmission frame and make calculation which includes TCP, UDP, or ICMPv6 pseudo-header data and will instead the transmission frame's original checksum field by the final calculation results.

After calculated by checksum offload module, the result can be found in IPPE bit (bit 12 in TDES0). The following shows the conditions under which the IPPE bit can be set:

1) In Store-and-Forward mode, frame has been forwarded to MAC transmitter but no EOF is written to TxFIFO

2) Frame is ended but the byte numbers which the payload length field of the frame indicates has not been reached

If the packet length is greater than the marked length, checksum module does not report errors, the excess data will be discarded as padding bytes. If the first condition of IPPE error is detected, the value of the checksum does not insert a TCP, UDP or ICMP header. If the second condition of IPPE error is detected, checksum calculation results will still insert the appropriate header fields.

**Note:** For ICMP packets over IPv4 frame, the checksum field in the ICMP packet must always be 0x0000 in both modes due to such packets are not defined pseudo-headers.

## MAC receive filters

The MAC filter is divided into error filtering (such as too short frame, CRC error and other bad frame filtering) and address filtering. This section mainly describes the address filtering.

### Address filtering

Address filtering use the static physical address (MAC address) filter and hash list filter for implementing the function. If the FAR bit in the ENET\_MAC\_FRMF register is '0' (by default), only the frame passed the filter will be received. This function is configured according to the parameters of the application (frame filter register) to filter the destination or/and source address of unicast or multicast frame (The difference between an individual address and a group address is determined by I/G bit in the destination address field) and report the result of the corresponding address filtering. The frame not pass through the filter will be discarded.

**Note:** If the FAR bit in the ENET\_MAC\_FRMF register is set to 1, frames are all thought passed the filter. In this case, even the filter result will also be updated in receive descriptor but the result will not affect whether current frame passes the filter or not.

#### Unicast frame destination address filter

For a unicast frame, application has two modes for filtering: the one is using static physical address (by setting HUF bit to '0'), the other is using hash list (by setting HUF bit to '1').

- Static physical address (SPA) filtering

In the filter mode, MAC supports using four MAC addresses for unicast frame filtering. In this way, the MAC compares all 6 bytes of the received unicast address to the programmed MAC address. MAC address 0 is always used and MAC address 1 to address 3 can be configured to use or not. Each byte of MAC address 1 to MAC address 3 register can be masked for comparison with the corresponding destination address byte of received frame by setting the corresponding mask byte bits (MB) in the corresponding register.

- Hash list filtering

In this filter mode, MAC uses a HASH mechanism. MAC uses a 64-bit hash list to filter the received unicast frame. This filter mode obeys the followings two filtering steps:

- 1) The MAC calculates the CRC value of the received frame's destination address
- 2) Using the high 6 bits of the calculated CRC value as the index to retrieve the hash list. If the corresponding value of hash list is 1, the received frame passes through the filter, conversely, fail the filter.

The advantage of this type of filter is that it can cover any possible address just using a small table. But the disadvantage is that the filter is imperfect and sometimes the frames should be dropped are also be received by mistake.

#### **Multicast frame destination address filter**

Application can enable the multicast frame MAC address filtering by cleaning the MFD bit in register ENET\_MAC\_FRMF. By configuring the value of HMF bit in ENET\_MAC\_FRMF register application can choose two ways just like unicast destination address filtering for address filtering.

#### **Hash or perfect address filter**

The destination address (DA) filter can be configured to pass a frame when its DA matches either the hash list filter or the static physical address filter by setting the HPFLT bit in the ENET\_MAC\_FRMF register and setting the corresponding HUF or HMF bit in the ENET\_MAC\_FRMF register.

#### **Broadcast frame destination address filter**

At default, the MAC unconditionally receives the broadcast frames. But when setting BFRMD bit in register ENET\_MAC\_FRMF, MAC discards all received broadcast frames.

#### **Unicast frame source address filter**

Enable MAC address 1 to MAC address 3 register and set the corresponding SAF bit in the MAC address high register, the MAC compares and filter the source address (SA) field in the received frame with the values programmed in the SA registers. MAC also supports the group filter on the source address. If the SAFLT bit in frame filter register ENET\_MAC\_FRMF is set, MAC drops the frame that failed the source address filtering; meanwhile the filter result will reflect by SAFF bit in RDES0 of DMA receive descriptor. When the SAFLT bit is set, the destination address filter is also at work, so the result of the filter is simultaneous determined by DA and SA filter. This means that, as long as a frame does not pass any one of the filters (DA filter or SA filter), it will be discarded. Only a frame passing the entire filter can be forwarded to the application.

#### **Reverse filtering operation**

MAC can reverse filter-match result at the final output whether the destination address filtering or source address filtering. By setting the DAIFLT and SAIFLT bits in ENET\_MAC\_FRMF register, this address filter reverse function can be enabled. DAIFLT bit is used for unicast and multicast frames' DA filtering result, SAIFLT bit is used for unicast and multicast frames

SA filtering result.

The following two tables summarize the destination address and source address filters working condition at different configuration.

**Table 23-4. Destination address filtering table**

Frame Type	PM	HPFL T	HUF	DAIFLT	HM F	MFD	BFRM D	DA filter operation
Broadcast	1	-	-	-	-	-	-	Pass
	0	-	-	-	-	-	0	Pass
	0	-	-	-	-	-	1	Fail
Unicast	1	-	-	-	-	-	-	Pass all frames
	0	-	0	0	-	-	-	Pass on perfect/group filter match
	0	-	0	1	-	-	-	Fail on perfect/group filter match
	0	0	1	0	-	-	-	Pass on hash filter match
	0	0	1	1	-	-	-	Fail on hash filter match
	0	1	1	0	-	-	-	Pass on hash or perfect/group filter match
	0	1	1	1	-	-	-	Fail on hash or perfect/group filter match
Multicast	1	-	-	-	-	-	-	Pass all frames
	-	-	-	-	-	1	-	Pass all frames
	0	-	-	0	0	0	-	Pass on perfect/group filter match and drop PAUSE control frames if PCFRM = 0x
	0	0	-	0	1	0	-	Pass on hash filter match and drop PAUSE control frames if PCFRM = 0x
	0	1	-	0	1	0	-	Pass on hash or perfect/group filter match and drop PAUSE control frames if PCFRM = 0x
	0	-	-	1	0	0	-	Fail on perfect/group filter match and drop PAUSE control frames if PCFRM = 0x
	0	0	-	1	1	0	-	Fail on hash filter match and drop PAUSE control frames if PCFRM = 0x
	0	1	-	1	1	0	-	Fail on hash or perfect/group filter match and drop PAUSE control frames if PCFRM = 0x

**Table 23-5. Source address filtering table**

Frame type	PM	SAIFLT	SAFLT	SA filter operation
Unicast	1	-	-	Pass all frames
	0	0	0	Pass status on perfect/group filter match but do not drop frames that fail

Frame type	PM	SAIFLT	SAFLT	SA filter operation
Unicast	0	1	0	Fail status on perfect/group filter match but do not drop frame
	0	0	1	Pass on perfect/group filter match and drop frames that fail
	0	1	1	Fail on perfect/group filter match and drop frames that fail

### Promiscuous mode

If the PM bit in ENET\_MAC\_FRMF register is set, promiscuous mode is enable. Then the address filter function is bypassed, all frames are thought passed through the filter. At the same time the receive status information DA / SA error bit is always '0'.

### Pause control frame filter

When MAC received pause frame, it will detect 6 bytes DA field in the frame. If UPFDT bit in ENET\_MAC\_FCTL register is 0, it is determined by whether the value of the DA field conforms to the unique value (0x0180C2000001) with IEEE-802.3 specification control frames. If UPFDT bit in ENET\_MAC\_FCTL register is set, MAC additionally compares DA field with the programmed MAC address for bit match. If DA field match and receive flow control is enabled (RFCEN bit in ENET\_MAC\_FCTL register is set), the corresponding pause control frame function will be triggered. Whether this filter passed pause frame is forwarded to memory is depending on the PCFRM[1:0] bit in ENET\_MAC\_FRMF register.

### Reception process of MAC

Received frames will be pushed to the Rx FIFO. The MAC strips the preamble and SFD of the frame, and starts pushing the frame data beginning with the first byte following the SFD to the Rx FIFO. If IEEE 1588 time stamp function is enabled, the MAC will record the current system time when a frame's SFD is detected. If the frame passes the address filter, this time stamp is passed on to the application by writing it to descriptor.

The MAC can automatically strip PAD and FCS field data when the length/type field of received frame is less than 1536 if APCD bit is set. MAC pushes the data of the frame into Rx FIFO up to the count specified in the length/type field, then starts dropping bytes (including the FCS field). If the value of length/type field is greater than or equal to 0x600, the automatically strip FCS field function is configured by the TFCD bit regardless of APCD.

If the watchdog timer is enabled (WDD bit in ENET\_MAC\_CFG is reset), a frame has more than 2048 bytes (DA + SA + LT + DATA + FCS) will be cut off receiving when has received 2048 bytes. If the watchdog timer is disabled, the MAC can extend the max receiving data bytes to 16384(16K Bytes), any data beyond this number will be cut off.

When Rx FIFO works at Cut-Through mode, it starts popping out data from Rx FIFO when the number of FIFO is greater than threshold value (RTHC bits in ENET\_DMA\_CTL register). After all data of a frame pop out, receive status word is sent to DMA for writing back to

descriptor. In this mode, if a frame has started to forward to application by DMA from FIFO, the forwarding will continue until the frame is end even if frame error is detected. Although the error frame is not discarded, the error status will reflect in descriptor status field.

When RxFIFO works at Store-and-Forward mode (set by RSFD bit in ENET\_DMA\_CTL), DMA reads frame data from RxFIFO only after RxFIFO has completed received the whole frame. In this mode, if the MAC is configured to discard all error frames, then only valid frames without any error can be read out from RxFIFO and forward to the application. Once the MAC detects an SFD signal on the interface, a receive operation is started. The MAC strips the preamble and SFD before processing the frame. The header fields are checked by filtering and the FCS field used to verify the CRC for the frame. The frame is discarded by MAC if it fails to pass the address filter.

## Reception management of MAC

### Receive operation on multi-frame handling

It is different from transmit operation, after receiving the last byte of a frame, the MAC can judge the status of the receiving operation, so the second received frame's forwarding is surely followed by the first received frame data and status.

### Receive flow control

In Full-duplex mode, the MAC can detect the pause control frames, and perform it by suspending a certain time which is indicated in pause time field of detected pause control frame and then to transmit data. This function can set by RFCEN bit in ENET\_MAC\_FCTL register. If this function is not enabled, the MAC will ignore the received pause frames. If this function is enabled, MAC can decode this frame. Type field, opcode field and pause time field in the frame are all recognized by the MAC. During the pause time period, if MAC received a new pause frame, the new pause time filed value is loaded to the pause time counter immediately. If the new pause time filed is zero, then the pause time counter stops and transmit operation recovers. Application can configure PCFRM bit in ENET\_MAC\_FRMF register to decide the solving method for such control frame.

### Receive checksum offload

Receive checksum offload is enabled when IPFCO bit in ENET\_MAC\_CFG register is set. Receive checksum offload can calculate the IPv4 header checksum and check whether it matches the contents of the IPv4 header checksum field. The MAC identifies IPv4 or IPv6 frames by checking for the value of 0x0800 or 0x86DD respectively in the received Ethernet frame type field. This method is also used to identify frames with VLAN tags. Header checksum error bits in DMA receive descriptor (the 7 bit in RDES0) reflects the header checksum result. This bit is set if received IP header has the following errors:

- Any mismatch between the IPv4 calculation result by checksum offload module and the value in received frame's checksum field.

- Any inconsistent between the data type of Ethernet type field and IP header version field.
- Received frame length is less than the length indicated in IPv4 header length field, or IPv4 or IPv6 header is less than 20 bytes.

Receive checksum offload also identifies the data type of the IP packet is TCP, UDP or ICMP, and calculate their checksum according to TCP, UDP or ICMP specification. Calculation process can include data of TCP/UDP/ICMPv6 pseudo-header. Payload checksum error bits in DMA receive descriptor (bit 0 in RDES0) reflects the payload checksum result. This bit is set if received IP payload has the following errors:

- Any mismatch between the TCP, UDP or ICMP checksum calculation result by checksum offload and the received TCP/UDP/ICMP frame's checksum field.
- Any inconsistent between the received TCP, UDP or ICMP data length and length of IP header.

The received checksum offload does not calculate the following conditions: Incomplete IP packets, IP packets with security features, packets of IPv6 routing header and data type is not TCP, UDP or ICMP.

### **Error handling**

- If RxFIFO becomes full but the last received byte is not the end of frame (EOF), the RxFIFO will discard the whole frame data and return an overflow status. Also the counter of counting the overflow condition times will plus 1.
- If the RxFIFO is configured in Store-and-Forward mode, the MAC can filter and discard all error frames. But according to the configuration of FERF and FUF bit in ENET\_DMA\_CTL register, RxFIFO can also receive and forward such error frame and the frame that length is less than the minimum length.
- If the RxFIFO is configured in Cut-Through mode, not all the error frames can be dropped. Only when the start of frame (SOF) has not been read from RxFIFO and the receive frame has been detected error status, the RxFIFO will discard the whole error frame.

### **Receive status word**

After receiving a complete frame, the MAC will analysis and record some state information about the frame and receiving process. These detail status information will write back to the receive descriptor and DMA status flag. Application can check these flags for upper protocol implementation.

**Note:** The value of frame length is 0 means that for some reason (such as FIFO overflow or dynamically modify the filter value in the receiving process, resulting did not pass the filter, etc), frame data is not written to FIFO completely.

### **MAC loopback mode**

Often, loopback mode is used for testing and debugging hardware and software system for

application. The MAC loopback mode is enabled by setting the LBM bit in ENET\_MAC\_CFG register. In this mode, the MAC transmitter sends the Ethernet frame to its own receiver. This mode is disabled by default.

### 23.3.3. MAC statistics counters: MSC

For knowing the statistics situation of transmitting and receiving frames, there is a group of counters designed for gathering statistics data. These MAC counters are called statistics counters (MSC). In Section ‘Register Description’, there is a detailed description of the function of these registers.

When the transmit frame does not appear the following situation, it can be called ‘fine frame’ and MSC transmit counters will automatically update:

- Frame underflow
- No carrier
- Lose of carrier
- Excessive deferral
- Late collision
- Excessive collision
- Jabber Timeout

When the receiving frame does not appear the following situation, it can be called ‘fine frame’ and MSC reception counters will automatically update:

- Alignment error
- CRC mismatch(calculated CRC value is different from FSC field value)
- Runt frame (frame length is shorter than 64 bytes)
- Length error (length field value is different from the actual received data bytes)
- Range error (length field value is larger than maximum size of defined in IEEE802.3, which is 1518 for untagged frame and 1522 for VLAN tagged frame)
- Error signal valid on pin MII\_RX\_ER

**Note:** Only when the discarded frame is a short frame whose length is less than 6 bytes (no complete receives the DA), MSC reception counter is updated.

### 23.3.4. Wake up management: WUM

Ethernet (ENET) module supports two wakeup methods from Deep-sleep mode. The one is remote wakeup frame and the other is Magic Packet wakeup frame. For reduce power consuming, the host system and Ethernet can be powered down and thus the circuit driven by HCLK or transmit clock is stop working. But the circuit driven by receive clock will continues working for listening wakeup frame. If application sets the PWD bit in ENET\_MAC\_WUM register, the Ethernet enters into power-down state. In power-down state, MAC ignores all the frame data on the interface until the power-down state is exited. For exiting power-down state, application can choose one of or both of the two methods mentioned above. Setting WFEN

bit in ENET\_MAC\_WUM register can make Ethernet wakeup if a remote wakeup frame received and setting MPE bit in ENET\_MAC\_WUM register can make Ethernet wakeup if a Magic Packet frame is received. When any type of wakeup frame is present on interface and corresponding wakeup function is enabled, Ethernet will generate a wakeup interrupt and exit power-down state at once.

### **Remote wakeup frame detection**

Setting WFEN bit in ENET\_MAC\_WUM register can enable remote wakeup detection. When the MAC is in power-down state and remote wakeup function enable bit is set, MAC wakeup frame filter is active. If the received frame passes the address filter and filter CRC-16 matches the incoming examined pattern, then MAC identified the received wakeup frame, and then MAC returns to normal working state. Even if the length of the wakeup frame exceeds 512 bytes, as long as the frame has a correct CRC value, it is still considered to be effective. After received the remote wakeup frame, the WUFR bit in ENET\_MAC\_WUM register will be set. If remote wakeup interrupt is not masked, then a WUM interrupt is generated.

### **Remote wakeup frame filter register**

Wakeup frame filter register is made up of eight different registers but shared the same register offset address. So the inner pointer points the next filter register when the filter register address is accessed by writing or reading. Whatever operation, write or read, it is strongly recommended to operate eight times sequentially. This means continuously write 8 times will configure the filter registers and continuously read 8 times will get the values of filter registers.

**Figure 23-6. Wakeup frame filter register**

Wakeup Frame Filter Register 0	Filter 0 Byte Mask											
Wakeup Frame Filter Register 1	Filter 1 Byte Mask											
Wakeup Frame Filter Register 2	Filter 2 Byte Mask											
Wakeup Frame Filter Register 3	Filter 3 Byte Mask											
Wakeup Frame Filter Register 4	Reserve	Filter 3 Command	Reserve	Filter 2 Command	Reserve	Filter 1 Command	Reserve	Filter 0 Command				
Wakeup Frame Filter Register 5	Filter 3 Offset		Filter 2 Offset		Filter 1 Offset		Filter 0 Offset					
Wakeup Frame Filter Register 6	Filter 1 CRC - 16				Filter 0 CRC - 16							
Wakeup Frame Filter Register 7	Filter 3 CRC - 16				Filter 2 CRC - 16							

- Filter n Byte mask

This register field defines using which bytes of the frame to determine the received frame is wakeup frame or not by filter n (n=0, 1, 2, 3). Bit 31 must be set to 0. Bit 30 to bit 0 are valid byte mask. If bit m(m means byte number) is set, the filter n offset + m of the receiving frame

is calculated by the CRC unit, conversely, filter n offset + m is ignored.

- Filter n command

This four bits command controls the operation of the filter n. The bit 3 of the field is address type selection bit. If this bit is 1, the detection only detects a multicast frame and if this bit is 0, the detection only detects a unicast frame. Bit 2 and bit 1 must be set to 0. Bit 0 is the filter switch bit. Setting it to 1 means enable and 0 means disable.

- Filter n offset

It is used in conjunction with filter n byte mask field. This register specifies offset (within the frame) of the first byte which the filter n uses to check. The minimum allowable value is 12, it represents the byte 13 in the frame (offset value 0 indicates the first byte of the frame).

- Filter n CRC-16

This register field contains the filter comparing CRC-16 code which is used for comparing the calculated CRC-16 from frame data.

## **Magic packet detection**

Another wakeup method is detecting Magic Packet frame (see ‘Magic Packet Technology’, Advanced Micro Devices). A Magic Packet frame is a special frame with formed packet solely intended for wakeup purposes. This packet can be received, analyzed and recognized by the Ethernet block and used to trigger a wakeup event. Setting MPE bit in ENET\_MAC\_WUM register can enable this function. This type of frame’s format is as follows: starts by 6 continuous bytes of the value 0xFF (0xFFFF FFFF FFFF) in anywhere of the frame behind the destination and source address field, then there are 16 duplicate MAC addresses without any interruption and pause. If there is any discontinuity between repeating it 16 times, MAC needs to re-detect 0xFFFF FFFF FFFF in the receive frame. WUM module continuously monitors each frame received. When a Magic Packet frame passing the address filter, MAC will detect its format with Magic Packet format, once the format is matched the WUM will make MAC wakeup from power down state. Then the MAC wakes up from power-down state after receiving a Magic Packet frame. Module also accepts multicast frames as Magic Packet frame.

Example: An example of a Magic Packet with station address 0xAABB CCDD EEFF is the following (MISC indicates miscellaneous additional data bytes in the packet):

<DESTINATION><SOURCE><MISC>

.....FF FF FF FF FF FF

AABB CCDD EEFF AABB CCDD EEFF AABB CCDD EEFF AABB CCDD EEFF

AABB CCDD EEFF AABB CCDD EEFF AABB CCDD EEFF AABB CCDD EEFF

AABB CCDD EEFF AABB CCDD EEFF AABB CCDD EEFF AABB CCDD EEFF

AABB CCDD EEFF AABB CCDD EEFF AABB CCDD EEFF AABB CCDD EEFF

## &lt;MISC&gt;&lt;FCS&gt;

Upon detecting a Magic Packet, the MPKR bit in ENET\_MAC\_WUM register will be set. If the Magic Packet interrupt is enabled, the corresponding interrupt will generate.

### Precautions during system power-down state

When the MCU is in Deep-sleep mode, if external interrupt line 19 is enabled, Ethernet WUM module can still detecting frames. Because the MAC in power-down state needs detecting Magic Packet or remote wakeup frame, the REN bit in ENET\_MAC\_CFG register must be maintained set. The transmit function should be turned disable during the power-down state by clearing the TEN bit in the ENET\_MAC\_CFG register. Moreover, the Ethernet DMA block should be disabled during the power-down state, because it is not necessary that the Magic Packet or remote wakeup frame is forwarded to the application. Application can disable the Ethernet DMA block by clearing the STE bit and the SRE bit (for the TxDMA and the RxDMA, respectively) in the ENET\_DMA\_CTL register.

Follow steps are recommended for application to enter and exit power-down state:

1. Wait the current sending frame completes and then reset the TxDMA block by clearing STE bit in ENET\_DMA\_CTL register.
2. Clear the TEN and REN bit in ENET\_MAC\_CFG register to disable the MAC's transmit and receive function.
3. Check the RS bit in ENET\_DMA\_STAT register, waiting receive DMA read out all the frames in the receive FIFO and then close RxDMA.
4. Configure and enable the external interrupt line 19, so that it can generate an interrupt or event. If EXTI line 19 is configured to generate an interrupt, application still needs to modify ENET\_WKUP\_IRQ interrupt handling procedures to clear the pending bit of the EXTI line 19.
5. Set the MPEN or WFEN (or both) bit in ENET\_MAC\_WUM register to enable Magic Packet or Remote Wakeup frame(or both) detection.
6. Setting PWD bit in ENET\_MAC\_WUM register to enter power-down state.
7. Setting REN bit in ENET\_MAC\_CFG register to make MAC's receive function work.
8. Make MCU enter Deep-sleep mode.
9. After received a wakeup type frame, the Ethernet module exits the power-down state.
10. Reading the ENET\_MAC\_WUM register to clear the power management event flags. Enable MAC's transmit function and enable TxDMA and RxDMA.
11. Initialize the MCU system clock: enable HXTAL and configure the RCU unit.

### 23.3.5. Precision time protocol: PTP

The majority of protocols are implemented by the UDP layer application software. The PTP module of the MAC is mainly to recording the transmitting and receiving PTP packets' precision time and returning it to application.

Specific details about the precise time protocol (PTP) please see the document "IEEE Standard 1588™".

#### Reference clock source

System reference time in Ethernet is maintained by a 64-bit register whose high 32-bit indicates 'second' time and low 32-bit indicates 'subsecond', this is defined in IEEE 1588 specification.

The input PTP reference clock is used to drive the system reference time (also called system time for short) and capture timestamp value for PTP frame. The frequency of this reference clock must be configured no less than the resolution of timestamp counter. The synchronization accuracy between the master node and slave node is around 0.1us.

#### Synchronization accuracy

The accuracy of time synchronization depends on the following factors:

- 1) PTP reference clock input period
- 2) Characteristics of the oscillator (drift)
- 3) Frequency of the synchronization procedure.

#### System time correction method

The 64-bit PTP system time update by the PTP input reference clock. The PTP system time is used as the source to record transmission/reception frame's timestamp. The system time initialization and calibration support two methods: coarse method and fine method. The purpose of calibration is to correct the frequency offset.

If the coarse correction method is selected, application can configure PTP timestamp update register (ENET\_PTP\_TSUH and ENET\_PTP\_TSUL) for system time initialization or correction. If TMSSTI bit is set, PTP timestamp update register is used for initialization and if TMSSTU bit is set, PTP timestamp update register is used for adjust system time by adding or subtracting.

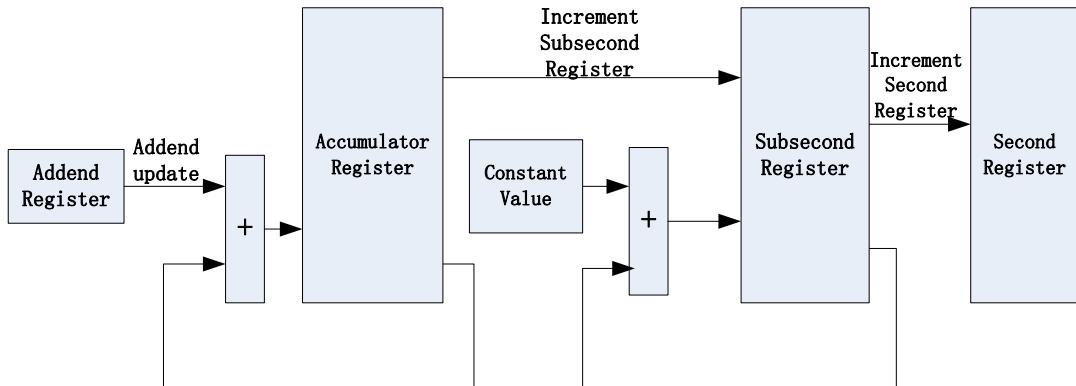
If fine correction method is selected, operation is different. The fine correction method corrects system time not in a single clock cycle. The fine correction frequency can be configured by application to make slave clock frequency smoothly adapt master clock without unpredictability large jitter.

This method is referred to the value of ENET\_PTP\_TSADDEND added to the accumulator in each HCLK cycle. PTP module will produce a pulse to increase the value of ENET\_PTP\_TSL

register when the accumulator overflowed. The increased value when this pulse occurs is in ENET\_PTP\_SSINC register.

**Figure 23-7. System time update using the fine correction method** shows the fine correction algorithm process:

**Figure 23-7. System time update using the fine correction method**



The following concrete example is used to describe the fine correction method how to update the system time:

Assuming the accuracy of the system time update circuit required to achieve 20ns, which means the frequency of update is 50MHz. If the reference clock of HCLK is 75MHz, the frequency ratio is calculated as 75/50, result is 1.5. Hence, the addend (TMSA bit in ENET\_PTP\_TSADDEND register) value to be set is  $2^{32}/1.5$ , which is equal to 0xAAAA AAAA. If the reference clock frequency drifts lower, for example, down to 65MHz, the frequency ratio changes to  $65/50=1.3$ , the value to be set in the addend register is  $2^{32}/1.30 = 0xC4EC 4EC4$ . If the reference clock drift higher, for example, up to 85MHz, the value addend register must be 0xA000 0000. Initially, the slave clock frequency is set to Clock Addend Value (0) in the addend register. This value is calculated as above. In addition to configuring the addend counter, application also needs to set subsecond increment register to ensure to achieve the precision of 20ns. The value of the register is to update values of timestamp low 32-bit register after accumulator register overflow. Because the timestamp low register (bit 0 to 30) represents the subsecond value of system time, the precision is  $10^9\text{ns}/2^{31}=0.46\text{ns}$ . So in order to make the system time accuracy to 20ns, sub second increment register value should be set to  $20/0.46 = 0d43$ .

**Note:** The algorithm described below based on constant delay transferred between master and slave devices (Master-to-Slave-Delay). Synchronous frequency ratio will be confirmed by the algorithm after a few Sync cycles.

Algorithm is as follows:

- Define the master sends a SYNC message to slave time: MSYNCT (n).
- Define the slave local time SLOCALT (n).
- Define the master local time MLOCALT (n).

Calculation:  $MLOCALT(n) = MSYNCT(n) + \text{Master-to-Slave-Delay}(n)$

- Define the master clock count number between two SYNC message sent:  
 $MCLOCKC(n)$   
 Calculation:  $MCLOCKC(n) = MLOCALT(n) - MLOCALT(n-1)$   
 Define the slave clock count number between two received SYNC messages:  $SCLOCKC(n)$   
 Calculation:  $SCLOCKC(n) = SLOCALT(n) - SLOCALT(n-1)$
- Define the difference between these two count numbers:  $DIFFCC(n)$   
 Calculation:  $DIFFCC(n) = MCLOCKC(n) - SCLOCKC(n)$
- Define the slave clock frequency-adjusting factor:  $SCFAF(n)$   
 Calculation:  $SCFAF(n) = (MCLOCKC(n) + DIFFCC(n)) / SCLOCKC(n)$
- Define the Clock Addend Value for addend register: Clock Addend Value(n)  
 $\text{Clock Addend Value}(n) = SCFAF(n) * \text{Clock Addend Value}(n-1)$

**Note:** During the actual operation, application may need more than once SYNC message between master and slave to lock.

### System time initialization procedure

Setting TMSEN bit in ENET\_PTP\_TSCTL register to 1, timestamp function is enabled. Each time after this bit is set from reset, application must initialize the timestamp counter at first. Initialization steps as follow:

1. Setting bit 9 in the ENET\_MAC\_INTMSK register to mask the timestamp trigger interrupt
2. Setting bit 0 in the ENET\_PTP\_TSCTL register to enable timestamp function
3. Configure the subsecond increment register according to the PTP clock frequency precision
4. If application hopes to use fine correction method, configure the timestamp addend register and set bit 5 in the ENET\_PTP\_TSCTL register to 1. If application hopes to use coarse correction method, please jump directly to step 7 and step 4-6 can be ignored.
5. Poll the bit 5 in the ENET\_PTP\_TSCTL register until it is cleared
6. Set bit 1 in the ENET\_PTP\_TSCTL register to 1 to choose fine correction method
7. Configure the timestamp update high and low register with the value of system time application wants to initialize
8. Send initialization command by setting bit 2 in the ENET\_PTP\_TSCTL register
9. The timestamp counter starts counting as soon as the initialization process complete

### System time update steps under coarse correction method

1. Program the offset (may be negative) value in the timestamp update high and low registers

2. Set bit 3 (TMSSTU) in the ENET\_PTP\_TSCTL register to update the timestamp register
3. Poll TMSSTU bit until it is cleared.

### **System time update steps under fine correction method**

1. Calculate the value of the desired system clock rate corresponding to the addend register (calculation formula has explained before)
2. Program the addend register, and set the bits 5 in ENET\_PTP\_TSCTL register
3. Program the target high and low register and reset the bit 9 of the ENET\_MAC\_INTMSK register to allow time stamp interrupt
4. Set bit 4 (TMSITEN) in ENET\_PTP\_TSCTL register
5. When an interrupt is generated by this event, read out the value of ENET\_MAC\_INTF register and clear the corresponding interrupt flag
6. Rewrite the old value of addend register to timestamp addend register and set bit 5 in ENET\_PTP\_TSCTL register

### **Transmission and reception of frames with the PTP feature**

After enabled the IEEE 1588 (PTP) timestamp function, timestamp is recorded when the frame's SFD field is outputting from the MAC or the MAC receives a frame's SFD field. Each transmitted frame can be marked in TxDMA descriptor to indicate whether a timestamp should be captured or not, which is unrelated with whether the transmitted frame has PTP feature or not, and the timestamp of all received frames will be recorded if ARFSEN bit in ENET\_PTP\_TSCTL register is set. If ARFSEN is reset, the received frame which passed the address filer should be matched with the configuration in ENET\_PTP\_TSCTL register. In another word, only the frame matched the PTP configuration is marked a PTP frame, and timestamp will be recorded in descriptor. To be marked as a PTP frame, the received frame PTP version should be coincide with PFSV bit and then the corresponding frame type enable bit(bit 13 to bit 11 in register ENET\_PTP\_TSCTL) is set. Specially, the non-IP payload PTP frame (PTP on normal 802.3 Ethernet frame), also the DA should be the special MAC address (e.g. the DA should be 0x0e00 00c2 8001 for PDELAY\_REQ/ PDELAY\_RESP/ PDELAY\_RESP\_FOLLOW\_UP message type, and the DA address 0x0000 0019 1B01 for other message type, detailed information refer to Specification IEEE1588-2008). If MAFEN is set, this special MAC address can be extended to MAC address1-3 with SAF is reset.

Together with the state information of frame, the recorded timestamp value will also be stored in the corresponding transmission/reception descriptor. The 64-bit timestamp information of transmission frame is written back to the transmit descriptor and the 64-bit timestamp information of reception frame is written back to the receive descriptor. See the detailed description in "Transmit DMA descriptor" and "Receive DMA descriptor"

### **PTP trigger internal connection with TIMER1**

MAC can provide trigger interrupt when the system time is no less than the target time. Using an interrupt imports a known latency and an uncertainty in the command execution time. In

order to calculate the time of this known latency part, when the system time is greater than target time, the PTP module sets an output signal. Set bit 29 of AFIO\_PCF0 register to 0 can make this signal internally connected to the ITI1 input of TIMER1. For this feature designed, no uncertainty is introduced because the clock of the TIMER1 and PTP reference clock (HCLK) are synchronous.

### PTP pulse-per-second (PPS) output signal

Application set bit 30 of AFIO\_PCF0 register to 1 to enable the PPS output function. This function can output a signal with the pulse width of 125ms by default (other width is detailed in register definition) which can be used to check the synchronization between all nodes in the network. To test the difference between the slave clock and the master clock, both of the slave and master can output PPS and connect them to one oscilloscope for clock measurement.

#### 23.3.6. DMA controller description

Ethernet DMA controller is designed for frame transmission between FIFO and system memory which can reduce the occupation of CPU. Communication between the CPU and the DMA is achieved by the following two kinds of data structures:

- Descriptor table (ring or chain type) and data buffer
- Control and status register

Applications need to provide the memory for storage of descriptor tables and data buffers. Descriptors that reside in the memory act as pointers to these buffers. Transmission has transmission descriptor and reception has reception descriptor. The base address of each table is stored in ENET\_DMA\_TDTADDR and ENET\_DMA\_RDTADDR register. Descriptors of transmission constituted by 4 descriptor word (TDES0-TDES3) when DFM=0 and 8 descriptor word (TDES0-TDES7) when DFM=1 (Enhanced descriptor mode). Likewise, reception descriptors constituted by 4 descriptor word (RDES0-RDES3) when DFM=0 and 8 descriptor word (RDES0-RDES7) when DFM=1. Each descriptor can point to a maximum of two buffers. The value of the buffer 2 can be programmed to the second data address or the next descriptor address which is determined by the configured descriptor table type: Ring or Chain. Buffer space only contains frame data which are located in host's physical memory space. One buffer can store only one frame data but one frame data can be stored in more than one buffer which means one buffer can only store a part of a frame. When chain structure is set, descriptor table is an explicitly one and when ring structure is set, descriptor table is an implicitly one. Explicit chaining of descriptors is accomplished by configuring the second address chained in both receive and transmit descriptors (RDES1[14] and TDES0[20]), at this time RDES2 and TDES2 are stored the data buffer address, RDES3 and TDES3 should be stored the next descriptor address, this connection method of descriptor table is called chain structure. Implicitly chaining of descriptors is accomplished by clearing the RDES1[14] and TDES0 [20], at this time RDES2, TDES2 and RDES3, TDES3 should be all stored the data buffer address, this connection method of descriptor table is called ring structure. When

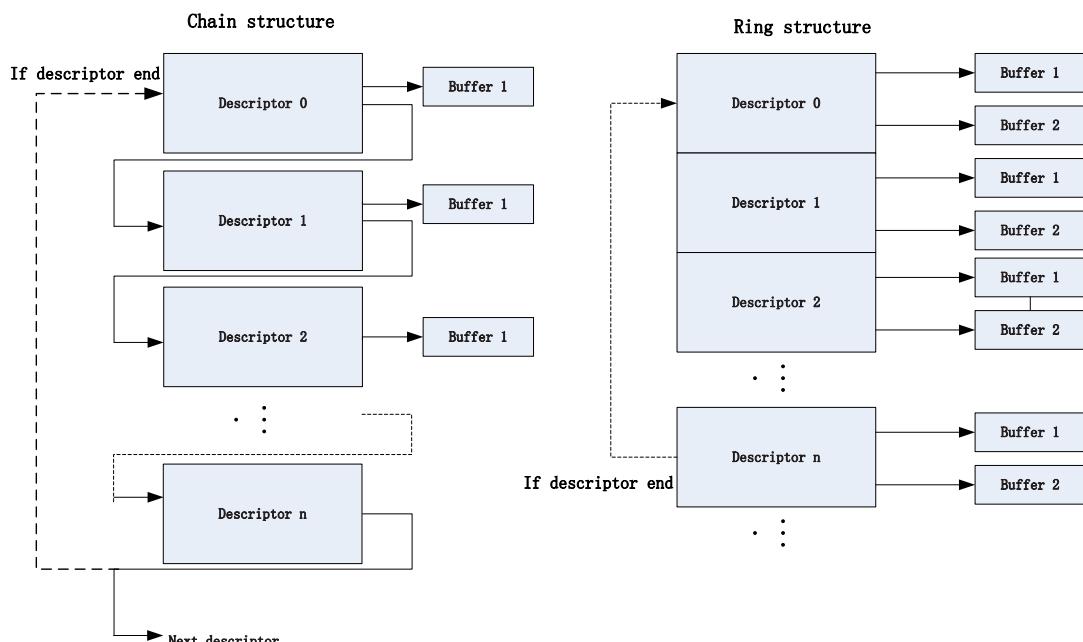
current descriptor's buffer address is used, descriptor pointer will point to the next descriptor. If chain structure is selected, the pointer points to the value of buffer 2. If ring structure is selected, the pointer points to an address calculated as below:

DFM=0: Next descriptor address = Current descriptor address + 16 + DPSL\*4

DFM=1: Next descriptor address = Current descriptor address + 32 + DPSL\*4

If current descriptor is the last one in descriptor table, application needs to set the bit 21 in TDES0 or bit 15 in RDES1 to inform DMA the current descriptor is the last one of the table in ring structure. At this time, the next descriptor pointer points back to the first descriptor address of the descriptor table. In chain structure, can also set TDES3 or RDES3 value to point back to the first descriptor address of the descriptor table. The DMA skips to the next frame buffer when the end of frame is detected.

**Figure 23-8. Descriptor ring and chain structure**



### Alignment rule for data buffer address

The DMA controller supports all alignment types: byte alignment, half-word alignment and word alignment. This means application can configure the buffer address to any address. But during the operation of the DMA controller, access address is always word align and is different between write and read access. Follow example describes the detail:

**Buffer Reading:** Assuming the transmit buffer address is 0x2000 0AB2, and 15 bytes need to be transferred. After starting operating, the DMA controller will read five word addresses which are 0x2000 0AB0, 0x2000 0AB4, 0x2000 0AB8, 0x2000 0ABC and 0x2000 0AC0. But when sending data to the FIFO, the first two bytes (0x2000 0AB0 and 0x2000 0AB1) and the last 3 bytes (0x2000 0AC1, 0x2000 0AC2 and 0x2000 0AC3) will be dropped.

Buffer Writing: Assuming the receive buffer address is 0x2000 0CD2, and 16 bytes need to be stored. After starting operating, the DMA controller will write five times 32-bit data from address 0x2000 0CD0 to 0x2000 0CE0. But the first 2 bytes (0x2000 0CD0 and 0x2000 0CD1) and the last 2 bytes (0x2000 0CE2 and 0x2000 0CE3) will be substituted by the virtual bytes.

**Note:** DMA controller will not write any data out of the defined buffer range.

### The effective length of the buffer

For the frame transmitting process, the effective length of the buffer is the same as the value configured by application in TDES1. As mentioned before, a transmitting frame can use one or more descriptors to indicate the frame information which means a frame data can be located in many buffers. When the DMA controller reads a descriptor which the FSG bit in TDES0 is set, it knows the current buffer is pointing to a new frame and the first byte of the frame is included. When the DMA controller reads a descriptor with FSG bit and LSG bit in TDES0 are both reset, it knows the current buffer is pointing to a part of current frame. When the DMA controller reads a descriptor with LSG bit in TDES0 is set, it know the current buffers is pointing to the last part of the current frame. Normally one frame is stored only in one buffer (because buffer size is large enough for a normal frame), so FSG bit and LSG bit are set in the same descriptor.

For the frame receiving process, the receive buffer size must be word align. But for word-align buffer address or not word-align buffer address, the operation is different from transmitting. When the receive buffer address is word align, it's no difference with transmitting process, the effective length of the buffer is the same as the value configured by application in RDES1. When the receive buffer address is not word align, the effective length of the buffer is less than the value configured by application in RDES1. The effective length of the buffer should be the size value minus the low two bits value of buffer address. For example, assuming the total buffer size is 2048 bytes and buffer address is 0x2000 0001, the low two bits are 0b01, the effective length of the buffer is 2047 bytes whose address range is from 0x20000001 (for the first received frame byte) to 0x2000 07FF.

When a start of frame (SOF) is received, the FSG bit is set by DMA controller and when the end of the frame (EOF) is received, the LSG bit is set. If the receive buffer size is programmed to be large enough to store the whole frame, the FSG and the LSG bit are set in the same descriptor. The actual frame length FRML can be read from RDES0. So application can calculate the left unused buffer space. The RxDMA always uses a new descriptor to receive the start of next frame.

### Arbitration for TxDMA and RxDMA controller

There are two types of arbitration method designed for improving the efficiency of DMA controller between transmission and reception: fixed-priority and round-robin. When DAB bit in ENET\_DMA\_BCTL register is reset, arbiter selects round-robin method. The arbiter allocates the data bus in the ratio set by the RTPR bits in ENET\_DMA\_BCTL, when both of TxDMA and RxDMA controller request access simultaneously. When DAB bit in

ENET\_DMA\_BCTL register is set, arbiter selects fixed-priority, and the RxDMA controller always has higher priority over the TxDMA.

### Error response to DMA controller

During the operation of the DMA controller, when a response error presents on the bus, the DMA controller considers a fatal error occurs and stops operating at once with error flags written to the DMA status register (ENET\_DMA\_STAT). After such fatal error (response error) occurs, application must reset the Ethernet module and reinitialize the DMA controller.

### DMA controller initialization for transmission and reception

Before using the DMA controller, the initialization must be done as follow steps:

1. Set the bus access parameters by writing the ENET\_DMA\_BCTL register
2. Mask unnecessary interrupt source by configuring the ENET\_DMA\_INTEN register
3. Program the Tx and Rx descriptor table start address by writing the ENET\_DMA\_TDTADDR register and the ENET\_DMA\_RDTADDR register
4. Configure filter option by writing related registers
5. According to the auto-negotiation result with external PHY, set the SPD bit and DPM bit for selecting the communication mode (Half-duplex/Full-duplex) and the communication speed (10Mbit/s or 100Mbit/s). Set the TEN and REN bit in ENET\_MAC\_CFG register to enable MAC transmit and receive operations.
6. Set STE bit and SRE bit in ENET\_DMA\_CTL register to enable TxDMA controller and RxDMA controller

**Note:** If the HCLK frequency is too much low, application can enable RxDMA before set REN bit in ENET\_MAC\_CFG register to avoid Rx FIFO overflow at start time.

### TxDMA configuration

#### Operate on second frame in buffer

When OSF bit in ENET\_DMA\_CTL is reset, the order of the transmitting is follows: the first is reading transmit descriptor, followed by reading data from memory and writing to FIFO, then sending frame data on interface through MAC and last wait frame data transmitting complete and writing back transmitting status.

Above procedure is TxDMA's standard transmitting procedure but when HCLK is much faster than TX\_CLK, the efficiency of transmitting two frames will be greatly reduced.

To avoid the case mentioned above, application can set OSF to 1. If so, the second frame data can be read from the memory and push into FIFO without waiting the first frame's status writing back. OSF function is only performed between two neighboring frames.

#### TxDMA operation mode (A) (default mode): Non-OSF

The TxDMA controller in Non-OSF mode proceeds as follows:

1. Initialize the frame data into the buffer space and configure the descriptor (TDES0-3) with DAV bit of TDES0 sets to 1
2. Enable TxDMA controller by setting STE bit in ENET\_DMA\_CTL register
3. The TxDMA controller starts continue polling and performing transmit descriptor. When the DAV bit in TDES0[31] that TxDMA controller read is cleared, or any error condition occurs, the controller will enter suspend state and at the same time both the transmit buffer unavailable bit in ENET\_DMA\_STAT and normal interrupt summary bit in ENET\_DMA\_STAT register are set. If entered into suspend state, operation proceeds to Step 8
4. When the DAV bit in TDES0[31] of the acquired descriptor is set, the DMA decodes the transmit frame configured and the data buffer address from the acquired descriptor
5. DMA retrieve data from the memory and push it into the TxFIFO of MAC
6. The TxDMA controller continues polling the descriptor table until the EOF data (LSG bit is set) is transferred. If the LSG bit of current descriptor is reset, it will be closed by resetting the DAV bit after all buffer data pushed into TxFIFO. Then the TxDMA controller waits to write back descriptor status and IEEE 1588 timestamp value if enabled
7. After the whole frame is transferred, the transmit status bit (TS bit in ENET\_DMA\_STAT register) is set only when INTC bit in TDES0[30] is set. Also an interrupt generates if the corresponding interrupt enable flag is set. The TxDMA controller returns to Step 3 for the next frame
8. In the suspend state, application can make TxDMA returns to running state by writing any data to ENET\_DMA\_TPEN register and clearing the transmit underflow flag. Then the TxDMA controller process turns to Step 3.

### **TxDMA operation mode (B): OSF**

The TxDMA controller supports transmitting two frames without waiting status write back of the first frame, this mode is called operation on second frame (OSF). When the frequency of system is much faster than the frequency of the MAC interface (10Mbit/s or 100Mbit/s), the OSF mode can improve the sending efficiency. Setting OSF bit in ENET\_DMA\_CTL register can enable this mode. When the TxDMA controller received EOF of the first frame, it will not enter the state of waiting status write back but to fetch the next descriptor, if the DAV bit and FSG bit of the next descriptor is set, the TxDMA controller immediately read the second frame data and push them into the MAC FIFO.

The TxDMA controller in OSF mode proceeds as follows:

1. Follow steps 1-6 operation in TxDMA default mode
2. The TxDMA controller retrieves the next descriptor without closing the previous frame's last descriptor in which the LSG bit is set

3. If the DAV bit of the next descriptor is set, the TxDMA controller starts reading the next frame's data from the buffer address. If the DAV bit of the next descriptor is reset, TxDMA controller enters suspend state and the next operation goes to Step 7.
4. TxDMA controller continues polling descriptor and frame data until the EOF is transferred. If a frame is described with more than one descriptor, the intermediate descriptors are all closed by TxDMA controller after fetched.
5. The TxDMA controller enters the state of waiting for the transmission status and time stamp of the previous frame (if timestamp enabled). With writing back status to descriptor, the DAV bit is also cleared by TxDMA controller
6. After the whole frame is transferred, the transmit status bit (TS bit in ENET\_DMA\_STAT register) is set only when INTC bit in TDES0[30] is set. Also an interrupt generates if the corresponding interrupt enable flag is set. The TxDMA controller returns to Step 3 for the next frame if no underflow error occurred in previous frame. If underflow error of the previous frame is occurred, the TxDMA controller enters in suspend state and the next operation goes to Step 7.
7. In suspend state, when the status information and timestamp value (if the function is enable) of the transmitting frame is available, the TxDMA controller writes them back to descriptor and then close it by setting DAV=0 of descriptor.
8. In suspend state, application can make TxDMA returns to running state by writing any data to ENET\_DMA\_TPEN register and clearing the transmit underflow flag. Then the TxDMA controller process goes to Step 1 or Step 2.

### **Transmit frame format in buffer**

According to IEEE 802.3 specification described before, a frame structure is made up of such fields: Preamble, SFD, DA, SA, QTAG (option), LT, DATA, PAD (option), and FCS.

The Preamble and SFD are automatically generated by the MAC, so the application only need store the DA, SA, QTAG(if needed), LT, DATA, DATA, PAD(if needed), FCS(if needed) parts. If the frame needs padding which means PAD and FCS parts are not stored in buffer, then application can configure the MAC to generate the PAD and FCS. If the frame only need FCS which means only FCS part is not stored in buffer, the application can configure the MAC to generate FCS. The DPAD bit and DCRC bit are designed to achieve the generate function of the PAD and FCS field.

### **Transmit frame processing**

As mentioned before, a frame can span over several buffers which means several descriptors. When the FSG bit is set, the descriptor indicates the start of the frame and when the LSG bit is set, the descriptor indicates the end of the frame. All the buffers among these descriptors store the whole frame data. When the last descriptor is fetched and buffer finished reading, the transmitting status will write back to it. The other descriptors (here means the descriptor whose LSG bit is reset) of the current frame will not be changed by TxDMA controller except

the DAV bit will be reset to 0. After starting transfer frame data from memory to FIFO, the transmitting has not actually start. The real start time for sending frame on interface is depended on TxDMA mode: Cut-Through mode or Store-and-Forward mode. The former mode starts sending when the byte number of FIFO is greater than configured threshold and the latter mode starts sending when the whole frame data are transferred into FIFO or when the FIFO is almost full.

### Suspend during transmit polling

The DMA controller keeps querying the transmit descriptor after the transmission is started. If either of the following conditions happens, the DMA controller will enter suspend state and the transmit polling will stop. Though the DMA entered suspend state, the descriptor pointer is maintained to the descriptor following of the last closed descriptor.

- The DMA controller fetches a descriptor with DAV=0, then it enters suspend state and stops polling. In this case, the NI bit and TBU bit in ENET\_DMA\_STAT register are set.
- The MAC FIFO is empty during sending a frame on interface which means an error of underflow occurs. In this case, the AI bit and TU bit in ENET\_DMA\_STAT register are set. Also the transmit error status will write back to transmit descriptor.

### Transmit DMA descriptor with IEEE 1588 timestamp format

When TTSEN bit is set, the timestamp function is enabled. The TxDMA controller writes transmit timestamp status TTMSS and timestamp back to descriptor after the frame transmission complete. The word address in descriptor for writing timestamp is depends on DFM bit in ENET\_DMA\_BCTL register. If the descriptor format is normal mode (DFM=0), TDES2 and TDES3 are used for timestamp recording and the old values in TDES2 and TDES3 are overwritten. If the descriptor format is enhanced mode (DFM=1), TDES6 and TDES7 are used for timestamp recording and the value in TDES2 and TDES3 are kept.

### TxDMA descriptors in normal mode

The normal mode descriptor structure consists of four 32-bit words: TDES0 ~ TDES3. The descriptions of TDES0, TDES1, TDES2 and TDES3 are given below:

**Note:** When a frame is described by more than one descriptor, only the control bits of the first descriptor are accept by TxDMA controller (except INTC). But the status and timestamp (if enabled) are written back to the last descriptor.

**Figure 23-9. Transmit descriptor in normal mode**

	31	16										0											
TDES0	AV	Ctrl [30:26]	T S E N	R e s	Ctrl [23:20]	Res [19: 18]	T M S	Status[16:0]															
TDES1	Reserved [31:29]	Buffer 2 byte size [28:16]										Reserved [15:13]	Buffer 1 byte size [12:0]										
TDES2	Buffer 1 address[31:0]/Timestamp Low[31:0]																						
TDES3	Buffer 2 address[31:0] or Next descriptor address[31:0]/Timestamp High[31:0]																						

### ■ TDES0: Transmit descriptor word 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DAV	INTC	LSG	FSG	DCRC	DPAD	TTSEN	Reserved	CM[1:0]	TERM	TCHM	Reserved	TTMSS	IPHE		
rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ES	JT	FRMF	IPPE	LCA	NCA	LCO	ECO	VFRM	COCNT[3:0]			EXD	UFE	DB	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits	Fields	Descriptions
31	DAV	DAV bit  The DMA clears this bit either when it completes the frame transmission or the buffer allocated in the descriptor is read completely. This bit of the frame's first descriptor must be set after all subsequent descriptors belonging to the same frame have been set.  0: The descriptor is available for CPU not for DMA 1: The descriptor is available for DMA not for CPU
30	INTC	Interrupt on completion bit  This is valid only when the last segment (TDES0[29]) is set.  0: TS bit in ENET_DMA_STAT is not set when frame transmission complete. 1: TS bit in ENET_DMA_STAT is set when frame transmission complete.
29	LSG	Last segment bit  This bit indicates that the buffer contains the last segment of the frame. 0: The buffer of descriptor is not stored the last part of frame 1: The buffer of descriptor is stored the last part of frame
28	FSG	First segment bit  This bit indicates that the buffer contains the first segment of a frame. 0: The buffer of descriptor is not stored the first block of frame 1: The buffer of descriptor is stored the first block of frame
27	DCRC	Disable CRC bit  This is valid only when the first segment (TDES0[28]) is set. 0: The MAC automatic append a CRC to the end of the transmitted frame

		1: The MAC does not append a CRC to the end of the transmitted frame
26	DPAD	<p>Disable adding pad bit</p> <p>This is valid only when the first segment (TDES0[28]) is set.</p> <p>0: The DMA automatically adds padding byte and CRC to a frame shorter than 64 bytes. Only the padding actually acts, the CRC is also appended. The DCRC bit is don't care.</p> <p>1: The MAC does not automatically add padding to a frame</p>
25	TTSEN	<p>Transmit timestamp function enable bit.</p> <p>This field is only valid when the First segment control bit (TDES0[28]) is set.</p> <p>0: Disable transmit timestamp function</p> <p>1: When TMSEN is set (ENET_PTP_TSCTL bit 0), IEEE 1588 hardware time stamping is activated for the transmit frame</p>
24	Reserved	Must be kept at reset value.
23:22	CM[1:0]	<p>Checksum mode bits</p> <p>0x0: Disabled checksum insertion function</p> <p>0x1: Only enable function for IP header checksum calculation and insertion</p> <p>0x2: Enable IP header checksum and payload checksum calculation and insertion, pseudo-header checksum is not calculated in hardware</p> <p>0x3: Enable IP Header checksum and payload checksum calculation and insertion, pseudo-header checksum is calculated in hardware.</p>
21	TERM	<p>Transmit end for ring mode bit</p> <p>This bit is used only in ring mode and has higher priority than TCHM</p> <p>0: The current descriptor is not the last descriptor in the table</p> <p>1: The descriptor table reached its final descriptor. The DMA descriptor pointer returns to the start address of the table.</p>
20	TCHM	<p>The second address chained mode bit</p> <p>This bit is used only in chain mode. When this bit, TCHM (TDES0[20]), is set, TB2S (TDES1[28:16]) is don't care.</p> <p>0: The second address in the descriptor is the second buffer address</p> <p>1: The second address in the descriptor is the next descriptor address</p>
19:18	Reserved	Must be kept at reset value.
17	TTMSS	<p>Transmit timestamp status bit</p> <p>This bit is only valid when the descriptor's last segment (LSG) control bit (TDES0[29]) is set.</p> <p>0: Timestamp was not captured</p> <p>1: A timestamp was captured for the described transmit frame and push into TDES2 (or TDES6 if DFM=1) and TDES3 (or TDES7 if DFM=1)</p>
16	IPHE	<p>IP header error bit</p> <p>IP header error occurs when any case of below happen:</p>

		IPv4 frames:
		1) The header length field has a value less than 0x5. 2) The header length field value in transmitting IPv4 frame is mismatch with the number of header bytes 3) The version field value does not match the length/type field value
		IPv6 frames:
		1) The main header length is not 40 bytes 2) The version field value does not match the length/type field value 0:The MAC transmitter did not detect error in the IP datagram header 1:The MAC transmitter detected an error in the IP datagram header
15	ES	<p>Error summary bit</p> <p>Following bits are logical ORed to generate this bit:</p> <ul style="list-style-type: none"> <li>TDES0[16]: IP header error</li> <li>TDES0[14]: Jabber timeout</li> <li>TDES0[13]: Frame flush</li> <li>TDES0[12]: IP payload error</li> <li>TDES0[11]: Loss of carrier</li> <li>TDES0[10]: No carrier</li> <li>TDES0[9]: Late collision</li> <li>TDES0[8]: Excessive collision</li> <li>TDES0[2]:Excessive deferral</li> <li>TDES0[1]: Underflow error</li> </ul>
14	JT	<p>Jabber timeout bit</p> <p>Only set when the JBD bit is reset</p> <p>0:No jabber timeout occurred 1:The MAC transmitter has experienced a jabber timeout</p>
13	FRMF	<p>Frame flushed bit</p> <p>This bit is set to flush the Tx frame by software</p>
12	IPPE	<p>IP payload error bit</p> <p>The transmitter checks the payload length received in the IPv4 or IPv6 header against the actual number of TCP, UDP or ICMP packet bytes received from the application and issues an error status in case of a mismatch</p> <p>0&gt;No IP payload error occurred 1: MAC transmitter detected an error in the TCP, UDP, or ICMP/IP datagram payload</p>
11	LCA	<p>Loss of carrier bit</p> <p>When the interface signal ‘CRS’ lost one or more cycles and no collision happened during transmitting, the loss of carrier condition occurs.</p> <p>This is valid only in Half-duplex mode.</p> <p>0:No loss of carrier occurred 1:A loss of carrier occurred during frame transmission</p>

10	NCA	No carrier bit 0: PHY carrier sense signal is active 1: The carrier sense signal from the PHY was not asserted during transmission
9	LCO	Late collision bit If a collision occurs when 64 bytes (including preamble and SFD) has already transferred, this situation called late collision. 0: No late collision occurred 1: Late collision situation occurred <b>Note:</b> This bit is not valid if the UFE bit is set
8	ECO	Excessive collision bit If the RTD=1 (retry function disable), this bit is set after the first collision. If the RTD=0 (retry function enable), this bit is set when failed 16 successive retry transmitting. When this bit is set, the transmission of current frame is aborted. 0: No excessive collision occurred 1: Excessive collision occurred
7	VFRM	VLAN frame bit 0:The transmitted frame was a normal frame 1: The transmitted frame was a VLAN-type frame
6:3	COCNT[3:0]	Collision count bits This 4-bit counter value indicates the number of collisions occurring before the frame was transmitted. The count is not valid when the ECO bit (TDES0[8]) is set
2	EXD	Excessive deferral bit This is valid when the DFC bit in the MAC configuration register is set 0: No excessive deferral occurred 1: The transmission has ended because of excessive deferral time is over 3036 bytes
1	UFE	Underflow error bit This bit indicates that the TxDMA comes across an empty TxFIFO while transmitting the frame before EOF which is caused by pushing data to TxFIFO late from memory. The transmission process enters the suspend state and sets both the TU (bit 5) and the TS (bit 0) in ENET_DMA_STAT 0: No underflow error occurred 1: Underflow error occurred and the MAC aborted the frame transmitting
0	DB	Deferred bit This bit indicates whether the transmitting frame is deferred because of interface signal CRS is active before MAC transmit frame. Valid only in Half-duplex mode 0:No transmission deferred 1:The MAC is deferred before transmission

**■ TDES1: Transmit descriptor word 1**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		TB2S[12:0]								rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		TB1S[12:0]								rw					

Bits	Fields	Descriptions
31:29	Reserved	Must be kept at reset value
28:16	TB2S[12:0]	Transmit buffer 2 size bits These bits indicate byte size of the second data buffer. This field is not valid if the TCHM bit (TDES0[20]) is set.
15:13	Reserved	Must be kept at reset value
12:0	TB1S[12:0]	Transmit buffer 1 size bits These bits indicate the byte size of the first data buffer. If this field is 0, the TxDMA ignores this buffer and uses buffer 2 (for TCHM=0) or the next descriptor (for TCHM=1).

**■ TDES2: Transmit descriptor word 2**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		TB1AP/TTS1[31:16]								rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		TB1AP/TTS1[15:0]								rw					

Bits	Fields	Descriptions
31:0	TB1AP/TTS1[31:0]	Transmit buffer 1 address pointer/Transmit frame timestamp low 32-bit value bits Before transmitting frame, application must configure these bits for transmit buffer 1 address (TB1AP). When the transmitting frame is complete, these bits can be changed to the timestamp low 32-bit value (TTS1) for transmitting frame if DFM=0. But if DFM=1, these bits will not change and keep the value of buffer address. When these bits stand for buffer 1 address (TB1AP), the alignment is no limitation. When these bits stand for timestamp low 32-bit value, the TTSEN and LSG bit of current descriptor must be set.

**■ TDES3: Transmit descriptor word 3**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		TB2AP/TTS1[31:16]								rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		TB2AP/TTS1[15:0]								rw					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TB2AP/TTSH[15:0]															

rw

Bits	Fields	Descriptions
31:0	TB2AP/TTSH[31:0]	<p>Transmit buffer 2 address pointer (or next descriptor address) / Transmit frame timestamp high 32-bit value bits</p> <p>Before transmitting frame, application must configure these bits for transmit buffer 2 address (TB2AP) or the next descriptor address which is decided by descriptor type is ring or chain. When the transmitting frame is complete, these bits can be changed to the timestamp high 32-bit value (TTSH) for transmitting frame if DFM=0 and TTSEN =1. But if DFM=1 or TTSEN =0, these bits will not change and keep the old value. When these bits stand for buffer 2 address (TCHM=0), the alignment is no limitation. When these bits stand for the next descriptor address (TCHM=1), these bits must be word-alignment. When these bits stand for timestamp high 32-bit value, the TTSEN and LSG bit of current descriptor must be set.</p>

### **TxDMA descriptors in enhanced mode**

The enhanced mode descriptor structure consists of eight 32-bit words: TDES0 ~ TDES7. The descriptions of TDES0, TDES1, TDES2 and TDES3 are the same with normal mode descriptor; TDES4, TDES5, TDES6 and TDES7 are given below:

**Note:** When a frame is described by more than one descriptor, only the control bits of the first descriptor are accept by DMA controller (except INTC). But the status and timestamp (if enabled) are written back to the last descriptor.

**Figure 23-10. Transmit descriptor in enhanced mode**

	31	16								0																						
TDES0	AMG	Ctrl [30:26]	T T S E N	R e s	Ctrl [23:20]	Res [19: 18]	T T M S S	Status[16:0]																								
TDES1	Reserved [31:29]		Buffer 2 byte size [28:16]								Reserved [15:13]		Buffer 1 byte size [12:0]																			
TDES2	Buffer 1 address[31:0]																															
TDES3	Buffer 2 address[31:0] or Next descriptor address[31:0]																															
TDES4	Reserved																															
TDES5	Reserved																															
TDES6	Timestamp Low[31:0]																															
TDES7	Timestamp High[31:0]																															

■ TDES4 Transmit descriptor word 4

All bits reserved

■ TDES5 Transmit descriptor word 5

All bits reserved

■ TDES6 Transmit descriptor word 6

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

Bits	Fields	Descriptions
31:0	TTSL[31:0]	Transmit frame timestamp low 32-bit value bits  When TTSEN =1 and LSG=1, there bits are updated by TxDMA for recording timestamp low 32-bit value of the current transmitting frame.

■ TDES7 Transmit descriptor word 7

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

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

rw

Bits	Fields	Descriptions
31:0	TTSH[31:0]	Transmit frame timestamp high 32-bit value bits When TTSEN =1 and LSG=1, these bits are updated by TxDMA for recording timestamp high 32-bit value of the current transmitting frame.

## RxDMA configuration

The receiving process of the RxDMA controller is described detailed as below:

1. Applications initialize the receive descriptors with the DAV bit (RXDES0[31]) is set
2. Setting the SRE bit in ENET\_DMA\_CTL register to make RxDMA controller entering running state. In running state, the RxDMA controller continually fetching the receive descriptors from descriptor table whose starting address is configured in ENET\_DMA\_RDTADDR register by application. If the DAV bit of the fetched receive descriptor is set, then this descriptor is used for receiving frame. But if the DAV bit is reset which means this receive descriptor cannot be used by RxDMA, the RxDMA controller will enter suspend state and operation goes to Step 9
3. From the valid receive descriptor (DAV=1), the RxDMA controller marks the receiving control bit and data buffer address
4. Processing the received frames and transfer data to the receive buffer from the RxIFO.
5. If all frame data has completely transferred or the buffer is full, the RxDMA controller fetches the next descriptor from receive descriptor table.
6. If the current receiving frame transfer is complete, the operation of RxDMA goes to Step7. But if not complete, two conditions may occur:
  - 1) The next descriptor's DAV bit is reset. The RxDMA controller sets descriptor error bit DERR in RDES0 if flushing function is enabled. The RxDMA controller closes current descriptor by resetting DAV bit and sets the LSG bit (if flushing is enabled) or resets the LSG bit (if flushing is disabled). Then the operation goes to Step 8.
  - 2) The next descriptor's DAV bit is set. The RxDMA controller closes current descriptor by resetting DAV bit and operation goes to Step 4.
7. If IEEE 1588 time stamping function is enabled, the RxDMA controller writes the time stamp value (if receiving frame meets the configured time stamping condition) to the current descriptor's RDES2 and RDES3 if DFM=0 or RDES6 and RDES7 if DFM=1. At

the same time (writing timestamp value) the RxDMA controller also writes the received frame's status word to the RDES0 with the DAV bit cleared and the LSG bit set.

8. The latest descriptor is fetched by RxDMA controller. If the fetched descriptor bit 31 (DAV) is set, the RxDMA controller operation goes to Step 4. If the fetched descriptor bit 31 is reset, the RxDMA controller enters the suspend state and sets the RBU bit in register ENET\_DMA\_STAT. If flushing function is enabled, the RxDMA controller will flush the received frame data in the RxFIFO before entering suspend state.
9. In suspended state, there are two conditions to exit. The first is writing data in the ENET\_DMA\_RPEN register by application. The second is when a new received frame is available which means the byte number of receiving frame is greater than threshold in Cut-Through mode or when the whole frame is received in Store-and-Forward mode. Once exiting suspend mode, the RxDMA controller fetches the next descriptor and the following operation goes to Step 2.

### Receive descriptor fetching regulation

Descriptor fetching occurs if any one or more of the following conditions are met:

- The time SRE bit is configured from 0 to 1 which makes the RxDMA controller entering running state
- The total buffer size (buffer 1 for chain mode or buffer 1 plus buffer 2 for ring mode) of the current descriptor cannot hold the current receiving frame. In other word, the last byte stored in buffer space is not the EOF byte
- After a complete frame is transferred to buffer and before current descriptor is closed
- In suspend state, the MAC received a new frame
- Writing any value to receive poll enable register ENET\_DMA\_RPEN

### Process of receiving frame

When a frame is presented on the interface, the MAC starts to receive it. At the same time, the address filter block is running for this received frame. If the received frame fails the address filtering it will be discarded from RxFIFO in MAC and not be forwarded to buffer by RxDMA controller. If the received frame passes the address filtering, it will be forwarded to buffer when the available time comes. If the RxDMA controller is configured in Cut-Through mode, the available time means the byte number of the received frame is equal or greater than the configured threshold. If the RxDMA controller is configured in Store-and-Forward mode, the available time means the complete frame is stored in RxFIFO. During receiving frame, if any one of the below cases occurs the MAC can discard the received frame data in RxFIFO and the RxDMA controller will not forward these data: 1) The received frame bytes is less than 64. 2) Collision occurred during frame receiving. 3) The premature termination for the receiving frame.

When the available time comes, the RxDMA controller starts transfer frame data from RxFIFO to the receive buffer. If the SOF is included in current receive buffer, the FDES bit in RDES0 is set when the RxDMA controller writing receive frame status to indicate this descriptor is

used for storing the first part of the frame. If the EOF is included in current receive buffer, the LDES bit in RDES0 is set when RxDMA controller writing receive frame status to indicate this descriptor is used for storing the last part of the frame. Often when the buffer size is larger than received frame, the FDES and LDES bit are set in the same descriptor. When the EOF is transferred to buffer or the receive buffer space is exhausted, the RxDMA controller fetches the next receive descriptor and closes previous descriptor by writing RDES0 with DAV=0. If the LDES bit is set, the other status are also be updated and the RS bit in ENET\_DMA\_STAT register will be set (immediately when DINTC=0 or delayed when DINTC=1). If the DAV bit of the next descriptor is set, the RxDMA controller repeats above operation when received a new frame. If the DAV bit of the next descriptor is reset, the RxDMA controller enters suspend state and sets RBU bit in ENET\_DMA\_STAT register. The pointer value of descriptor address table is retained and be used for the starting descriptor address after exiting suspend state.

### Processing after a new frame received in suspend state

When a new frame is available (see available definition in the previous paragraph), the RxDMA controller fetches the descriptor. If the DAV bit in RDES0 is set, the RxDMA controller exits suspend state and returns to running state for frame reception. But if the DAV bit in RDES0 is reset, application can choose whether these received frame data in RxFIFO are flushed or not by configuring DAFRF bit in ENET\_DMA\_CTL register. If DAFRF=0, the RxDMA controller discards these received frame data and makes the missed frame counter (MSFC) increase one. If DAFRF=1, these frame data are will not be flushed and MSFC counter will not increase until the RxFIFO is full. If the DAV bit is reset in fetched descriptor, the RBU bit in ENET\_DMA\_STAT register will be set and the RxDMA controller will be still in suspend state.

### Receive DMA descriptor with IEEE 1588 timestamp format

If the IEEE 1588 function enabled, the MAC writes the timestamp value to RDES2 and RDES3 (DFM=0) or RDES6 and RDES7 (DFM=1) after a frame with timestamp reception complete and before the RxDMA controller clears the DAV bit.

### RxDMA descriptors in normal mode

In normal descriptor mode, the descriptor structure consists of four 32-bit words: RDES0 ~ RDES3. The detailed description of RDES0, RDES1, RDES2 and RDES3 are given below.

**Figure 23-11. Receive descriptor in normal mode**

	31	16												0										
RDES0	D A V	Status[30:0]																						
RDES1	CR TL	Res [30: 29]	Buffer 2 byte size [28:16]								CTRL [15: 14]	R e s	Buffer 1 byte size [12:0]											
RDES2	Buffer 1 address[31:0]/Timestamp Low[31:0]																							
RDES3	Buffer 2 address[31:0] or Next descriptor address[31:0]/Timestamp High[31:0]																							

■ RDES0: Receive descriptor word 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DAV	DAFF														FRML[13:0]
rw	rw														rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERRS	DERR	SAFF	LERR	OERR	VTAG	FDES	LDES	IPHERR/ TSV	LCO	FRMT	RWDT	RERR	DBERR	CERR	PCERR/E XSV
rw	rw	rw	rw	rw	rw	rw	rw								

Bits	Fields	Descriptions
31	DAV	Descriptor available bit  This bit indicates the DMA controller can use this descriptor. The DMA clears this bit either when it completes the frame reception or when the buffers in this descriptor are full  0: The descriptor is owned by the CPU 1: The descriptor is owned by the DMA
30	DAFF	Destination address filter fail bit  0: A frame passed the destination address filter 1: A frame failed the destination address filter
29:16	FRML[13:0]	Frame length bits  These bits indicate the byte length of the received frame that was transferred to the buffer (including CRC when received frame is not a type frame). If received frame is a type frame, including CRC or not is controlled by TFCD bit in ENET_MAC_CFG). This field is valid only when LDES=1 (RDES0[8]) and DERR=0 (RDES0[14]). If LDES=0 and ERRS=0, these bits indicate the accumulated number of bytes that have been transferred for the current frame.
15	ERRS	Error summary bit  This field is valid only when the LDES (RDES0[8]) is set.  This bit is logical ORed by the following bits when DFM is equal to 0: RDES0[14]: Descriptor error. RDES0[11]: Overflow error RDES0[6]: Late collision

		RDES0[4]: Watchdog timeout RDES0[3]: Receive error RDES0[1]: CRC error REDS0[7] = 0, REDS0[5] = 1 and REDS0[0] = 1: payload checksum error REDS0[7] = 1, REDS0[5] = 1 and REDS0[0] = 0: header checksum error REDS0[7] = 1, REDS0[5] = 1 and REDS0[0] = 1: both header and payload checksum errors This bit is logical ORed by the following bits when DFM is equal to 1: REDS4[4]: IP frame payload error REDS4[3]: IP frame header error RDES0[14]: Descriptor error REDS0[11]: Overflow error REDS0[6]: Late collision REDS0[4]: Watchdog timeout RDES0[3]: Receive error RDES0[1]: CRC error
14	DERR	Descriptor error bit  This field is valid only when the LDES (RDES0[8]) is set.  When the current buffer cannot hold current received frame and the next descriptor's DAV bit is reset, the descriptor error occurs. 0: No descriptor error occurred 1: Descriptor error occurred
13	SAFF	SA filtering fail bit  0: No source address filter fail occurred 1: A received frame failed the SA filter
12	LERR	Length error bit  This bit is valid only when the FRMT (RDES0[5]) bit is reset.  This bit indicates the mismatch between the length field in received and the actual frame length. 0: No length error occurred 1: Length error occurred
11	OERR	Overflow error bit  When RxFIFO is overflow and the frame data has been partly forwarded to descriptor buffer, the overflow error bit sets. 0: No overflow error occurred 1: RxFIFO overflowed and frame data is not valid
10	VTAG	VLAN tag bit  0: Received frame is not a tag frame 1: Received frame is a tag frame
9	FDES	First descriptor bit  This bit indicates that current descriptor contains the SOF of the received frame.

		0: The current descriptor does not store the SOF of the received frame 1: The current descriptor buffer saves the SOF of the received frame
8	LDES	Last descriptor bit  This bit indicates that current descriptor contains the EOF of the received frame 0: The current descriptor buffer does not store EOF of the received frame 1: The current descriptor buffer saves the EOF of the received frame
7	IPHERR/TSV	IP frame header error bit/Timestamp valid bit  When DFM=0, bit 7, 5 and 0 indicate some special cases refer to the error status table.  When DFM=1, this bit indicates the timestamp value is taken and write to the RDES6 and RDES7. This bit is valid only when LDES is set.
6	LCO	Late collision bit  This bit indicates a collision occurs after 64 bytes have been received This bit only valid in Half-duplex mode. 0: No late collision occurred 1: Late collision has occurred
5	FRMT	Frame type bit  When DFM=0, bit 7, 5 and 0 indicate some special cases refer to the error status table.  When DFM=1, this bit indicates the received frame is an Ethernet type frame or a tagged frame.  If the received frame is runt frame, this bit is not valid for application. 0: The received frame is an IEEE802.3 frame without tagged. 1: The received frame is an Ethernet-type frame (the length/type field is greater than or equal to 0x0600, or this is a tagged frame)
4	RWDT	Receive watchdog timeout bit  When WDD=0, this bit indicates a frame with more than 2048 bytes was detected. When WDD=1, this bit indicates a frame with more than 16384 bytes was detected. 0: No receive watchdog timeout occurred 1: Watchdog timer overflowed during receiving and current frame is only a part of frame.
3	RERR	Receive error bit  This bit indicates the interface signal RX_ER asserted when RX_DV signal is active during frame receiving process. 0:No receive error occurred 1:Receive error occurred
2	DBERR	Dribble bit error bit  This bit is valid only in MII interface mode and indicates there is an incomplete byte (odd cycles during reception) received.

		0:No dribble bit error occurred 1:Dribble bit error occurred
1	CERR	<p>CRC error bit</p> <p>This bit is valid only when the LDES (RDES0[8]) is set and indicates FCS field in received frame is mismatch with the calculation result of the hardware</p> <p>0:No CRC error occurred 1:A CRC error occurred</p>
0	PCERR/EXSV	<p>Payload checksum error bit/Extended status valid bit</p> <p>When DFM=0, bit 7, 5 and 0 indicate some special cases refer to the error status table.</p> <p>When DFM=1, this bit indicates the descriptor RDES4 is valid for application.</p> <p>This bit only valid when LDES is set.</p> <p>0:RDES4 is not valid for application 1:RDES4 is valid for application</p>

The following table shows the combination meaning for bit 7, 5, and 0 in RDES0:

**Table 23-6. Error status decoding in RDES0, only used for normal descriptor (DFM=0)**

Bit 7: IPHERR	Bit 5: FRMT	Bit 0: PCERR	Frame status
0	0	0	IEEE 802.3 normal frame (Length field value is less than 0x0600 and not tagged)
0	0	1	IPv4 or IPv6 frame, no header checksum error, payload checksum is bypassed because of unsupported payload type
0	1	0	IPv4 or IPv6 frame, checksum checking pass
0	1	1	IPv4 or IPv6 frame, payload checksum error. This error may case by following condition: 1) Calculated checksum value mismatch the checksum field 2) byte number of received payload mismatch length field
1	0	0	Reserved
1	0	1	A type (length/type field equal or greater than 0x0600) or tagged frame but neither IPv4 nor IPv6. Offload check engine is bypassed.
1	1	0	IPv4 or IPv6 frame, but an header checksum error detected This error may case by following condition: 1) Type value inconsistent with version value 2) Calculated header checksum mismatch the header checksum field 3) Expected IP header bytes is not received enough
1	1	1	IPv4 or IPv6 frame, both header and payload checksum detected errors

- RDES1: Receive descriptor word 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
DINTC	Reserved	RB2S[12:0]														
rw															rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
RERM	RCHM	Reserved	RB1S[12:0]													
rw	rw														rw	

Bits	Fields	Descriptions
31	DINTC	Disable interrupt on completion bit 0: RS bit in ENET_DMA_STAT register will immediately set after receiving the completed, then if enabled the corresponding interrupt, the interrupt will trigger. 1: RS bit in ENET_DMA_STAT register will not immediately set after receiving the completed, but will set after a configurable delay time.
30:29	Reserved	Must be kept at reset value
28:16	RB2S[12:0]	Receive buffer 2 size bits The second buffer size in bytes. The buffer size must be a multiple of 4. This field is ignored if RCHM (RDES1[14]) is set
15	RERM	Receive end of ring mode bit This bit indicates the final descriptor in table is arrived and the next descriptor address is automatically set to the configured start descriptor address. 0: Current descriptor is not the last descriptor in table 1: Current descriptor is the last descriptor in table
14	RCHM	Receive chained mode for second address bit 0: The second address points to the second buffer address. 1: The second address points to the next descriptor address. RB2S (RDES1[28:16]) is ignored. <b>Note:</b> If the RERM=1, the next descriptor returns to base address even this bit is set to 1.
13	Reserved	Must be kept at reset value
12:0	RB1S[12:0]	Receive buffer 1 size bits The first buffer size in bytes. The buffer size must be a multiple of 4. If this field is 0, the RxDMA controller ignores this buffer and uses buffer 2 (RCHM=0) or the next descriptor (RCHM=1)

#### ■ RDES2: Receive descriptor word 2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RB1AP/RTSL[31:16]															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

RB1AP/RTSL[15:0]

rw

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:0	RB1AP/RTSL[31:0]	<p>Receive buffer 1 address pointer / Receive frame timestamp low 32-bit</p> <p>These bits are designed for two different functions: buffer address pointer (RB1AP) or timestamp low 32-bit value (RTSL).</p> <p>RB1AP: Before fetching this descriptor by RxDMA controller, these bits are configured to the buffer 1 address by application. This buffer 1 address pointer is used for RxDMA controller to store the received frame if RB1S is not 0. The buffer address alignment has no limitation.</p> <p>RTSL: When timestamp function is enabled and LDES is set, these bits will be changed to timestamp low 32-bit value by RxDMA controller if received frame passed the filter and satisfied the snapshot condition. If the received frame does not meet the snapshot condition, these bits will keep RB1AP value.</p>

#### ■ RDES3: Receive descriptor word 3

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

RB2AP/RTSH[31:16]

rw

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

RB2AP/RTSH[15:0]

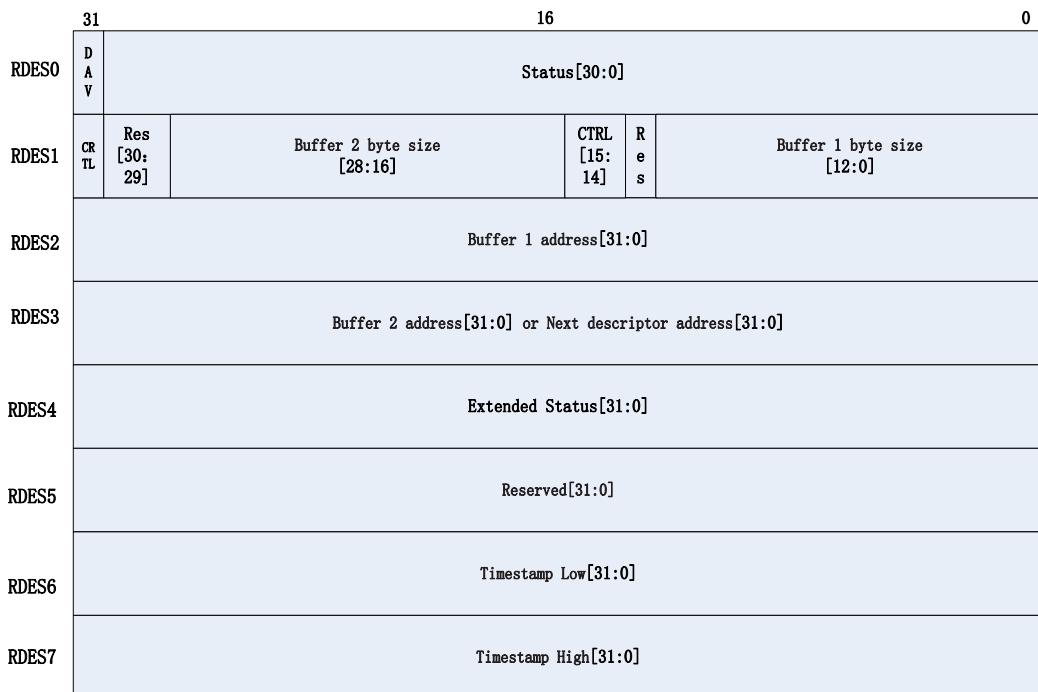
rw

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:0	RB2AP/RTSH[31:0]	<p>Receive buffer 2 address pointer (next descriptor address) / Receive frame timestamp high 32-bit value bits</p> <p>These bits are designed for two different functions: buffer address pointer or next descriptor address (RB1AP) or timestamp high 32-bit value (RTSH).</p> <p>RB2AP: Before fetching this descriptor by RxDMA controller, these bits are configured to the buffer 2 address (RCHM=0) or the next descriptor address (RCHM=1) by application. This buffer 2 address pointer is used for RxDMA controller to store the received frame if RB1S is not 0 when RCHM=0. If RCHM=1 and RERM=0, this address pointer is used for fetching the next descriptor. If RCHM=1 and RERM=1, these bits are ignored.</p> <p>When this address is used for next descriptor address, the word alignment is needed. The other conditions have no limitation for these bits.</p> <p>RTSH: When timestamp function is enabled and LDES is set, these bits will be changed to timestamp high 32-bit value by RxDMA controller if received frame passed the filter and satisfied the snapshot condition. If the received frame does not meet the snapshot condition, these bits will keep RB2AP value.</p>

### RxDMA descriptors in enhanced mode

In enhanced descriptor mode, the descriptor structure consists of eight 32-bit words: RDES0 ~ RDES7. The description of RDES0, RDES1, RDES2 and RDES3 are the same with descriptors in normal mode. The description of RDES4, RDES5, RDES6, and RDES7 are given below.

**Figure 23-12. Receive descriptor in enhanced mode**



#### ■ RDES4: Receive descriptor word 4

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	PTPVF	PTPOEF		PTPMFT[3:0]		IPF6	IPF4	IPCKSB	IPPLDER R	IPHERR		IPPLDT[2:0]				
	rw	rw		rw		rw	rw	rw	rw	rw	rw		rw		rw	

Bits	Fields	Descriptions
31:14	Reserved	Must be kept at reset value
13	PTPVF	PTP version format bit 0: Version 1 format 1: Version 2 format
12	PTPOEF	PTP on Ethernet frame bit 0: Received PTP frame is a IP-UDP frame if PTPMT is not zero

		1: Received PTP frame is a IEEE802.3 Ethernet frame
11:8	PTPMT[3:0]	<p>PTP message type bits</p> <p>PTP message type is decoded to following number:</p> <ul style="list-style-type: none"> <li>0x0: Not PTP frame received</li> <li>0x1: SYNC</li> <li>0x2: FOLLOW_UP</li> <li>0x3: DELAY_REQ</li> <li>0x4: DELAY_RESP</li> <li>0x5: For peer-to-peer transparent clock: PDELAY_REQ For ordinary or boundary clock: ANNOUNCE</li> <li>0x6: For peer-to-peer transparent clock: PDELAY_RESP For ordinary or boundary clock: MANAGEMENT</li> <li>0x7: For peer-to-peer transparent clock: PDELAY_RESP_FOLLOW_UP For ordinary or boundary clock: SIGNALING</li> </ul>
7	IPF6	<p>IP frame in version 6 bit</p> <p>0: Received frame is not a IPv6 frame</p> <p>1: Received frame is a IPv6 frame</p>
6	IPF4	<p>IP frame in version 4 bit</p> <p>0: Received frame is not a IPv4 frame</p> <p>1: Received frame is a IPv4 frame</p>
5	IPCKSB	<p>IP frame checksum bypassed bit</p> <p>This bit is only valid when received frame is a IPv4 or IPv6 frame</p> <p>0: Received frame checksum checking function is not bypassed</p> <p>1: Received frame checksum checking function is bypassed</p>
4	IPPLDERR	<p>IP frame payload error bit</p> <p>This bit can be set by any of below cases: 1) the calculated checksum by hardware mismatch with the TCP, UDP or ICMP checksum field in frame. 2) payload length value in IP header mismatch the received payload length</p> <p>0: Payload error not occurred in received frame</p> <p>1: Payload error occurred in received frame</p>
3	IPHERR	<p>IP frame header error bit</p> <p>This bit can be set by any of below cases: 1) the calculated checksum by hardware mismatch with the IP header checksum field value. 2) Type field in IP frame is not consistent with version field (e.g. ‘type’ field value is 0x0800 but ‘version’ field value is not 0x4, ‘type’ field value is 0x86dd but ‘version’ field value is not 0x6)</p> <p>0: IP header error not occurred</p> <p>1: IP header error occurred</p>
2:0	IPPLDT[2:0]	<p>IP frame payload type bits</p> <p>These bits are valid only when IPFCO=1, IPHERR=0 and LDES=1.</p>

- 0x0: Unsupported payload type or IP payload bypassed
- 0x1: payload type is UDP
- 0x2: payload type is TCP
- 0x3: payload type is ICMP
- 0x4~0x7: Reserved

■ **RDES5: Receive descriptor word 5**

All bits reserved

■ **RDES6: Receive descriptor word 6**

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

Bits	Fields	Descriptions
31:0	RTSL[31:0]	Receive frame timestamp low 32-bit value When timestamp function is enabled and LDES is set, these bits will be written to timestamp low 32-bit value by RxDMA controller if received frame passed the filter and satisfied the snapshot condition.

■ **RDES7: Receive descriptor word 7**

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

Bits	Fields	Descriptions
31:0	RTSH[31:0]	Receive frame timestamp high 32-bit value When timestamp function is enabled and LDES is set, these bits will be written to timestamp high 32-bit value by RxDMA controller if received frame passed the filter and satisfied the snapshot condition.

### 23.3.7. Example for a typical configuration flow of Ethernet

After power-on reset or system reset, the following operation flow is a typical process for application to configure and run Ethernet:

■ **Enable Ethernet clock.**

Program the RCU module to enable the HCLK and Ethernet Tx/Rx clock.

- **Setup the communication interface.**

Configure AFIO\_PCF0 to define which interface mode is selected (MII or RMII).  
Configure GPIO module to make selected PADs to alternate function.

- **Wait the resetting complete**

Polling the ENET\_DMA\_BCTL register until the SWR bit is reset. (SWR bit is set by default after power-on reset or system reset)

- **Obtain and configure the parameters in PHY register**

According to the frequency of HCLK, configure the SMI clock frequency and access external PHY register to obtain the information of PHY (e.g. support Half/Full duplex or not, support 10M/100Mbit speed or not, and so on). Based on supported mode of external PHY, configure ENET\_MAC\_CFG register consistent with PHY register.

- **Initialize the DMA in Ethernet module for transaction**

Configure the ENET\_DMA\_BCTL, ENET\_DMA\_RDTADDR, ENET\_DMA\_TDTADDR, ENET\_DMA\_CTL registers to initialize the DMA module. (Detailed information refer to [DMA controller description](#))

- **Initialize the physical memory space for descriptor table and data buffer**

According to the address value in ENET\_DMA\_RDTADDR and ENET\_DMA\_TDTADDR register, program transmitting and receiving descriptors (with DAV=1) and data buffer.

- **Enable MAC and DMA module to start transmit and receive**

Set TEN and REN bit in ENET\_MAC\_CFG register to make MAC work for transmit and receive. Set STE and SRE bit in ENET\_DMA\_CTL register to make DMA controller work for transmit and receive.

- **If transmitting frames is needed**

- 1) Choose one or more programmed transmitting descriptor, write the transmit frame data into buffer address which is decided in TDES.
- 2) Set the DAV bit in these one or more transmit frame descriptor.
- 3) Write any value in ENET\_DMA\_TPEN register to make TxDMA exit suspend state and start transmitting
- 4) There are two methods for application to confirm whether current transmitting frame is complete or not. The first method is that application can poll the DAV bit of current transmit descriptor until it is reset, this means the transmitting is complete. The second method can be used only when INTC=1. Application can poll the TS bit in ENET\_DMA\_STAT register until it is set, this means the transmitting is complete.

- **If receiving frames is enabled**

- 1) Check the first receive descriptor in descriptor table (whose address is configured in ENET\_DMA\_RDTADDR register).
- 2) If DAV bit in RDES0 is reset, then the descriptor is used and receive buffer space has stored the receive frame.
- 3) Handling this receive frame data.
- 4) Set DAV bit of this descriptor to release this descriptor for new frame receiving.
- 5) Check next descriptor in table, then goes to Step 2.

### 23.3.8. Ethernet interrupts

There are two interrupt vectors in Ethernet module. The first interrupt vector is made up of normal operation interrupts and the second vector is made up of WUM events for wakeup which is mapped to the EXTI line 19.

All of the MAC and DMA controller interrupt are connected to the first interrupt vector. The description for the MAC interrupt and DMA controller interrupt are showed behind.

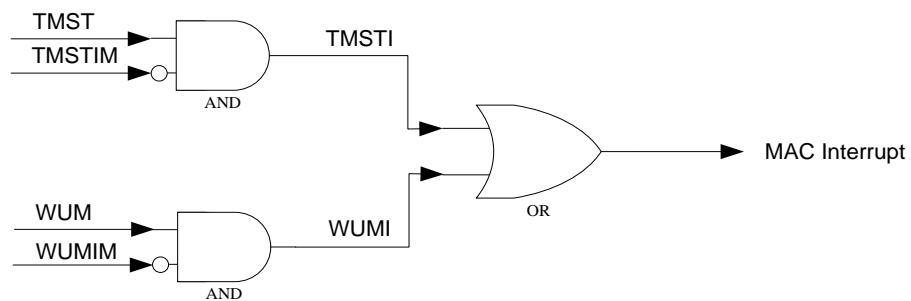
The WUM block event is connected to the second interrupt vector. The event can be remote wakeup frame received event or/and Magic Packet wakeup frame received event. This interrupt is inner mapped on the EXTI line 19. So, if the EXTI line 19 is enabled and configured to trigger by rising edge, the Ethernet WUM event can make the system exiting Deep-sleep mode after a WUM event occurred. In addition, if the WUM interrupt is not masked, both the EXTI line 19 interrupt and Ethernet normal interrupt to CPU are both generated.

**Note:** Because of the WUM registers are designed in RX\_CLK domain, clear these registers by reading them will need a long time delay (depends on the frequency disparity between HCLK and RX\_CLK). To avoid entering the same event interrupt twice, it's strongly recommended that application polls the WUFR and MPKR bit until they reset to zero during the interrupt service routine.

#### MAC interrupts

All of the MAC events can be read from ENET\_MAC\_INTF and each of them has a mask bit for masking corresponding interrupt. The MAC interrupt is logical ORed of all interrupts.

**Figure 23-13. MAC interrupt scheme**



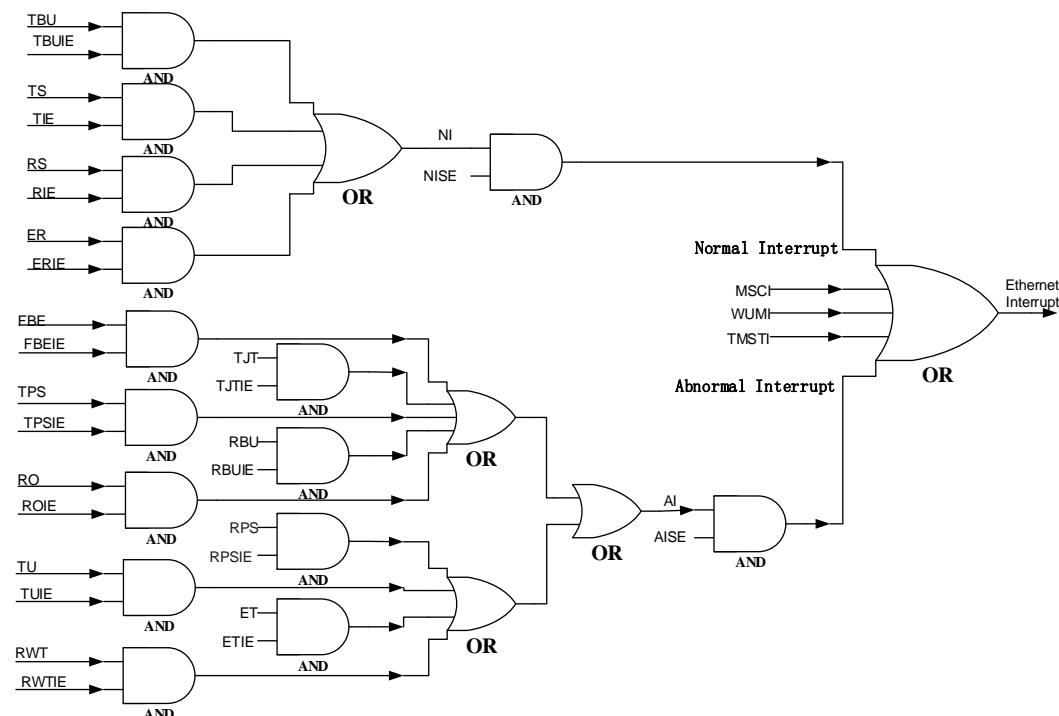
### DMA controller interrupts

The DMA controller has two types of event: Normal and Abnormal.

No matter what type the event is, it has an enable bit (just like mask bit) to control the generating interrupt or not. Each event can be cleared by writing 1 to it. When all of the events are cleared or all of the event enable bits are cleared, the corresponding summary interrupt bit is cleared. If both normal and abnormal interrupts are cleared, the DMA interrupt will be cleared.

Below block diagram illustrates the Ethernet module interrupt connection:

**Figure 23-14. Ethernet interrupt scheme**



## 23.4. Register definition

ENET base address: 0x4002 8000

Byte (8-bit) access, half word (16-bit) access and word (32-bit) access are all supported for application.

### 23.4.1. MAC configuration register (ENET\_MAC\_CFG)

Address offset: 0x0000

Reset value: 0x0000 8000

This register configures the operation mode of the MAC. It also configures the MAC receiver and MAC transmitter operating mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved					TFCD	Reserved	WDD	JBD	Reserved		IGBS[2:0]		CSD		
rw					rw		rw		rw		rw		rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	SPD	ROD	LBM	DPM	IPFCO	RTD	Reserved	APCD	BOL[1:0]	DFC	TEN	REN	Reserved		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			

Bits	Fields	Descriptions
31:26	Reserved	Must be kept at reset value
25	TFCD	Type Frame CRC Dropping 0 :FCS field(last 4 bytes) of frame will not be dropped before forwarding 1: FCS field(last 4 bytes) of frame will be dropped before forwarding <b>Note:</b> This bit only valid when LT field of frame greater than 0x0600
24	Reserved	Must be kept at reset value
23	WDD	Watchdog disable bit This bit indicates the maximum bytes for receiving, data beyond this will be cut off. 0: The MAC allows no more than 2048 bytes of the frame being received 1: The MAC disables the watchdog timer on the receiver, and can receive frames of up to 16384 bytes
22	JBD	Jabber disable bit This bit indicates the maximum bytes for transmitting data, data beyond this will be cut off. 0: The maximum transmission byte is 2048 1: The maximum transmission byte can be 16384
21:20	Reserved	Must be kept at reset value
19:17	IGBS[2:0]	Inter frame gap bit selection bits

		<p>These bits can select the minimum inter frame gap bit time between two neighboring frames during transmission.</p> <p>0x0: 96 bit times      0x1: 88 bit times      0x2: 80 bit times      0x3: 72 bit times      0x4: 64 bit times      0x5: 56 bit times(For Half-duplex, must be reserved)      0x6: 48 bit times(For Half-duplex, must be reserved)      0x7: 40 bit times(For Half-duplex, must be reserved)</p>
16	CSD	<p>Carrier sense disable bit</p> <p>0: The MAC transmitter generates carrier sense error and aborts the transmission      1: The MAC transmitter ignores the MII CRS signal during frame transmission in Half-duplex mode. Loss of carrier error and no carrier error will not be generated.</p>
15	Reserved	Must be kept at reset value
14	SPD	<p>Fast Ethernet speed bit</p> <p>Indicates the speed in Fast Ethernet mode:</p> <p>0: 10 Mbit/s      1: 100 Mbit/s</p>
13	ROD	<p>Receive own disable bit</p> <p>This bit is not applicable if the MAC is operating in Full-duplex mode</p> <p>0: The MAC receives all packets that are given by the PHY while transmitting      1: The MAC disables the reception of frames in Half-duplex mode</p>
12	LBM	<p>Loopback mode bit</p> <p>0: The MAC operates in normal mode      1: The MAC operates in loopback mode at the MII.</p>
11	DPM	<p>Duplex mode bit</p> <p>0: Half-duplex mode enable      1: Full-duplex mode enable</p>
10	IPFC0	<p>IP frame checksum offload bit</p> <p>0: The checksum offload function in the receiver is disabled      1: IP frame checksum offload function enabled for received IP frame</p>
9	RTD	<p>Retry disable bit</p> <p>This bit is applicable only in the Half-duplex mode</p> <p>0: The MAC attempts retries up to 16 times based on the settings of BOL      1: The MAC attempts only 1 transmission.</p>
8	Reserved	Must be kept at reset value
7	APCD	<p>Automatic pad/CRC drop bit</p> <p>This bit only valid for a non tagged frame and its length field value is equal or less</p>

		than 1536.
		0: The MAC forwards all received frames without modify it
		1: The MAC strips the Pad/FCS field on received frames
6:5	BOL[1:0]	Back-off limit bits  When a collision occurred, the MAC needs to retry sending current frame after delay some time. The base time unit for this delay time (dt) called slot time which means 1 slot time is equal to 512 bit times. This delay time (dt) is a random integer number calculated by following formula : $0 \leq dt < 2^k$  0x0: $k = \min(n, 10)$ 0x1: $k = \min(n, 8)$ 0x2: $k = \min(n, 4)$ 0x3: $k = \min(n, 1)$ , $n = \text{number of times for retransmission attempt}$ <b>Note:</b> This bit is valid only in Half-duplex mode
4	DFC	Deferral check bit  0: The deferral check function is disabled. MAC defers sending until the CRS goes inactive.  1: The deferral check function is enabled in the MAC. If deferred more than 24288 bit times, excessive deferral error occurs and MAC abort transmitting frame. If CRS signal active during deferral time running, the deferral time will reset and restart. <b>Note:</b> This bit is valid only in Half-duplex mode
3	TEN	Transmitter enable bit  0: The MAC transmit function is disabled after finish the transmission of the current frame, and no frames to be transmitted anymore  1: The transmit function of the MAC is enabled for transmission
2	REN	Receiver enable bit  0: The MAC reception function is disabled after finish the reception of the current frame, and no frames will be received anymore.  1: The MAC reception function is enabled for receiving frames
1:0	Reserved	Must be kept at reset value

### 23.4.2. MAC frame filter register (ENET\_MAC\_FRMF)

Address offset: 0x0004

Reset value: 0x0000 0000

This register configures the filtering method for receiving frames

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

rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				HPFLT	SAFLT	SAIFLT	PCFRM[1:0]	BFRMD	MFD	DAIFLT	HMF	HUF	PM		

rw      rw

Bits	Fields	Descriptions
31	FAR	<p>Frames all receive bit</p> <p>This bit controls the receive filter function.</p> <p>0: Only the frame passed the filter can be forwarded to application.</p> <p>1: All received frame are forwarded to application. But filter result will also be updated to receive descriptor status.</p>
30:11	Reserved	Must be kept at reset value
10	HPFLT	<p>Hash or perfect filter bit</p> <p>0: If the HUF or HMF bit is set, only frames that match the hash filter are passed</p> <p>1: If the HUF or HMF bit is set, the receive filter passes frames that match either the perfect filtering or the hash filtering</p>
9	SAFLT	<p>Source address filter bit</p> <p>Enable source address filtering function besides destination address filtering.</p> <p>The filter also compares the SA field value in received frames with the values configured in the enabled SA registers. If SA comparison matches, the SA match bit in the receive descriptor status is set high</p> <p>0: Source address function in filter disable</p> <p>1: Source address function in filter enable</p>
8	SAIFLT	<p>Source address inverse filtering bit</p> <p>This bit makes the result of SA matching inverse.</p> <p>0: Not inverse for source address filtering</p> <p>1: Inverse source address filtering result. When SA matches the enabled SA registers, filter marks it as failing the SA address filter</p>
7:6	PCFRM[1:0]	<p>Pass control frames bits</p> <p>These bits set the forwarding conditions for all control frames (including unicast and multicast pause frame).</p> <p>For pause control frame, the processing (not forwarding) depends only on RFCEN in ENET_MAC_FCTL[2]</p> <p>0x0: MAC prevents all control frames from reaching the application</p> <p>0x1: MAC only forwards all other control frames except pause control frame</p> <p>0x2: MAC forwards all control frames to application even if they fail the address filter</p> <p>0x3: MAC forwards control frames that only pass the address filter</p>
5	BFRMD	<p>Broadcast frames disable bit</p> <p>0:The address filters pass all received broadcast frames</p> <p>1:The address filters filter all incoming broadcast frames</p>

4	MFD	Multicast filter disable bit 0:Filtering of multicast frame depends on the HMF bit 1:All received frames with a multicast destination address (first bit in the destination address field is '1' and not all bits in the destination are '1') are passed
3	DAIFLT	Destination address inverse filtering bit This bit makes the result of DA filtering inverse 0: Not inverse DA filtering result 1: Inverse DA filtering result
2	HMF	Hash multicast filter bit 0: The filter uses perfect mode for filtering multicast frame. 1: The filter uses hash mode for filtering multicast frame
1	HUF	Hash unicast filter bit 0: The filter uses perfect mode for filtering unicast frame 1: The filter uses hash mode for filtering unicast frame
0	PM	Promiscuous mode bit This bit can make the filter bypassed which means all received frames are thought pass the filer and DA/SA filtering result status in descriptor is always '0'. 0: Promiscuous mode disabled 1: Promiscuous mode enabled

### 23.4.3. MAC hash list high register (ENET\_MAC\_HLH)

Address offset: 0x0008

Reset value: 0x0000 0000

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

Bits	Fields	Descriptions
31:0	HLH[31:0]	Hash list high bits These bits take the high 32-bit value of hash list

### 23.4.4. MAC hash list low register (ENET\_MAC\_HLL)

Address offset: 0x000C

Reset value: 0x0000 0000

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

Bits	Fields	Descriptions
31:0	HLL[31:0]	Hash list low bits These bits take the low 32-bit value of hash list

### 23.4.5. MAC PHY control register (ENET\_MAC\_PHY\_CTL)

Address offset: 0x0010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PA[4:0]				PR[4:0]				Reserved		CLR[2:0]		PW		PB	
rw				rw				rw		rw		rw		rc_w1	

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:11	PA[4:0]	PHY address bits These bits choose which PHY device is to be accessed
10:6	PR[4:0]	PHY register bits These bits choose the register address in selected PHY device
5	Reserved	Must be kept at reset value
4:2	CLR[2:0]	Clock range bits MDC clock divided factor select which is decided by HCLK frequency range 0x0: HCLK/42 (HCLK range: 60-100 MHz) 0x1: HCLK/62 (HCLK range: 100-120 MHz ) 0x2: HCLK/16 (HCLK range: 20-35 MHz) 0x3: HCLK/26 (HCLK range: 35-60 MHz) other: Reserved
1	PW	PHY write bit This bit indicate the PHY operation mode 0: Sending read operation to PHY

## 1: Sending write operation to PHY

0	PB	PHY busy bit This bit indicates the running state of operation on PHY. Application sets this bit to 1 and should wait it cleared by hardware. Application must make sure this bit is zero before writing data to ENET_MAC_PHY_CTL register and reading/writing data from/to ENET_MAC_PHY_DATA register
---	----	---

### 23.4.6. MAC PHY data register (ENET\_MAC\_PHY\_DATA)

Address offset: 0x0014

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
PD[15:0]															

rw

Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	PD[15:0]	PHY data bits For reading operation, these bits contain the data from external PHY. For writing operation, these bits contain the data will be sent to external PHY.

### 23.4.7. MAC flow control register (ENET\_MAC\_FCTL)

Address offset: 0x0018

Reset value: 0x0000 0000

This register configures the generation and reception of the control frames.

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

Reserved	DZQP	Reserved	PLTS[1:0]	UPFDT	RFCEN	TFCEN	FLCB/BK PA
----------	------	----------	-----------	-------	-------	-------	------------

rw                    rw                    rw                    rw                    rw                    rw                    rw

Bits	Fields	Descriptions
31:16	PTM[15:0]	Pause time bits

These bits configured the pause time filed value in transmit pause control frame.

15:8	Reserved	Must be kept at reset value
7	DZQP	<p>Disable Zero-quanta pause bit</p> <p>0: Enable automatic zero-quanta generation function for pause control frame.</p> <p>1: Disable the automatic zero-quanta generation function for pause control frame</p>
6	Reserved	Must be kept at reset value
5:4	PLTS[1:0]	<p>Pause low threshold bits</p> <p>These bits configure the threshold of the pause timer for retransmitting frames automatically. Application must make sure the low threshold bits are greater than 0 and less than configured pause time. The low threshold calculation formula is PTM-PLTS. For example, if PTM = 0x80 (128 slot-times), and PLTS = 0x1 (28 slot-times), then the second pause frame is automatically transmitted when pause timer counted at 100 (128 - 28) slot-times after the first pause frame is transmitted</p> <p>0x0: Pause time minus 4 slot times</p> <p>0x1: Pause time minus 28 slot times</p> <p>0x2: Pause time minus 144 slot times</p> <p>0x3: Pause time minus 256 slot times</p> <p><b>Note:</b> One slot time equals the time of transmitting 512 bits on the MII interface</p>
3	UPFDT	<p>Unicast pause frame detect bit</p> <p>0: Only the unique multicast address for pause frame which is specified in IEEE802.3 can be detected.</p> <p>1: Besides the unique multicast address, MAC can also use the MAC0 address (ENET_MAC_ADDR0H and ENET_MAC_ADDR0L register) to detecting pause frame.</p>
2	RFCEN	<p>Receive flow control enable bit</p> <p>0: Decode function for pause frame is disabled</p> <p>1: Enable decoding function for the received pause frame and process it. The MAC disables its transmitter for a specified (pause time field value in received frame) time</p>
1	TFCEN	<p>Transmit flow control enable bit</p> <p>0: Disable the flow control operation in the MAC. Both pause frame sending in Full-duplex mode and back-pressure feature in Half-duplex mode are not performed.</p> <p>1: Enable the flow control operation in the MAC. Both pause frame sending in Full-duplex mode and back-pressure feature in Half-duplex mode can be performed by transmitter.</p>
0	FLCB/BKPA	<p>Flow control busy/back pressure activate bit</p> <p>This bit only valid when TFCEN is set.</p> <p>This bit can send a pause frame in Full-duplex mode or activate the back pressure function in Half-duplex mode by application.</p> <p>For Full-duplex mode, application must make sure this bit is 0 before writing ENET_MAC_FCTL register. After set by application, MAC sends a pause frame to</p>

interface and this bit will keep set until the pause frame has completed transmitting. For Half-duplex mode, MAC can enter back-pressure state by application setting this bit. When the MAC is in back-pressure state, any frame presented on interface will make the MAC send a JAM pattern to inform outside a collision occurred.

### 23.4.8. MAC VLAN tag register (ENET\_MAC\_VLT)

Address offset: 0x001C

Reset value: 0x0000 0000

This register configures the IEEE 802.1Q VLAN Tag to identify the VLAN frames. The MAC compares the 13<sup>th</sup> and 14<sup>th</sup> byte (length/type field) of the receiving frame with 0x8100, and the following 2 bytes (the 15<sup>th</sup> and 16<sup>th</sup> byte) are compared with the VLAN tag.

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

Bits	Fields	Descriptions
31:17	Reserved	Must be kept at reset value
16	VLTC	<p>12-bit VLAN tag comparison bit</p> <p>This bit selects 12 or 16 bit VLAN tag for comparison.</p> <p>0: All 16 bits (the 15<sup>th</sup> and 16<sup>th</sup> byte) of the VLAN tag in received frame are used for comparison.</p> <p>1: Only low 12 bits of the VLAN tag in received frame are used for comparison.</p>
15:0	VLTI[15:0]	<p>VLAN tag identifier (for receive frames) bits</p> <p>These bits are configured for detecting VLAN frame using 802.1Q VLAN tag format. The format shows below:</p> <ul style="list-style-type: none"> <li>VLTI[15:13]: UP(user priority)</li> <li>VLTI[12]: CFI(canonical format indicator)</li> <li>VLTI[11:0]: VID(VLAN identifier)</li> </ul> <p>When comparison bits (VLTI[11:0] if VLTC=1 or VLTI[15:0] if VLTC=0) are all zeros, VLAN tag comparison is bypassed and every frame with type filed value of 0x8100 is considered a VLAN frame.</p> <p>When comparison bits not all zeros, VLAN tag comparison use bit VLTI[11:0] (if VLTC=1) or VLTI[15:0] (if VLTC=0) for checking.</p>

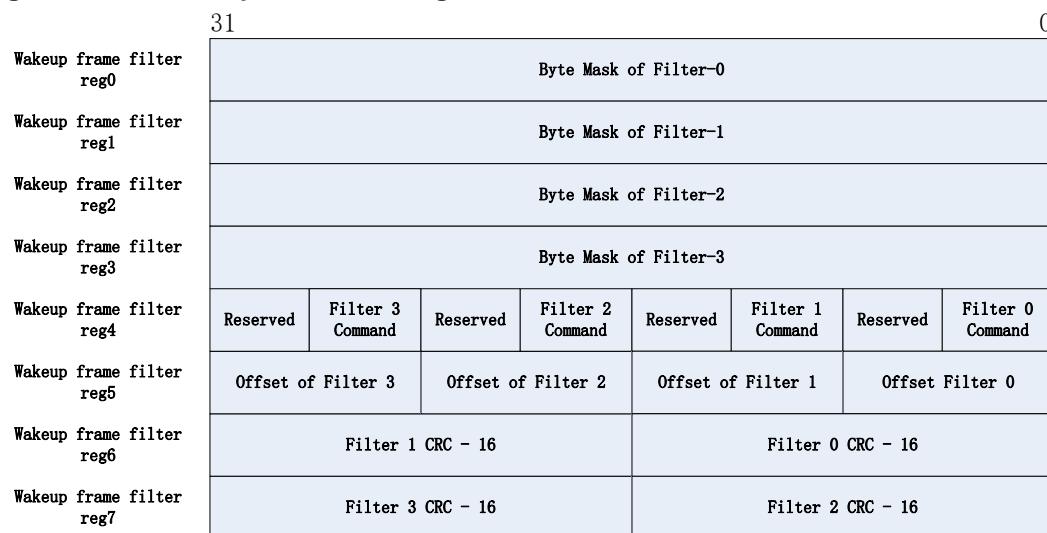
### 23.4.9. MAC remote wakeup frame filter register (ENET\_MAC\_RWFF)

Address offset: 0x0028

Reset value: 0x0000 0000

The MAC remote wakeup frame filter register is actually a pointer to eight (with same address offset) such wakeup frame filter registers. Eight sequential write operations to this address with the offset (0x0028) will write all wakeup frame filter registers. Eight sequential read operations from this address with the offset (0x0028) will read all wakeup frame filter registers.

**Figure 23-15. Wakeup frame filter register**

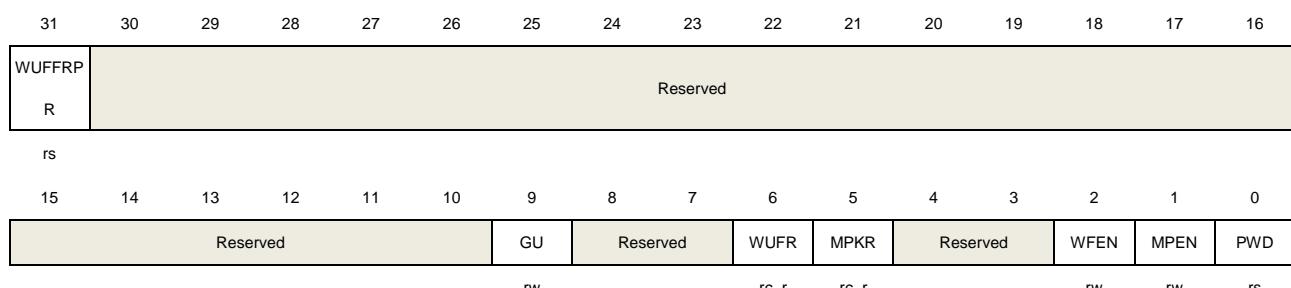


#### 23.4.10. MAC wakeup management register (ENET\_MAC\_WUM)

Address offset: 0x002C

Reset value: 0x0000 0000

This register configures the request of wakeup events and monitors the wakeup events.



Bits	Fields	Descriptions
31	WUFFRPR	Wakeup frame filter register pointer reset bit This bit can reset the inner pointer of ENET_MAC_RWFF register by application set it to 1. Hardware clears it when resetting completes. 0: No effect 1: Reset the ENET_MAC_RWFF register inner pointer
30:10	Reserved	Must be kept at reset value
9	GU	Global unicast bit

		0: Not all of received unicast frame is considered to be a wakeup frame 1: Any received unicast frame passed address filtering is considered to be a wakeup frame
8:7	Reserved	Must be kept at reset value
6	WUFR	<p>Wakeup frame received bit</p> <p>This bit is cleared when this register is read</p> <p>0:Has not received the wake-up frame 1:The wakeup event was generated due to reception of a wakeup frame</p>
5	MPKR	<p>Magic packet received bit</p> <p>This bit is cleared when this register is read</p> <p>0:Has not received the Magic Packet frame 1:The wakeup event was generated by the reception of a Magic Packet frame</p>
4:3	Reserved	Must be kept at reset value
2	WFEN	<p>Wakeup frame enable bit</p> <p>0: Disable generating a wakeup event due to wakeup frame reception 1: Enable generating a wakeup event due to wakeup frame reception</p>
1	MPEN	<p>Magic Packet enable bit</p> <p>0:Disable generating a wakeup event due to Magic Packet reception 1: Enable generating a wakeup event due to Magic Packet reception</p>
0	PWD	<p>Power down bit</p> <p>This bit is set by application and reset by hardware. When this bit is set, MAC drops all received frames. When power-down mode exit because of wakeup event occurred, hardware resets this bit.</p>

### 23.4.11. MAC debug register (ENET\_MAC\_DBG)

Address offset: 0x0034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved					TXFF	TXFNE	Reserved	TXFW	TXFRS[1:0]		PCS	SOMT[1:0]		MTNI	
ro	ro	ro	ro	ro	ro	ro	ro	ro	ro	ro	ro	ro	ro	ro	ro
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					RXFS[1:0]		Reserved	RXFRS[1:0]		RXFW	Reserved	RXAFS[1:0]		MRNI	
ro	ro	ro	ro	ro	ro	ro	ro	ro	ro	ro	ro	ro	ro	ro	ro

Bits	Fields	Descriptions
31:26	Reserved	Must be kept at reset value
25	TXFF	<p>TxFIFO Full flag</p> <p>0: TxFIFO not full</p>

		1: TxFIFO is full
24	TXFNE	TxFIFO not empty flag 0: TxFIFO is empty 1: TxFIFO is not empty
23	Reserved	Must be kept at reset value
22	TXFW	TxFIFO is writing 0: TxFIFO is not doing write operation 1: TxFIFO is doing write operation
21:20	TXFRS[1:0]	TxFIFO read operation status 0x0: Idle state 0x1: Reading state 0x2: Waiting feedback Tx status from MAC transmitter 0x3: Write Tx descriptor status or flush the TxFIFO
19	PCS	Pause condition status 0: MAC transmitter is not in pause condition 1: MAC transmitter is under pause condition and will delay transmitting frame
18:17	SOMT[1:0]	Status of MAC transmitter 0x0:Idle state 0x1:Waiting feedback of previous frame status or the end of IFG/BACKOFF period 0x2:For Full-duplex mode, indicates pause control frame is transmitting 0x3:Reading input frame from FIFO for transmission
16	MTNI	MAC transmit state not idle 0: MAC transmitter is in idle state 1: MAC transmitter is not in idle state
15:10	Reserved	Must be kept at reset value
9:8	RXFS	RxFIFO state 0x0:RxFIFO empty 0x1:RxFIFO number of value is lower than flow-control low threshold 0x2:RxFIFO number of value is greater than flow-control high threshold 0x3:RxFIFO full
7	Reserved	Must be kept at reset value
6:5	RXFRS[1:0]	RxFIFO read operation status 0x0: Idle state 0x1: Reading state 0x2: Reading frame status(including time-stamp) 0x3: Flushing frame
4	RXFW	RxFIFO is writing 0: RxFIFO is not doing write operation

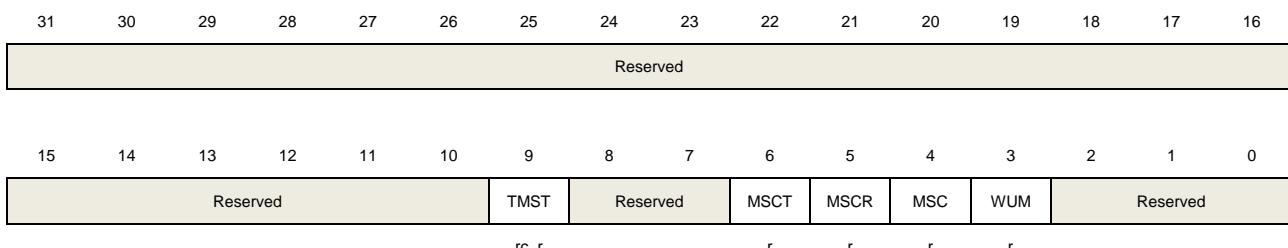
1: RxFIFO is doing write operation

3	Reserved	Must be kept at reset value
2:1	RXAFS[1:0]	Rx asynchronous FIFO status RXAFS[1]:Rx asynchronous FIFO reading state in HCLK Clock domain RXAFS[0]:Rx asynchronous FIFO writing state in MAC RX_CLK Clock domain
0	MRNI	MAC receive state not idle 0: MAC receiver is in idle state 1: MAC receiver is not in idle state

### 23.4.12. MAC interrupt flag register (ENET\_MAC\_INTF)

Address offset: 0x0038

Reset value: 0x0000 0000



Bits	Fields	Descriptions
31:10	Reserved	Must be kept at reset value
9	TMST	Time stamp trigger status bit This bit is cleared when ENET_PTP_TS register is read 0: The system time value is less than the value specified in the target time registers 1: The system time value equals or exceeds the value specified in the target time registers
8:7	Reserved	Must be kept at reset value
6	MSCT	MSC transmit status bit 0: All the bits in register ENET_MSC_TINTF are cleared 1: An interrupt is generated in the ENET_MSC_TINTF register
5	MSCR	MSC receive status bit 0: All the bits in register ENET_MSC_RINTF are cleared 1: An interrupt is generated in the ENET_MSC_RINTF register
4	MSC	MSC status bit This bit is logic ORed from MSCT and MSCR bit. 0: Both MSCT and MSCR bits in this register are low

1: Any of bit 6 (MSCT) or bit 5 (MSCR) is set high

3	WUM	WUM status bit This bit is logic ORed from WUFR and MPKR bit in ENET_MAC_WUM register. 0: Wakeup frame or Magic Packet frame is not received 1: A Magic packet or remote wakeup frame is received in power down Mode
2:0	Reserved	Must be kept at reset value

### 23.4.13. MAC interrupt mask register (ENET\_MAC\_INTMSK)

Address offset: 0x003C

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					TMSTIM	Reserved					WUMIM	Reserved			
rw								rw							

Bits	Fields	Descriptions
31:10	Reserved	Must be kept at reset value
9	TMSTIM	Timestamp trigger interrupt mask bit 0:Unmask the timestamp interrupt generation 1:Mask the timestamp interrupt generation
8:4	Reserved	Must be kept at reset value
3	WUMIM	WUM interrupt mask bit 0: Unmask the interrupt generation due to the WUM bit in ENET_MAC_INTF register 1: Mask the interrupt generation due to the WUM bit in ENET_MAC_INTF register
2:0	Reserved	Must be kept at reset value

### 23.4.14. MAC address 0 high register (ENET\_MAC\_ADDR0H)

Address offset: 0x0040

Reset value: 0x8000 FFFF

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

ADDR0H[15:0]
--------------

rw

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31	MO	Always read 1 and must be kept
30:16	Reserved	Must be kept at reset value
15:0	ADDR0H[15:0]	MAC address0 high16-bit These bits contain the high 16-bit (bit 47 to 32) of the 6-byte MAC address0. These bits are used for address filtering in frame reception and address inserting in pause frame transmitting during transmit flow control.

#### 23.4.15. MAC address 0 low register (ENET\_MAC\_ADDR0L)

Address offset: 0x0044

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDR0L[31:16]															
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR0L[15:0]															
rw															

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:0	ADDR0L[31:0]	MAC address0 low 32-bit These bits contain the low 32-bit (bit 31 to 0) of the 6-byte MAC address0. These bits are used for address filtering in frame reception and address inserting in pause frame transmitting during transmit flow control.

#### 23.4.16. MAC address 1 high register (ENET\_MAC\_ADDR1H)

Address offset: 0x0048

Reset value: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
AFE	SAF		MB[5:0]					Reserved									
rw		rw								rw							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
ADDR1H[15:0]																	
rw																	

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

---

31	AFE	Address filter enable bit 0: The address filter ignores the MAC address1 for filtering 1: The address filter uses the MAC address1 for perfect filtering
30	SAF	Source address filter bit 0: The MAC address1[47:0] is used to comparing with the DA field of the received frame 1: The MAC address1[47:0] is used to comparing with the SA field of the received frame
29:24	MB[5:0]	Mask byte bits When they are set high, the MAC does not compare the corresponding byte of received DA/SA with the contents of the MAC address1 registers. Each bit controls one byte mask as follows: MB[5]: ENET_MAC_ADDR1H [15:8] MB[4]: ENET_MAC_ADDR1H [7:0] MB[3]: ENET_MAC_ADDR1L [31:24] MB[2]: ENET_MAC_ADDR1L[23:16] MB[1]: ENET_MAC_ADDR1L[15:8] MB[0]: ENET_MAC_ADDR1L [7:0]
23:16	Reserved	Must be kept at reset value
15:0	ADDR1H[15:0]	MAC address1 high [47:32] bits This field contains the high 16-bit (bit 47 to 32) of the 6-byte MAC address1

### 23.4.17. MAC address 1 low register (ENET\_MAC\_ADDR1L)

Address offset: 0x004C

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDR1L[31:16]															
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR1L[15:0]															
rw															

Bits	Fields	Descriptions
31:0	ADDR1L[31:0]	MAC address1 low 32-bit This field contains the low 32-bit of the 6-byte MAC address1

### 23.4.18. MAC address 2 high register (ENET\_MAC\_ADDR2H)

Address offset: 0x0050

Reset value: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFE	SAF	MB[5:0]								Reserved					
rw	rw			rw											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR2H[15:0]															
rw															

Bits	Fields	Descriptions
31	AFE	Address filter enable bit 0:The address filter ignores the MAC address2 for filtering 1:The address filter uses the MAC address2 for perfect filtering
30	SAF	Source address filter bit 0:The MAC address2[47:0] is used to comparing with the DA fields of the received frame 1:The MAC address2[47:0] is used to comparing with the SA fields of the received frame
29:24	MB[5:0]	Mask byte bits When they are set high, the MAC does not compare the corresponding byte of received DA/SA with the contents of the MAC address2 registers. Each bit controls one byte mask as follows: MB[5]: ENET_MAC_ADDR2H [15:8] MB[4]: ENET_MAC_ADDR2H [7:0] MB[3]: ENET_MAC_ADDR2L [31:24] MB[2]: ENET_MAC_ADDR2L[23:16] MB[1]: ENET_MAC_ADDR2L[15:8] MB[0]: ENET_MAC_ADDR2L [7:0]
23:16	Reserved	Must be kept at reset value
15:0	ADDR2H[15:0]	MAC address2 high 16-bit This field contains the high 16-bit (bit 47 to 32) of the 6-byte MAC address2

### 23.4.19. MAC address 2 low register (ENET\_MAC\_ADDR2L)

Address offset: 0x0054

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDR2L[31:16]															
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR2L[15:0]															

rw

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:0	ADDR2L[31:0]	MAC address2 low 32-bit This field contains the low 32-bit of the 6-byte MAC address2

### 23.4.20. MAC address 3 high register (ENET\_MAC\_ADDR3H)

Address offset: 0x0058

Reset value: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
AFE	SAF		MB[5:0]					Reserved									
rw	rw		rw														
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
ADDR3H[15:0]															rw		

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31	AFE	Address filter enable bit 0:The address filter ignores the MAC address3 for filtering 1:The address filter use the MAC address3 for perfect filtering
30	SAF	Source address filter bit 0:The MAC address3[47:0] is used to comparing with the DA fields of the received frame 1:The MAC address3[47:0] is used to comparing with the SA fields of the received frame
29:24	MB[5:0]	Mask byte bits When they are set high, the MAC does not compare the corresponding byte of received DA/SA with the contents of the MAC address3 registers. Each bit controls one byte mask as follows: MB[5]: ENET_MAC_ADDR3H [15:8] MB[4]: ENET_MAC_ADDR3H [7:0] MB[3]: ENET_MAC_ADDR3L [31:24] MB[2]: ENET_MAC_ADDR3L[23:16] MB[1]: ENET_MAC_ADDR3L[15:8] MB[0]: ENET_MAC_ADDR3L [7:0]
23:16	Reserved	Must be kept at reset value
15:0	ADDR3H[15:0]	MAC address3 high 16-bit This field contains the high 16-bit (bit 47 to 32) of the 6-byte MAC address3

### **23.4.21. MAC address 3 low register (ENET\_MAC\_ADDR3L)**

Address offset: 0x005C

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDR3L[31:16]															
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR3L[15:0]															
rw															

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:0	ADDR3L[31:0]	MAC address3 low 32-bit  This field contains the low 32-bit of the 6-byte MAC address3

### **23.4.22. MAC flow control threshold register (ENET\_MAC\_FCTH)**

Address offset: 0x1080

Reset value: 0x0000 0015

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															

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:7	Reserved	Must be kept at reset value
6:4	RFD[2:0]	Threshold of deactive flow control  This field configures the threshold of the deactive flow control. The value should always be less than the Threshold of active flow control value configured in bits[2:0]. When the value of the unprocessed data in RxFIFO is less than this value configured, the flow control function will deactivate.  0x0: 256 bytes 0x1: 512 bytes 0x2: 768 bytes 0x3: 1024 bytes 0x4: 1280 bytes 0x5: 1536 bytes 0x6,0x7: 1792 bytes

---

3	Reserved	Must be kept at reset value
2:0	RFA[2:0]	<p>Threshold of active flow control</p> <p>This field configures the threshold of the active flow control. If flow control function is enabled, when the value of the unprocessed data in RxFIFO is more than this value configured, the flow control function will active.</p> <p>0x0: 256 bytes 0x1: 512 bytes 0x2: 768 bytes 0x3: 1024 bytes 0x4: 1280 bytes 0x5: 1536 bytes 0x6,0x7: 1792 bytes</p>

### 23.4.23. MSC control register (ENET\_MSC\_CTL)

Address offset: 0x0100

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								AFHPM	PMC	MCFZ	RTOR	CTSR	CTR		

Bits	Fields	Descriptions
31:6	Reserved	Must be kept at reset value
5	AFHPM	<p>Almost full or half preset mode</p> <p>0: Preset all MSC counters to almost-half(0x7FFF FFF0) value</p> <p>1: Preset all MSC counters to almost-full(0xFFFF FFF0) value</p> <p><b>Note:</b> This bit is valid only when PMC is set</p>
4	PMC	<p>Preset MSC counter</p> <p>0: No effect</p> <p>1: Preset MSC counters to a preset value. Preset value depends on AFHPM.</p>
3	MCFZ	<p>MSC counter freeze bit</p> <p>0: MSC counters are not frozen</p> <p>1: Freezes all the MSC counters to their current value. RTOR bit can work on this frozen state.</p>
2	RTOR	<p>Reset on read bit</p> <p>0: The MSC counters are not reset after reading MSC counter</p> <p>1: The MSC counters are reset to zero after read them</p>

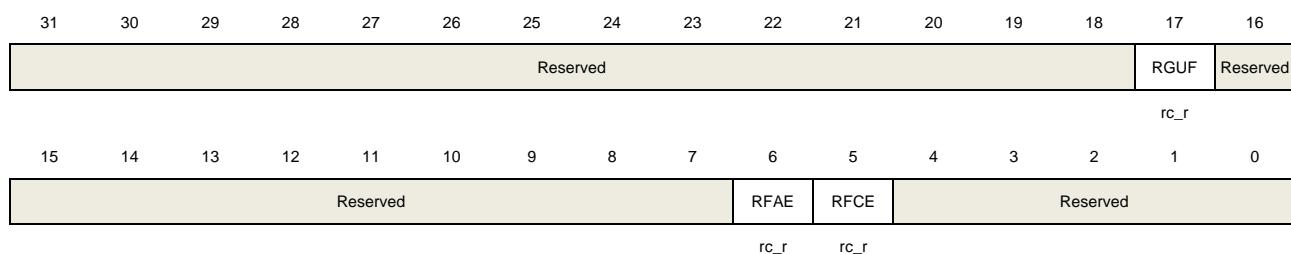
---

1	CTSR	Counter stop rollover bit 0: The counters roll over to zero after they reached the maximum value 1: The counters do not roll over to zero after they reached the maximum value
0	CTR	Counter reset bit Cleared by hardware 1 clock after set. This bit is cleared automatically after 1 clock cycle 0: No effect 1: Reset all counters

#### 23.4.24. MSC receive interrupt flag register (ENET\_MSC\_RINTF)

Address offset: 0x0104

Reset value: 0x0000 0000



Bits	Fields	Descriptions
31:18	Reserved	Must be kept at reset value
17	RGUF	Received good unicast frames bit 0: Good unicast frame received counter is less than half of the maximum value 1: Good unicast frame received counter reaches half of the maximum value
16: 7	Reserved	Must be kept at reset value
6	RFAE	Received frames alignment error bit 0: Alignment error frame received counter is less than half of the maximum value 1: Alignment error frame received counter reaches half of the maximum value
5	RFCE	Received frames CRC error bit 0: CRC error frame received counter is less than half of the maximum value 1: CRC error frame received counter reaches half of the maximum value
4:0	Reserved	Must be kept at reset value

#### 23.4.25. MSC transmit interrupt flag register (ENET\_MSC\_TINTF)

Address offset: 0x0108

Reset value: 0x0000 0000

---

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

Reserved												TGF	Reserved				
rc_r																	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
TGFMSC	TGFSC	Reserved															
rc_r	rc_r																

Bits	Fields	Descriptions
31:22	Reserved	Must be kept at reset value
21	TGF	Transmitted good frames bit 0: Good frame transmitted counter is less than half of the maximum value 1: Good frame transmitted counter reaches half of the maximum value
20:16	Reserved	Must be kept at reset value
15	TGFMSC	Transmitted good frames more single collision bit 0: Good frame after more than a single collision transmitted counter is less than half of the maximum value 1:Good frame after more than a single collision transmitted counter reaches half of the maximum value
14	TGFSC	Transmitted good frames single collision bit 0: Good frame after a single collision transmitted counter is less than half of the maximum value 1: Good frame after a single collision transmitted counter reaches half of the maximum value
13:0	Reserved	Must be kept at reset value

### 23.4.26. MSC receive interrupt mask register (ENET\_MSC\_RINTMSK)

Address offset: 0x010C

Reset value: 0x0000 0000

The Ethernet MSC receive interrupt mask register maintains the masks for interrupts generated when receive statistic counters reach half their maximum value

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

Bits	Fields	Descriptions
31:18	Reserved	Must be kept at reset value

17	RGUFIM	Received good unicast frames interrupt mask bit 0: Unmask the interrupt when the RGUF bit is set 1: Mask the interrupt when RGUF bit is set
16:7	Reserved	Must be kept at reset value
6	RFAEIM	Received frames alignment error interrupt mask bit 0: Unmask the interrupt when the RFAE bit is set 1: Mask the interrupt when the RFAE bit is set
5	RFCEIM	Received frame CRC error interrupt mask bit 0: Unmask the interrupt when RFCE bit is set 1: Mask the interrupt when the RFCE bit is set
4:0	Reserved	Must be kept at reset value

### 23.4.27. MSC transmit interrupt mask register (ENET\_MSC\_TINTMSK)

Address offset: 0x0110

Reset value: 0x0000 0000

The MSC transmit interrupt mask register configures the mask bits for interrupts generation

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

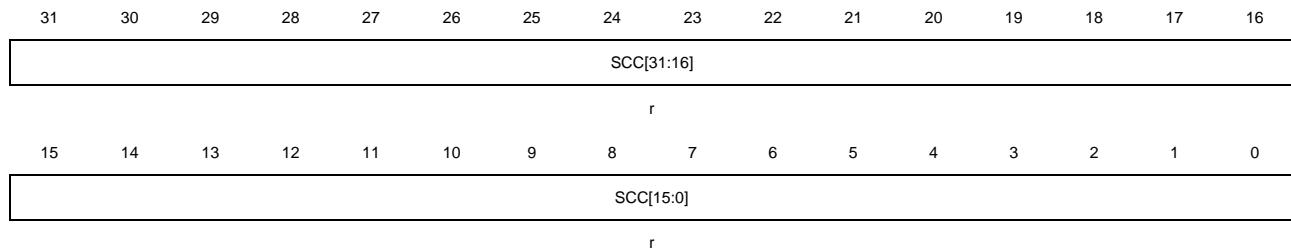
Bits	Fields	Descriptions
31:22	Reserved	Must be kept at reset value
21	TGFIM	Transmitted good frames interrupt mask bit 0: Unmask the interrupt when the TGF bit is set 1: Mask the interrupt when the TGF bit is set
20:16	Reserved	Must be kept at reset value
15	TGFMS CIM	Transmitted good frames more single collision interrupt mask bit 0: Unmask the interrupt when the TGFMS C bit is set 1: Mask the interrupt when the TGFMS C bit is set
14	TGFSCIM	Transmitted good frames single collision interrupt mask bit 0: Unmask the interrupt when the TGFSC bit is set 1: Mask the interrupt when the TGFSC bit is set
13:0	Reserved	Must be kept at reset value

### **23.4.28. MSC transmitted good frames after a single collision counter register (ENET\_MSC\_SCCNT)**

Address offset: 0x014C

Reset value: 0x0000 0000

This register counts the number of successfully transmitted frames after a single collision in Half-duplex mode.



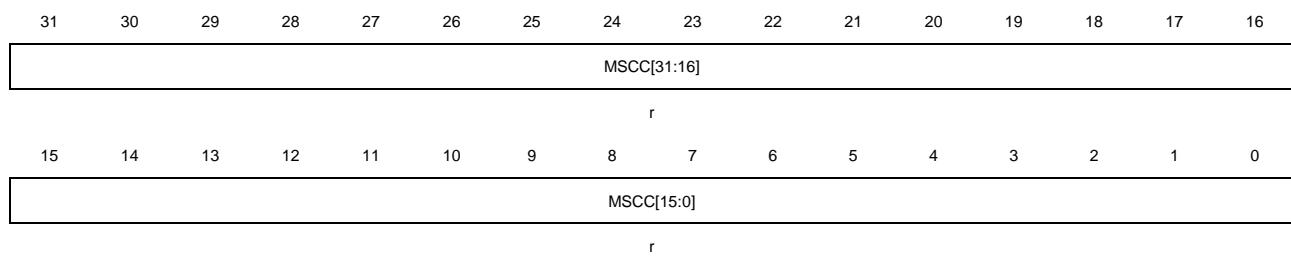
<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:0	SCC[31:0]	Transmitted good frames single collision counter bits These bits count the number of a transmitted good frames after only a single collision

### **23.4.29. MSC transmitted good frames after more than a single collision counter register (ENET\_MSC\_MSICCNT)**

Address offset: 0x0150

Reset value: 0x0000 0000

This register counts the number of successfully transmitted frames after more than one single collision in Half-duplex mode.



<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:0	MSCC[31:0]	Transmitted good frames more one single collision counter bits These bits count the number of a transmitted good frames after more than one single collision

### **23.4.30. MSC transmitted good frames counter register (ENET\_MSC\_TGFCNT)**

Address offset: 0x0168

Reset value: 0x0000 0000

This register counts the number of good frames transmitted.

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

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:0	TGF[31:0]	Transmitted good frames counter bits These bits count the number of transmitted good frames

### **23.4.31. MSC received frames with CRC error counter register (ENET\_MSC\_RFCECNT)**

Address offset: 0x0194

Reset value: 0x0000 0000

This register counts the number of frames received with CRC error.

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

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:0	RFCER[31:0]	Received frames with CRC error counter bits These bits count the number of receive frames with CRC error

### **23.4.32. MSC received frames with alignment error counter register (ENET\_MSC\_RFAECNT)**

Address offset: 0x0198

Reset value: 0x0000 0000

This register counts the number of received frames with alignment error.

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

Bits	Fields	Descriptions
31:0	RFAER[31:0]	Received frames alignment error counter bits These bits count the number of receive frames with alignment error

### 23.4.33. MSC received good unicast frames counter register (ENET\_MSC\_RGUFCNT)

Address offset: 0x01C4

Reset value: 0x0000 0000

This register counts the number of good unicast frames received.

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

Bits	Fields	Descriptions
31:0	RGUF[31:0]	Received good unicast frames counter bits These bits count the number of good unicast frames received.

### 23.4.34. PTP time stamp control register (ENET\_PTP\_TSCTL)

Address offset: 0x0700

Reset value: 0x0000 2000

This register configures the generation and updating for timestamp.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													MAFEN	CKNT[1:0]	
													rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MNMSEN	ETMSEN	IP4SEN	IP6SEN	ESEN	PFSV	SCROM	ARFSEN	Reserved	TMSARU	TMSITEN	TMSSTU	TMSSTI	TMSFCU	TMSEN	
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>
31:19	Reserved	Must be kept at reset value
18	MAFEN	MAC address filter enable for PTP frame 0: No effect 1: Enable MAC address1-3 to filter the PTP frame when received frame's type field is 0x88f7
17:16	CKNT[1:0]	Clock node type for time stamp 0x0: Type of ordinary clock 0x1: Type of boundary clock 0x2: Type of end-to-end transparent clock 0x3: Type of peer-to-peer transparent clock
15	MNMSEN	Received master node message snapshot enable This bit is valid only when CKNT=0x0 or 0x1. 0: Snapshot is only taken for slave node message 1: Snapshot is only take for master node message
14	ETMSEN	Received event type message snapshot enable 0: All type messages are taken snapshot except Announce, Management and Signaling message 1: Only event type messages (SYNC, Delay_Req, Pdelay_Req and Pdelay_Resp) are taken snapshot
13	IP4SEN	Received IPv4 snapshot enable 0: Do not take snapshot for IPv4 frame 1: Take snapshot for IPv4 frame
12	IP6SEN	Received IPv6 snapshot enable 0: Do not take snapshot for IPv6 frame 1: Take snapshot for IPv6 frame
11	ESEN	Received Ethernet snapshot enable 0: Do not take snapshot when received non type frame 1: Take snapshot when received non type frame
10	PFSV	PTP frame snooping version 0:Version 1(Revision of IEEE STD. 1588-2002/1588-2008) 1:Version 2(Revision of IEEE STD. 1588-2008)
9	SCROM	Subsecond counter rollover mode 0: Binary rollover mode. Subsecond rollovers when reach 0x7FFF_FFFF 1: Digital rollover mode. Subsecond rollovers when reach 0x3B9A_C9FF(0d999_999_999)
8	ARFSEN	All received frames snapshot enable 0: Not all received frames are taken snapshot

		1: All received frames are taken snapshot
5	TMSARU	<p>Time stamp addend register update bit</p> <p>This bit is cleared when the update is completed. This register bit must be read as zero before application set it.</p> <p>0: The timestamp addend register's contents are not updated to the PTP block for fine correction</p> <p>1: The timestamp addend register's contents are updated to the PTP block for fine correction</p>
4	TMSITEN	<p>Timestamp interrupt trigger enable bit</p> <p>0: Disable timestamp interrupt</p> <p>1: A timestamp interrupt is generated when the system time becomes greater than the value written in target time register.</p> <p><b>Note:</b> When the timestamp trigger interrupt generated, this bit is cleared</p>
3	TMSSTU	<p>Timestamp system time update bit</p> <p>Both the TMSSTU and TMSSTI bits must be read as zero before application set this bit</p> <p>0: The system time is maintained without any change</p> <p>1: The system time is updated (added to or subtracted from) with the value specified in the timestamp update (high and low) registers. It is cleared by hardware when the update finished.</p>
2	TMSSTI	<p>Timestamp system time initialize bit</p> <p>This bit must be read as zero before application set it.</p> <p>0: The system time is maintained without any change</p> <p>1: Initializing the system time with the value in timestamp update (high and low) registers. It is cleared by hardware when the initialization finished.</p>
1	TMSFCU	<p>Timestamp fine or coarse update bit</p> <p>0: The system timestamp uses the coarse method for updating</p> <p>1: The system timestamp uses the fine method for updating</p>
0	TMSEN	<p>Timestamp enable bit</p> <p>0: Disable timestamp function</p> <p>1: Enable timestamp function for transmit and receive frames</p> <p><b>Note:</b> After setting this to 1, application must initialize the system time.</p>

**Table 23-7. Supported time stamp snapshot with PTP register configuration**

CKNT (Bit 17:16)	MNMSEN (Bit 15)	ETMSEN (Bit 14)	Supported message type for snapshot
0X	X(*)	0	SYNC, Follow_Up, Delay_Req, Delay_Resp
0X	1	1	Delay_Req
0X	0	1	SYNC
10	X	0	SYNC, Follow_Up, Delay_Req, Delay_Resp

10	X	1	SYNC, Follow_Up
11	X	0	SYNC, Follow_Up, Delay_Req, Delay_Resp, Pdelay_Req, Pdelay_Resp
11	X	1	SYNC, Pdelay_Req, Pdelay_Resp

\*: X means do not care

### 23.4.35. PTP subsecond increment register (ENET\_PTP\_SSINC)

Address offset: 0x0704

Reset value: 0x0000 0000

This register configures the 8-bit value for the incrementing subsecond register. In coarse mode, this value is added to the system time every HCLK clock cycle. In fine mode, this value is added to the system time when the accumulator reaches overflow.

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										STMSSI[7:0]					
rw															

Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value
7:0	STMSSI[7:0]	System time subsecond increment bits  In every update operation, these bits are added to the subsecond value of system time.

### 23.4.36. PTP time stamp high register (ENET\_PTP\_TSH)

Address offset: 0x0708

Reset value: 0x0000 0000

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

Bits	Fields	Descriptions
31:0	STMS[31:0]	System time second bits  These bits show the current second of the system time.

### 23.4.37. PTP time stamp low register (ENET\_PTP\_TSL)

Address offset: 0x070C

Reset value: 0x0000 0000

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

Bits	Fields	Descriptions
31	STS	System time sign bit 0: Time value is positive 1: Time value is negative
30:0	STMSS[30:0]	System time subseconds bits These bits show the current subsecond of the system time with 0.46 ns accuracy if required accuracy is 20 ns.

### 23.4.38. PTP time stamp update high register (ENET\_PTP\_TSUH)

Address offset: 0x0710

Reset value: 0x0000 0000

This register configures the high 32-bit of the time to be written to, added to, or subtracted from the system time value. The timestamp update registers (high and low) initialize or update the system time maintained by the MAC core. Application must write both of these registers before setting the TMSSTI or TMSSTU bits in the timestamp control register.

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

Bits	Fields	Descriptions
31:0	TMSUS[31:0]	Time stamp update second bits These bits are used for initializing or adding/subtracting to second of the system time

### 23.4.39. PTP time stamp update low register (ENET\_PTP\_TSUL)

Address offset: 0x0714

Reset value: 0x0000 0000

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

Bits	Fields	Descriptions
31	TMSUPNS	<p>Timestamp update positive or negative sign bit</p> <p>When TMSSTI is set, this bit must be 0.</p> <p>0: Timestamp update value is added to system time</p> <p>1: Timestamp update value is subtracted from system time</p>
30:0	TMSUSS[30:0]	<p>Timestamp update subsecond bits</p> <p>These bits are used for initializing or adding/subtracting to subsecond of the system time</p>

### 23.4.40. PTP time stamp addend register (ENET\_PTP\_TSADDEND)

Address offset: 0x0718

Reset value: 0x0000 0000

This register value is used only in fine update mode for adjusting the clock frequency. This register value is added to a 32-bit accumulator in every clock cycle and the system time updates when the accumulator reaches overflow.

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

Bits	Fields	Descriptions
31:0	TMSA[31:0]	<p>Time stamp addend bits</p> <p>These registers contain a 32-bit time value which is added to the accumulator register to achieve time synchronization</p>

### **23.4.41. PTP expected time high register (ENET\_PTP\_ETH)**

Address offset: 0x071C

Reset value: 0x0000 0000

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

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:0	ETSH[31:0]	Expected time high bits These bits store the expected target second time.

### **23.4.42. PTP expected time low register (ENET\_PTP\_ETL)**

Address offset: 0x0720

Reset value: 0x0000 0000

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

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:0	ETSL[31:0]	Expected time low bits These bits store the expected target nanosecond time (signed).

### **23.4.43. PTP time stamp flag register (ENET\_PTP\_TSF)**

Address offset: 0x0728

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													TTM	TSSCO	

ro      ro

Bits	Fields	Descriptions
31:2	Reserved	Must be kept at reset value
1	TTM	Target time match bit 0: System time is not equal or greater than target time. 1: System time is equal or greater than target time <b>Note:</b> Reading ENET_PTP_TSF register will clear this bit.
0	TSSCO	Timestamp second counter overflow bit 0: Timestamp second counter has not overflowed 1: Timestamp second counter is greater than 0xFFFF FFFF

#### 23.4.44. PTP PPS control register (ENET\_PTP\_PPSCTL)

Address offset: 0x072C

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													PPSOFC[3:0]		

rw

Bits	Fields	Descriptions
31:4	Reserved	Must be kept at reset value
3:0	PPSOFC	PPS output frequency configure 0x0: 1Hz(Pulse width: 125ms for binary rollover, 100ms for digital rollover) 0x1: 2Hz(Pulse width: 50% duty cycle for binary rollover) 0x2: 4Hz(Pulse width: 50% duty cycle for binary rollover) .... 0xF: 32768( $2^{15}$ )Hz(Pulse width: 50% duty cycle for binary rollover) <b>Note:</b> If digital rollover is selected, only PPSOFC=0 is recommended.

### 23.4.45. DMA bus control register (ENET\_DMA\_BCTL)

Address offset: 0x1000

Reset value: 0x0002 0101

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved					MB	AA	FPBL	UIP	RXDP[5:0]					FB	
					rw	rw	rw	rw			rw			rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTPR[1:0]		PGBL[5:0]					DFM	DPSL[4:0]					DAB	SWR	
					rw	rw		rw			rw		rw	rw	rs

Bits	Fields	Descriptions
31:27	Reserved	Must be kept at reset value
26	MB	Mixed burst 0: AHB master interface only transfer fixed burst length with 16 and below 1: AHB master interface will transfer burst length greater than 16 with INCR <b>Note:</b> MB and FB should be and must be only one of bit is set.
25	AA	Address-aligned bit 0: Disable address-aligned 1: Enabled address-aligned. If the FB=1, all AHB interface address is aligned to the start address LS bits (bit 1 to 0). If the FB=0, the AHB interface first access address (accessing the data buffer's start address) is not aligned, but subsequent burst access addresses are aligned to the address
24	FPBL	Four times PGBL mode bit 0: The PGBL value programmed (bits [22:17] and bits [13:8]) for the DMA data number of beats to be transferred 1: Multiple the PGBL value programmed (bits [22:17] and bits [13:8]) four times for the DMA data number of beats to be transferred
23	UIP	Use independent PGBL bit 0: The PGBL value in bits [13:8] is applicable for both TxDMA and RxDMA engines 1: The RxDMA uses the RXDP[5:0] bits as burst length while the PGBL[5:0] is used by TxDMA
22:17	RXDP[5:0]	RxDMA PGBL bits If UIP=0, these bits are not valid. Only when UIP=1, these bits is configured for the maximum number of beats to be transferred in one RxDMA transaction. 0x01: max beat number is 1 0x02: max beat number is 2 0x04: max beat number is 4 0x08: max beat number is 8

---

		0x10: max beat number is 16 0x20: max beat number is 32 Other: Reserved
16	FB	<p>Fixed burst bit</p> <p>0: The AHB can use SINGLE and INCR burst transfer operations 1: The AHB can only use SINGLE, INCR4, INCR8 or INCR16 during start of normal burst transfers</p> <p><b>Note:</b> MB and FB should be and must be only one of bit is set.</p>
15:14	RTPR[1:0]	<p>RxDMA and TxDMA transfer priority ratio bits</p> <p>These bits indicate the access ratio between RxDMA and TxDMA.</p> <p>0x0: RxDMA : TxDMA = 1:1 0x1: RxDMA : TxDMA = 2:1 0x2: RxDMA : TxDMA = 3:1 0x3: RxDMA : TxDMA = 4:1</p> <p><b>Note:</b> This bit is valid only when the arbitration mode is Round-robin (DAB=0)</p>
13:8	PGBL[5:0]	<p>Programmable burst length bits</p> <p>These bits indicate the maximum number of beats to be transferred in one DMA transaction. When UIP=1, the PGBL value is only used for TxDMA. When UIP=0, the PGBL value is used for both TxDMA and RxDMA.</p> <p>0x01: max beat number is 1 0x02: max beat number is 2 0x04: max beat number is 4 0x08: max beat number is 8 0x10: max beat number is 16 0x20: max beat number is 32 Other: Reserved</p>
7	DFM	<p>Descriptor format mode</p> <p>0: Normal mode descriptor 1: Enhanced mode descriptor</p>
6:2	DPSL[4:0]	<p>Descriptor skip length bit</p> <p>These bits are valid only between two ring mode descriptors. They define the number of words (32-bit) to skip between two unchained descriptors. The address skipping starts from the end of current descriptor to the start of next descriptor.</p> <p>When DPSL value equals zero, the descriptor table is taken as contiguous by the DMA, in ring mode</p>
1	DAB	<p>DMA arbitration bit</p> <p>This bit indicates the arbitration mode between RxDMA and TxDMA.</p> <p>0: Round-robin mode and DMA access priority is given in RTPR 1: Fixed mode. RxDMA has higher priority than TxDMA</p>
0	SWR	Software reset bit

This bit can reset all core internal registers located in CLK\_TX and CLK\_RX.

It is cleared by hardware when the reset operation is complete in all clock domains.

**Note:** Application must make sure this bit is 0 before writing any MAC core registers.

0: Core and inner register are not in reset state

1: Reset all core internal registers

### 23.4.46. DMA transmit poll enable register (ENET\_DMA\_TPEN)

Address offset: 0x1004

Reset value: 0x0000 0000

This register is used by the application to make the TxDMA controller poll the transmit descriptor table. The TxDMA controller can go into suspend state because of an underflow error in a transmitted frame or the descriptor unavailable (DAV=0). Application can write any value into this register for attempting to re-fetch the current descriptor.

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

Bits	Fields	Descriptions
31:0	TPE[31:0]	<p>Transmit poll enable bits</p> <p>Writing to this register with any value makes DMA read the current descriptor address which is indicated in ENET_DMA_CTDADDR register. If the fetched current descriptor is available (DAV=1), DMA exits suspend state and resumes working. If the fetched current descriptor is unavailable (DAV=0), the DMA returns to suspend state again and the TBU bit in ENET_DMA_STAT register will be set.</p>

### 23.4.47. DMA receive poll enable register (ENET\_DMA\_RPEN)

Address offset: 0x1008

Reset value: 0x0000 0000

This register is used by the application to make the RxDMA controller poll the receive descriptor table. Writing to this register makes the RxDMA controller exit suspend state.

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

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

rw\_wt

Bits	Fields	Descriptions
31:0	RPE[31:0]	<p>Receive poll enable bits</p> <p>Writing to this register with any value makes DMA read the current descriptor address which is indicated in ENET_DMA_CRDADDR register. If the fetched current descriptor is available (DAV=1), DMA exits suspend state and resumes working. If the fetched current descriptor is unavailable (DAV=0), the DMA returns to suspend state again and the RBU bit in ENET_DMA_STAT register will be set.</p>

#### 23.4.48. DMA receive descriptor table address register (ENET\_DMA\_RDTADDR)

Address offset: 0x100C

Reset value: 0x0000 0000

This register points to the start of the receive descriptor table. The descriptor table is located in the physical memory space and must be word-aligned. This register can only be written when RxDMA controller is in stop state. Before starting RxDMA reception process, this register must be configured correctly.

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

rw

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

rw

Bits	Fields	Descriptions
31:0	SRT[31:0]	<p>Start address of receive table bits</p> <p>These bits indicate the start address of the receive descriptor table. SRT[1:0] are internally taken as zero so SRT[1:0] are read only.</p>

#### 23.4.49. DMA transmit descriptor table address register (ENET\_DMA\_TDTADDR)

Address offset: 0x1010

Reset value: 0x0000 0000

This register points to the start of the transmit descriptor table. The descriptor table is located in the physical memory space and must be word-aligned. This register can only be written when TxDMA controller is in stop state. Before starting TxDMA transmission process, this register must be configured correctly.

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

Bits	Fields	Descriptions
31:0	STT[31:0]	Start address of transmit table bits These bits indicate the start address of the transmit descriptor table. STT[1:0] are internally taken as zero so STT[1:0] are read only.

#### 23.4.50. DMA status register (ENET\_DMA\_STAT)

Address offset: 0x1014

Reset value: 0x0000 0000

This register contains all the status bits that the DMA controller recorded. Writing 1 to meaningful bits in this register clears them but writing 0 has no effect. Each bit (bits [16:0]) can be masked by masking the corresponding bit in the ENET\_DMA\_INTEN register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	TST	WUM	MSC	Reserved	EB[2:0]		TP[2:0]		RP[2:0]		NI				
r	r	r			r		r		r		r		r		rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AI	ER	FBE	Reserved	ET	RWT	RPS	RBU	RS	TU	RO	TJT	TBU	TPS	TS	
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
31:30	Reserved	Must be kept at reset value
29	TST	Timestamp trigger status bit This bit indicates a timestamp event occurred. It is cleared by application through clearing TMST bit. If the corresponding interrupt mask bit is reset, an interrupt is generated. 0: Timestamp event has not occurred 1: Timestamp event has occurred
28	WUM	WUM status bit This bit indicates a WUM event occurred. It is cleared when both two source event status bits are cleared. If the corresponding interrupt mask bit is reset, an interrupt is generated. 0: WUM event has not occurred 1: WUM event has occurred

27	MSC	MSC status bit  This bit indicates a MSC event occurred. It is cleared when all of event sources are cleared. If the corresponding interrupt mask bit is reset, an interrupt is generated. 0: MSC event has not occurred 1: MSC event has occurred
26	Reserved	Must be kept at reset value
25:23	EB[2:0]	Error bits status bit  When FBE=1, these bits decode the type of error that caused a bus response error on AHB bus.  EB[0] 1: Error during data transfer by TxDMA 0: Error during data transfer by RxDMA EB[1] 1: Error during read transfer 0: Error during write transfer EB[2] 1: Error during descriptor access 0: Error during data buffer access
22:20	TP[2:0]	Transmit process state bit  These bits decode the TxDMA state.  0x0: Stopped; Reset or Stop Transmit Command issued 0x1: Running; Fetching transmit transfer descriptor 0x2: Running; Waiting for status 0x3: Running; Reading Data from host memory buffer and queuing it to transmit buffer (TxFIFO) 0x4, 0x5: Reserved 0x6: Suspended; Transmit descriptor unavailable or transmit buffer underflow 0x7: Running; Closing transmit descriptor
19:17	RP[2:0]	Receive process state bit  These bits decode the RxDMA state.  0x0: Stopped: Reset or Stop Receive Command issued 0x1: Running: Fetching receive transfer descriptor 0x2: Reserved 0x3: Running: Waiting for receive packet 0x4: Suspended: Receive descriptor unavailable 0x5: Running: Closing receive descriptor 0x6: Reserved 0x7: Running: Transferring the receive packet data from receive buffer to host memory
16	NI	Normal interrupt summary  The NI bit is logical ORed of the following if the corresponding interrupt bit is enabled in the ENET_DMA_INTEN register: TS (ENET_DMA_STAT [0]): Transmit interrupt

TBU (ENET\_DMA\_STAT [2]): Transmit buffer unavailable

RS (ENET\_DMA\_STAT [6]): Receive interrupt

ER (ENET\_DMA\_STAT [14]): Early receive interrupt

**Note:** Each time when this bit is set, application must cleared its source bit by writing 1 to that bit.

15	AI	Abnormal interrupt summary bit  The AI bit is logical ORed of the following if the corresponding interrupt bit is enabled in the ENET_DMA_INTEN register:  TPS (ENET_DMA_STAT [1]):Transmit process stopped TJT (ENET_DMA_STAT [3]):Transmit jabber timeout RO (ENET_DMA_STAT [4]): Receive FIFO overflow TU (ENET_DMA_STAT [5]): Transmit underflow RBU (ENET_DMA_STAT [7]): Receive buffer unavailable RPS (ENET_DMA_STAT [8]): Receive process stopped RWT (ENET_DMA_STAT [9]): Receive watchdog timeout ET (ENET_DMA_STAT [10]): Early transmit interrupt FBE (ENET_DMA_STAT [13]): Fatal bus error <b>Note:</b> Each time when this bit is set, application must cleared its source bit by writing 1 to that bit.
14	ER	Early receive status bit  This bit is automatically cleared when the ENET_DMA_STAT [6] is set. 0: The first buffer has not been filled 1: The first buffer has filled with received frame
13	FBE	Fatal bus error status bit  This bit indicates a response error on AHB interface is occurred and the error type can be decoded by EB[2:0] bits. 0: Bus error has not occurred 1: A bus error occurred and the corresponding DMA stops all operations
12:11	Reserved	Must be kept at reset value
10	ET	Early transmit status bit 0: The frame to be transmitted has not fully transferred into the TxFIFO 1: The frame to be transmitted has fully transferred into the TxFIFO
9	RWT	Receive watchdog timeout status bit 0: No received a frame with a length greater than 2048 bytes 1: A frame with a length greater than 2048 bytes is received
8	RPS	Receive process stopped status bit 0: The receive process is not in stop state 1: The receive process is in stop state
7	RBU	Receive buffer unavailable status bit 0: The DAV bit in fetched next receive descriptor is set

		1: The DAV bit in fetched next receive descriptor is reset and RxDMA enters suspend state.
6	RS	<p>Receive status bit</p> <p>0: Frame reception has not completed</p> <p>1: Frame reception has completed</p>
5	TU	<p>Transmit underflow status bit</p> <p>0: Underflow error has not occurred during frame transmission</p> <p>1: The TxFIFO encountered an underflow error during frame transmission and entered suspend state</p>
4	RO	<p>Receive overflow status bit</p> <p>0: Receive overflow error has not occurred during frame reception</p> <p>1: The RxFIFO encountered an overflow error during frame reception. If a part of frame data has transferred to the memory, the overflow status in RDES0[11] is also set</p>
3	TJT	<p>Transmit jabber timeout status bit</p> <p>0: Transmit jabber timeout has not occurred during frame transmission</p> <p>1: The transmit jabber timer expired. The TxDMA controller cancels the current transmission and enters stop state. This also causes JT bit in TDES0 set.</p>
2	TBU	<p>Transmit buffer unavailable status bit</p> <p>0: The DAV bit in fetched next transmit descriptor is set</p> <p>1: The DAV bit in fetched next transmit descriptor is reset and TxDMA enters suspend state.</p>
1	TPS	<p>Transmit process stopped status bit</p> <p>0: The transmission is not in stop state</p> <p>1: The transmission is in stop state</p>
0	TS	<p>Transmit status bit</p> <p>This bit can only be set when both LSG and INTC are set in TDES0.</p> <p>0: Current frame transmission is not finished</p> <p>1: Current frame transmission is finished.</p>

### 23.4.51. DMA control register (ENET\_DMA\_CTL)

Address offset: 0x1018

Reset value: 0x0000 0000

This register configures both the transmitting and receiving operation modes and commands.  
This register should be written at last during the process of DMA initialization.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved					DTCERFD	RSFD	DAFRF	Reserved	TSFD	FTF	Reserved		TTHC[2]		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TTHC[1:0]	STE		Reserved				FERF	FUF	Reserved	RTHC[1:0]	OSF	SRE	Reserved		

rw            rw            rw            rw            rw            rs            rw            rw            rw            rw            rw            rw            rw            rw            rw

Bits	Fields	Descriptions
31:27	Reserved	Must be kept at reset value
26	DTCERFD	Dropping of TCP/IP checksum error frames disable bit 0: All error frames will be dropped when FERF=0 1: The received frame with only payload error but no other errors will not be dropped.
25	RSFD	Receive Store-and-Forward bit 0: The RxFIFO operates in Cut-Through mode. The forwarding threshold depends on the RTHC bits 1: The RxFIFO operates in Store-and-Forward mode. The RTHC bits are don't care and the frame forwarding starts after the whole frame has pushed into RxFIFO.
24	DAFRF	Disable flushing of received frames bit 0: The RxDMA flushes all frames because of unavailable receive descriptor 1: The RxDMA does not flush any frames even though receive descriptor is unavailable
23:22	Reserved	Must be kept at reset value
21	TSFD	Transmit Store-and-Forward bit 0: The TxFIFO operates in Cut-Through mode. The TTHC bits in ENET_DMA_CTL register defines the start popping time from TxFIFO 1: The TxFIFO operates in Store-and-Forward mode. Transmission on interface starts after the full frame has been pushed into the TxFIFO. The TTHC bits are don't care in this mode. <b>Note:</b> This bit can be changed when transmission is in stop state
20	FTF	Flush transmit FIFO bit This bit can be set by application to reset TxFIFO inner control register and logic. If set, all data in TxFIFO are flushed. It is cleared by hardware after the flushing operation is finish. <b>Note:</b> Before this bit is reset, this register (ENET_DMA_CTL) must not be written.
19:17	Reserved	Must be kept at reset value
16:14	TTHC[2:0]	Transmit threshold control bit These bits control the start transmitting byte threshold of the TxFIFO. When TSFD=1, these bits are ignored. 0x0: 64

		0x1: 128 0x2: 192 0x3: 256 0x4: 40 0x5: 32 0x6: 24 0x7: 16
13	STE	<p>Start/stop transmission enable bit</p> <p>0: The TxDMA controller will enter stop state after transmitting complete if the current frame is being transmitted. After complete transmitting, the next descriptor address will become current descriptor address for the address pointer. If the TxDMA controller is in suspend state, reset this bit make the controller entering stop state.</p> <p>1: The TxDMA controller will enter running state. TxDMA controller fetches current descriptor address for frame transmitting. Transmit descriptor's fetching can either from base address in ENET_DMA_TDTADDR register or from the pointer position when transmission was stopped previously. If the DAV bit of current descriptor is reset, TxDMA controller enters suspend state and the TBU bit will be set. This bit should be set after all other DMA registers have been configured otherwise the action of TxDMA is unpredictable.</p>
12:8	Reserved	Must be kept at reset value
7	FERF	<p>Forward error frames bit</p> <p>0: When Rx FIFO is in Cut-Through mode (RSFD=0), if frame error (CRC error, collision error, checksum error, watchdog timeout, overflow error) is detected before popping Rx FIFO data to memory, Rx FIFO drops this error frame. But if frame error is detected after popping Rx FIFO data to memory, Rx FIFO will not drop this frame data. When Rx FIFO is in Store-and-Forward mode, once frame error is detected during reception the Rx FIFO drops this frame.</p> <p>1: All frame received with error except runt error are forwarded to memory</p>
6	FUF	<p>Forward undersized good frames bit</p> <p>0: The Rx FIFO drops all frames whose length is less than 64 bytes. However, if this frame has already started forwarding (may due to lower value of receive threshold in Cut-Through mode), the whole frame will be forwarded.</p> <p>1: The Rx FIFO forwards received frame whose frame length is less than 64 bytes but without any other error.</p>
5	Reserved	Must be kept at reset value
4:3	RTHC[1:0]	<p>Receive threshold control bit</p> <p>These bits control the threshold bytes of the Rx FIFO.</p> <p><b>Note:</b> These bits are valid only when the RSFD=0 and are ignored when the RSFD=1.</p>

		0x0: 64
		0x1: 32
		0x2: 96
		0x3: 128
2	OSF	<p>Operate on second frame bit</p> <p>0: The TxDMA controller process the second transmit frame after the status of the first frame is written back to descriptor</p> <p>1: The TxDMA controller process the second transmit frame after pushed all first frame data into TxFIFO but before the status of the first frame is written back to descriptor</p>
1	SRE	<p>Start/stop receive enable bit</p> <p>0: The RxDMA controller will enter stop state after transfer complete if current received frame is transmitting to memory by RxDMA. After transfer complete, the next descriptor address in the receive table will become the current descriptor address when restart the RxDMA controller. Only RxDMA controller is in running state or suspend state, this bit can be reset by application.</p> <p>1: The RxDMA controller will enter running state. RxDMA controller fetches receive descriptor from receive descriptor table for receiving frames. The descriptor address can either from current address in the ENET_DMA_RDTADDR register or the address after previous frame stopped by application. If the DAV bit in fetched descriptor is reset, RxDMA controller will enter suspend state and RBU bit will be set. Setting this bit can only when RxDMA controller is in stop state or suspend state. This bit should be set after all other DMA registers have been configured otherwise the action of RxDMA is unpredictable.</p>
0	Reserved	Must be kept at reset value

### 23.4.52. DMA interrupt enable register (ENET\_DMA\_INTEN)

Address offset: 0x101C

Reset value: 0x0000 0000

This register configures the interrupts which are reflected in ENET\_DMA\_STAT register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															NIE
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AIE	ERIE	FBEIE	Reserved	ETIE	RWTIE	RPSIE	RBUIE	RIE	TUIE	ROIE	TJTIE	TBUIE	TPSIE	TIE	
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	NIE	<p>Normal interrupt summary enable bit</p> <p>0: A normal interrupt is disabled.</p> <p>1: A normal interrupt is enabled</p> <p>This bit enables the following bits:</p> <ul style="list-style-type: none"> <li>TS (ENET_DMA_STAT [0]): Transmit interrupt</li> <li>TBU (ENET_DMA_STAT [2]): Transmit buffer unavailable</li> <li>RS (ENET_DMA_STAT [6]): Receive interrupt</li> <li>ER (ENET_DMA_STAT [14]): Early receive interrupt</li> </ul>
15	AIE	<p>Abnormal interrupt summary enable bit</p> <p>0: An abnormal interrupt is disabled.</p> <p>1: An abnormal interrupt is enabled</p> <p>This bit enables the following bits:</p> <ul style="list-style-type: none"> <li>TPS (ENET_DMA_STAT [1]): Transmit process stopped</li> <li>TJT (ENET_DMA_STAT [3]): Transmit jabber timeout</li> <li>RO (ENET_DMA_STAT [4]): Receive FIFO overflow</li> <li>TU (ENET_DMA_STAT [5]): Transmit underflow</li> <li>RBU (ENET_DMA_STAT [7]): Receive buffer unavailable</li> <li>RPS (ENET_DMA_STAT [8]): Receive process stopped</li> <li>RWT (ENET_DMA_STAT [9]): Receive watchdog timeout</li> <li>ET (ENET_DMA_STAT [10]): Early transmit interrupt</li> <li>FBE (ENET_DMA_STAT [13]): Fatal bus error</li> </ul>
14	ERIE	<p>Early receive interrupt enable bit</p> <p>0: The early receive interrupt is disabled</p> <p>1: The early receive interrupt is enabled</p>
13	FBEIE	<p>Fatal bus error interrupt enable bit</p> <p>0: The fatal bus error enable interrupt is disabled</p> <p>1: The fatal bus error enable interrupt is enabled</p>
12:11	Reserved	Must be kept at reset value
10	ETIE	<p>Early transmit interrupt enable bit</p> <p>0: The early transmit interrupt is disabled</p> <p>1: The early transmit interrupt is enabled</p>
9	RWTIE	<p>Receive watchdog timeout interrupt enable bit</p> <p>0: The receive watchdog timeout interrupt is disabled</p> <p>1: The receive watchdog timeout interrupt is enabled</p>
8	RPSIE	<p>Receive process stopped interrupt enable bit</p> <p>0: The receive stopped interrupt is disabled</p> <p>1: The receive stopped interrupt is enabled</p>
7	RBUIE	<p>Receive buffer unavailable interrupt enable bit</p> <p>0: The receive buffer unavailable interrupt is disabled</p>

		1: The receive buffer unavailable interrupt is enabled
6	RIE	Receive interrupt enable bit 0: The receive interrupt is disabled 1: The receive interrupt is enabled
5	TUIE	Transmit underflow interrupt enable bit 0: The underflow interrupt is disabled 1: The underflow interrupt is enabled
4	ROIE	Receive overflow interrupt enable bit 0: The overflow interrupt is disabled 1: The overflow interrupt is enabled
3	TJTIE	Transmit jabber timeout interrupt enable bit 0: The transmit jabber timeout interrupt is disabled 1: The transmit jabber timeout interrupt is enabled
2	TBUIE	Transmit buffer unavailable interrupt enable bit 0: The transmit buffer unavailable interrupt is disabled 1: The transmit buffer unavailable interrupt is enabled
1	TPSIE	Transmit process stopped interrupt enable bit 0: The transmission stopped interrupt is disabled 1: The transmission stopped interrupt is enabled
0	TIE	Transmit interrupt enable bit 0: The transmit interrupt is disabled 1: The transmit interrupt is enabled

### 23.4.53. DMA missed frame and buffer overflow counter register (ENET\_DMA\_MFBOCNT)

Address offset: 0x1020

Reset value: 0x0000 0000

There are two counters designed in DMA controller for tracking the number of missed frames during receiving. The counter value can be read from this register for debug purpose.

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

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

31:28	Reserved	Must be kept at reset value
27:17	MSFA[10:0]	Missed frames by the application bits These bits indicate the number of frames dropped by RxFIFO
16	Reserved	Must be kept at reset value
15:0	MSFC[15:0]	Missed frames by the controller bits These bits indicate the number of frames missed by the RxDMA controller because of the unavailable receive buffer. Each time the RxDMA controller flushes one frame, this counter will plus 1.

#### 23.4.54. DMA receive state watchdog counter register (ENET\_DMA\_RSWDC)

Address offset: 0x1024

Reset value: 0x0000 0000

The watchdog counter value register for RS bit (ENET\_DMA\_STAT register) set after delay a configured time.

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								WDCFRS[7:0]							

rw

Bits	Fields	Descriptions
31:8	Reserved	Must be kept at reset value
7:0	WDCFRS[7:0]	Watchdog counter for receive status (RS) bit. These bits are only valid when DINTC (RXDES1) is set. When DINTC=1 and a frame is received, the RS bit will be set delay a time of WDCFRS*256 HCLK after receiving complete.

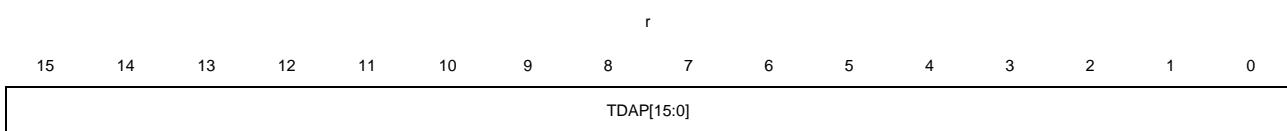
#### 23.4.55. DMA current transmit descriptor address register (ENET\_DMA\_CTDADDR)

Address offset: 0x1048

Reset value: 0x0000 0000

This register points to the start descriptor address of the current transmit descriptor read by the TxDMA controller.

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



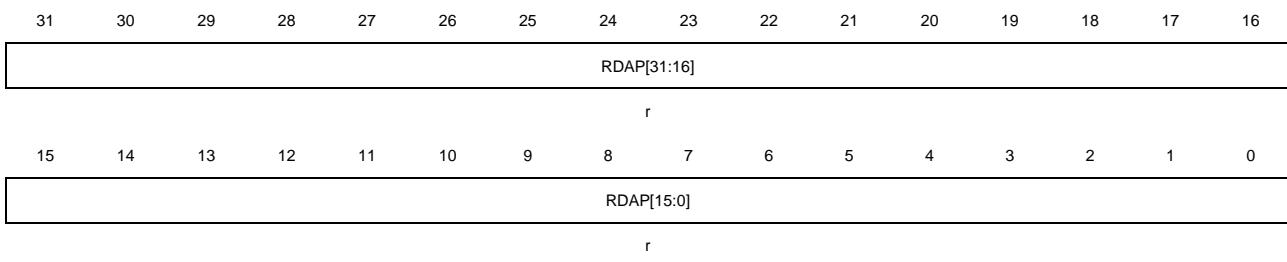
Bits	Fields	Descriptions
31:0	TDAP[31:0]	Transmit descriptor address pointer bits These bits are automatically updated by TxDMA controller during operation.

### 23.4.56. DMA current receive descriptor address register (ENET\_DMA\_CRDADDR)

Address offset: 0x104C

Reset value: 0x0000 0000

This register points to the start descriptor address of the current receive descriptor read by the RxDMA controller.



r

RDAP[15:0]

r

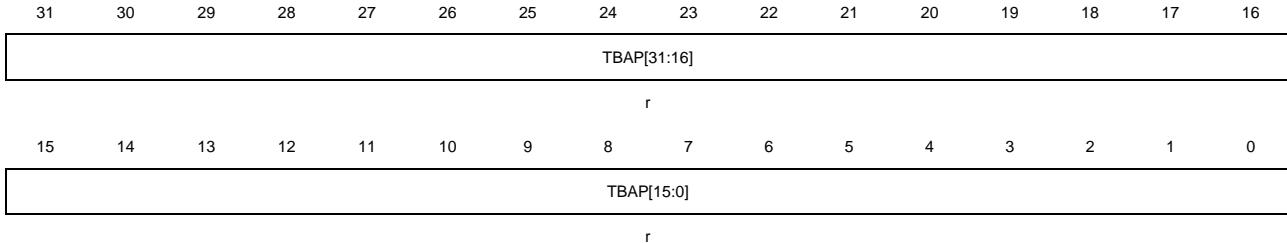
Bits	Fields	Descriptions
31:0	RDAP[31:0]	Receive descriptor address pointer bits These bits are automatically updated by RxDMA controller during operation.

### 23.4.57. DMA current transmit buffer address register (ENET\_DMA\_CTBADDR)

Address offset: 0x1050

Reset value: 0x0000 0000

This register points to the current transmit buffer address being read by the TxDMA controller.



r

TBAP[15:0]

r

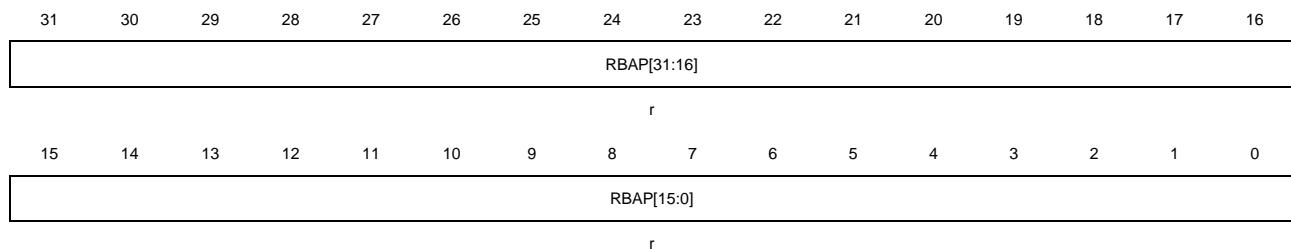
<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:0	TBAP[31:0]	Transmit buffer address pointer bits These bits are automatically updated by TxDMA controller during operation.

### 23.4.58. DMA current receive buffer address register (ENET\_DMA\_CRBADDR)

Address offset: 0x1054

Reset value: 0x0000 0000

This register points to the current receive buffer address being read by the RxDMA controller.



<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:0	RBAP[31:0]	Receive buffer address pointer bits These bits are automatically updated by RxDMA controller during operation.

## 24. Universal Serial Bus full-speed device interface (USBD)

The USBD is only available on GD32F303 series.

### 24.1. Overview

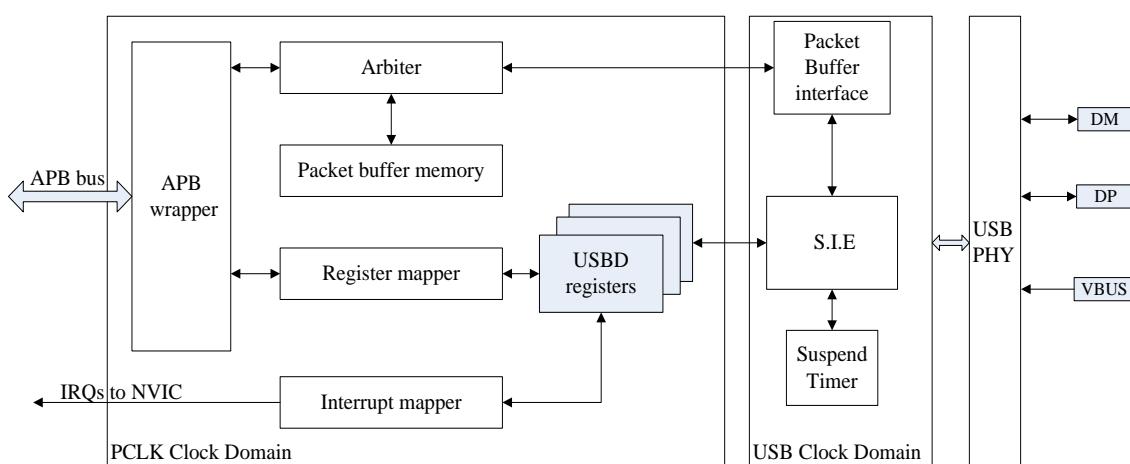
The Universal Serial Bus full-speed device interface (USBD) module provides a device solution for implementing a USB 2.0 full-speed compliant peripheral. It contains a full-speed internal USB PHY and no more external PHY chip is needed. USBD supports all the four types of transfer (control, bulk, interrupt and isochronous) defined in USB 2.0 protocol.

### 24.2. Main features

- USB 2.0 full-speed device controller.
- Support up to 8 configurable bidirectional endpoints.
- Support double-buffered bulk/isochronous endpoints.
- Support USB 2.0 Link Power Management.
- Each endpoint supports control, bulk, isochronous or interrupt transfer types (exclude endpoint 0, endpoint 0 only support control transfers).
- Support USB suspend/resume operations.
- Shared dedicated 512-byte SRAM used for data packet buffer with CAN.
- Integrated USB PHY.

### 24.3. Block diagram

**Figure 24-1. USBD block diagram**



## 24.4. Signal description

**Table 24-1. USBD signal description**

I/O port	Type	Description
VBUS	Input	Bus power port
DM	Input/Output	Differential D-
DP	Input/Output	Differential D+

**Note:** As soon as the USBD is enabled, these pins are connected to the USBD internal transceiver automatically.

## 24.5. Clock configuration

According to the USB standard definition, the USB full-speed module adopt fixed 48MHz clock. It is necessary to configure two clock for using USBD, one is the USB controller clock, its frequency must be configured to 48MHz, and the other one is the APB1 to USB interface clock which is also APB1 bus clock, its frequency can be above or below 48MHz.

**Note:** In order to meet the system requirements of packet buffer interface and USB data transfer rate, the frequency of the APB1 bus clock must be greater than 24MHz, so as to avoid data buffer overflow and underflow.

48MHz clock of USB controller can be generated by dividing MCU internal or external crystal oscillator by a programmable prescaler, then multiplicating the frequency through PLL.

- Regard two frequency division of 8MHz internal oscillator as the input of the PLL, then 12 frequencies doubling the clock.
- Regard 8MHz external oscillator as the input of the PLL, firstly frequency doubling, then adopt USB frequency divider to divide frequency.

When the USB clock is generated by external crystal, only 7 USB frequency prescaler can be used as 1, 1.5, 2, 2.5, 3, 3.5 and 4 (3, 3.5 and 4 cannot be used, because the frequency of GD32F30x MCU cannot reach above 120MHz). Thus, for obtaining 48MHz clock, PLL frequencies doubling can only be 48MHz, 72MHz, 96MHz and 120MHz.

**Note:** Regardless of using internal or external crystal oscillator to generate USB clock, the clock accuracy must reach  $\pm 500\text{ppm}$ . If the accuracy of the USB clock cannot meet the condition, data transfer may not conform to the requirements of the USB specification, and even it may cause USB not working directly.

## 24.6. Function overview

### 24.6.1. USB endpoints

USBD supports 8 USB endpoints that can be individually configured.

Each endpoint supports:

- Single/Double buffer (endpoint 0 can't use double buffer).
- One endpoint buffer descriptor.
- Programmable buffer starting address and buffer length.
- Configurable response to a packet.
- Control transfer (endpoint 0 only).

#### Endpoint buffer

The function of the device operation is to transfer a request in the memory image to and from the Universal Serial Bus. To efficiently manage USB endpoint communications, USBD implements a dedicated data packet buffer of 512-bytes SRAM memory accessed directly by the USB peripheral. It is mapped to the APB1 peripheral memory, from 0x4000 6000 to 0x4000 6400. The total capacity is 1KB, but USBD uses actually only 512 bytes for the bus width reason.

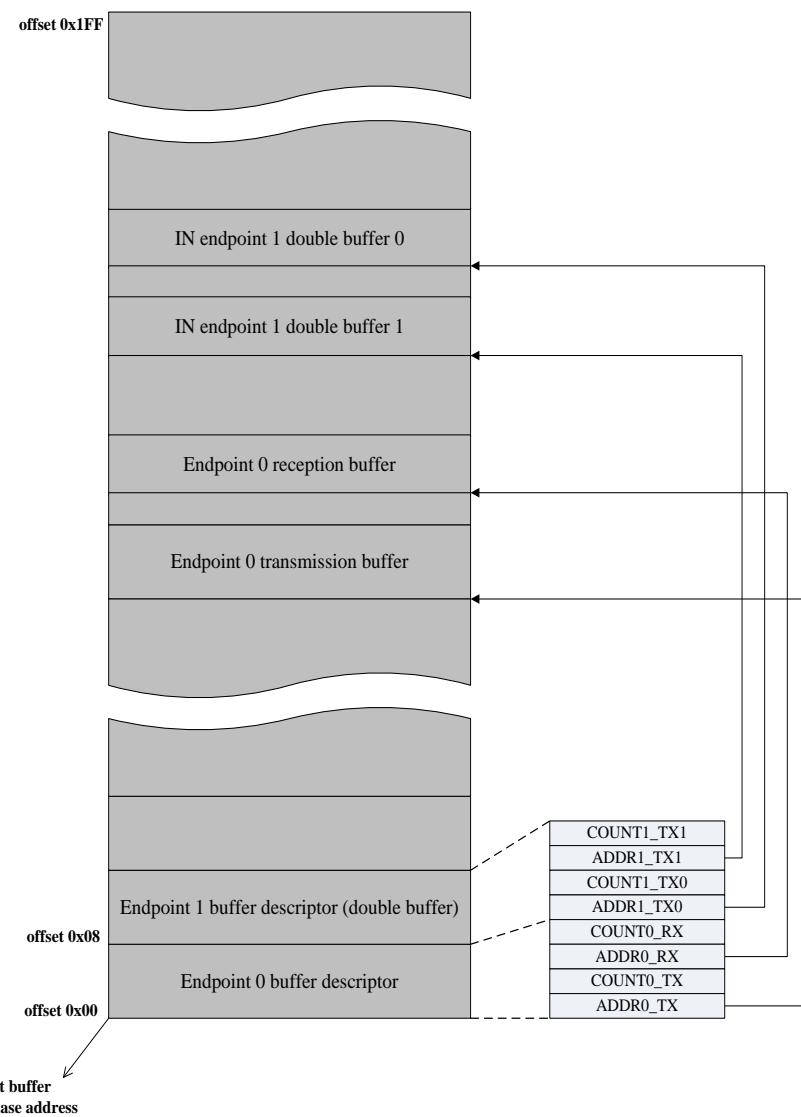
Each endpoint can be associated with one or two data packet buffers used to store the current data payload. The bidirectional endpoint has usually two buffers, one is used for transmission and the other one is for reception. The mono-directional endpoint only has one buffer for data operation.

**Note:** The USBD and CAN share the dedicated 512-byte SRAM memory.

#### Endpoint buffer descriptor table

USBD implements an endpoint buffer descriptor table which defines the buffer address and length and which is also located in the endpoint data packet buffer. The endpoint buffer descriptor is used as a communication port between the application firmware and the SIE in system memory. Every endpoint direction requires two 16-bit words buffer descriptor. Therefore, each table entry includes 4 16-bit words (Tx and Rx two direction) and is aligned to 8-byte boundary. When an endpoint is double-buffered, the SIE will use the two buffers in ping-pong operation mode. The endpoint buffer descriptor table is pointed to by the USBD endpoint buffer address register.

The relationship between endpoint buffer descriptor table entries and packet buffer areas is depicted in [Figure 24-2. an example with buffer descriptor table usage \(USBD\\_BADDR = 0\).](#)

**Figure 24-2. an example with buffer descriptor table usage (USBD\_BADDR = 0)**


**Note:** This figure is not drawn on the actual scale, and it is addressed through the USB bus 16-bit mode.

### Double-buffered endpoints

The double-buffered feature is used to improve bulk transfer performance. To implement the new flow control scheme, the USB peripheral should know which packet buffer is currently in use by the application software, so to be aware of any conflict. Since in the USBD\_EPxCS register, there are two data toggle bits (TX\_DTG and RX\_DTG) but only one is used by USBD for hardware data handling (due to the unidirectional constraint required by double-buffering feature), the other one can be used by the application software to show which buffer it is currently using. This new buffer flag is called software buffer bit (SW\_BUF). In [Table 24-2. Double-buffering buffer flag definition](#), the correspondence between USBD\_EPxCS register bits and DTG/SW\_BUF definition is explained.

**Table 24-2. Double-buffering buffer flag definition**

Buffer flag	Tx endpoint	Rx endpoint
DTG	TX_DTG (USBD_EPxCS bit 6)	RX_DTG (USBD_EPxCS bit 14)
SW_BUF	RX_DTG (USBD_EPxCS bit 14)	TX_DTG (USBD_EPxCS bit 6)

The DTG bit and the SW\_BUF bit are responsible for the flow control. When a transfer completes, the USB peripheral toggle the DTG bit; when the data have been copied, the application software need to toggle the SW\_BUF bit. Except for the first time, if the value of DTG bit is equal to the SW\_BUF's, the transfer will pause, and the host is NAK. When the two bits are not equal, the transfer resume.

**Table 24-3. Double buffer usage**

Endpoint Type	DTOG	SW_BUF	Packet buffer used by the USB peripheral	Packet buffer used by the application software
OUT	0	1	EPxRBADDR/EPxRBCNT buffer description table locations.	EPxTBADDR/EPxTBCNT buffer description table locations.
	1	0	EPxTBADDR/EPxTBCNT buffer description table locations.	EPxRBADDR/EPxRBCNT buffer description table locations.
IN	0	1	EPxTBADDR/EPxTBCNT buffer description table locations.	EPxRBADDR/EPxRBCNT buffer description table locations.
	1	0	EPxRBADDR/EPxRBCNT buffer description table locations.	EPxTBADDR/EPxTBCNT buffer description table locations.

### Endpoint memory requests arbitration

As the USBD is connected to the APB1 bus through an APB1 interface, so USB APB1 interface will accept memory requests coming from the APB1 bus and from the USB interface. The arbiter will resolve the conflicts by giving priority to APB1 accesses, while always reserving half of the memory bandwidth to complete all USB transfers. This time-duplex scheme implements a virtual dual-port SRAM that allows memory access, when an USB transaction is happening. Multiword APB1 transfers of any length are also allowed by this scheme.

## 24.6.2. Operation procedure

### USB transaction process

After the endpoint is configured and a transaction is required, the hardware will detect the token packet. When a token is recognized by the USBD, the data transfer is performed. When all the data has been transferred, the proper handshake packet over the USBD is generated or expected according to the direction of the transfer.

After the transaction process is completed, an endpoint-specific interrupt is generated. In the interrupt routine, the application can process it accordingly.

Transaction formatting is performed by the hardware, including CRC generation and checking.

Once the endpoint is enabled, endpoint control and status register, buffer address and COUNT field should not be modified by the application software. When the data transfer operation is completed, notified by a STIF interrupt event, they can be accessed again to re-enable a new operation.

### IN transaction

When a configured and valid endpoint receives an IN token packet, it will send the data packet to the host. If the endpoint is not valid, a NAK or STALL handshake is sent according to the endpoint status.

In the data packet transfer process, a configured data PID will be sent firstly, then the actual data in endpoint buffer memory is loaded into the output shift register to be transmitted. After the data are sent, the computed CRC will be sent by hardware.

When receiving the ACK sent from the host, then the USB peripheral will toggle the data PID and set the endpoint status to be NAK. At the same time, the successful transfer interrupt will be triggered. In the interrupt service routine, application fill the data packet memory with data, start next transfer by re-enable the endpoint by setting the endpoint status VALID.

### OUT and SETUP transaction

USBD handle OUT and SETUP tokens in similar way, the difference details about SETUP packets would be shown in the following section about control transfer.

After the received endpoint is configured and enabled, host will send OUT/SETUP token to the device. When receiving the token, USBD will access the endpoint buffer descriptor to initialize the endpoint buffer address and length. Then the received data bytes subsequently are packed in words (LSB mode) and transferred to the endpoint buffer. When detecting the end of data packet, the computed CRC and received CRC are compared. If no errors occur, an ACK handshake packet is sent to the host.

When the transaction is completed correctly, USBD will toggle the data PID and set the endpoint status to be NAK. Then the endpoint successful transfer interrupt will be triggered by hardware. In the interrupt service routine, the application can get the transaction type and read the received data from the endpoint buffer. After the received data is processed, the application should initiate further transactions by setting the endpoint status valid.

If any error happens during reception, the USBD set the error interrupt bit and still copy data into the packet memory buffer, but will not send the ACK packet. The USBD itself can recover from reception errors and continue to handle next transfer. The USBD never override outside the data buffer, which is controlled by the internal register. The received 2-byte CRC is also copied to the packet memory buffer, immediately following data bytes. If the length of data is

greater than actually allocated length, the excess data are not copied. This is a buffer overrun situation. A STALL handshake is sent, and this transaction fails.

If an addressed endpoint is not valid, a NAK or STALL handshake packet is sent instead of the ACK, according to the endpoint status and no data is written to the endpoint data buffers.

### Control transfers

Control transfers require that a SETUP transaction be started from the host to a device to describe the type of control access that the device should perform. The SETUP transaction is followed by zero or more control DATA transactions that carry the specific information for the requested access. Finally, a STATUS transaction completes the control transfer and allows the endpoint to return the status of the control transfer to the client software. After the STATUS transaction for a control transfer is completed, the host can advance to the next control transfer for the endpoint.

USBD always use endpoint 0 in two directions as default control endpoint to handle control transfers. It is aware of the number and direction of data stages by interpreting the contents of SETUP transaction, and is required to set the unused direction endpoint 0 status to STALL except the last data stage.

At the last data stage, the application software set the opposite direction endpoint 0 status to NAK. This will keep the host waiting for the completion of the control operation. If the operation completes successfully, the software will change NAK to VALID, otherwise to STALL. If the status stage is an OUT, the STATUS\_OUT bit should be set, so that a status transaction with non-zero data will be answered STALL to indicate an error happen.

According to USB specification, device isn't allowed to abort current command and then start new command, so that device must answer a SETUP packet with an ACK handshake packet, not with a NAK or STALL handshake packet.

When the configured control endpoint 0 receives a SETUP token, the USBD accepts the data, performing the required data transfers and sends back an ACK handshake. If there is unsuccessfully handling data transfer about previously issued request, the USB discard SETUP token and regard current condition as error, and then urge the host to send the request token again.

### Isochronous transfers

Isochronous transfers can guarantee constant data rate and bounded latency, but do not support data retransmission in response to errors on the bus. Consequently, the isochronous transaction does not have a handshake phase, and have no ACK packet after the data packet. Data toggling is not supported, and DATA0 PID is only used to start a data packet.

The isochronous endpoint status only can be set DISABLED and VALID, any other value is illegal. The application software can implement double-buffering to improve performance. By swapping transmission and reception data packet buffer on each transaction, the application software can copy the data into or out of a buffer, at the same time the USB peripheral handle

the data transmission or reception of data in another buffer. The DTOG bit indicates which buffer that the USB peripheral is currently using.

The application software initializes the DTOG according to the first buffer to be used. At the end of each transaction, the RX\_ST or TX\_ST bit is set, depending on the enabled direction regardless of CRC errors or buffer-overrun conditions (if errors occur, the ERRIF bit will be set). At the same time, The USB peripheral will toggle the DTOG bit, but will not affect the STAT bit.

### 24.6.3. USB events and interrupts

Each USB action is always initiated by the application software, driven by one USB interrupt or event. After system reset, the application needs to wait for a succession of USB interrupts and events.

#### Reset events

##### System and power-on reset

Upon system and power-on reset, the application software should first provide all required clock to the USB module and interface, then de-assert its reset signal so as to be able to access its registers, last switch on the analog part of the device related to the USB transceiver.

The USB firmware should do as follows:

- Reset CLOSE bit in USBD\_CTL register.
- Wait for the internal reference voltage to be stable.
- Clear SETRST bit in USBD\_CTL register.
- Clear the USBD\_INTF register to remove the spurious pending interrupt and then enable other unit.

##### USB reset (RESET interrupt)

When this event occurs, the USB peripheral status is the same as the moment system reset.

The USB firmware should do as follows:

- Set USBEN bit in ADDR register to enable USB module in 10ms.
- Initialize the USBD\_EP0CS register and its related packet buffers.

#### Suspend and resume events

The USB module can be forced to place in low-power mode (SUSPEND mode) by writing in the USB control register (USBD\_CTL) whenever required. At this time, all static power consumption is avoided and the USB clock can be slowed down or stopped. It will be resumed when detect activity at the USB bus while in low-power mode.

The USB protocol insists on power management by the USB device. This becomes even more important if the device draws power from the bus (bus-powered device). The following constraints should be met by the bus-powered device.

- A device in the non-configured state should draw a maximum of 100mA from the USB bus.
- A configured device can draw only up to what is specified in the Max Power field of the configuration descriptor. The maximum value is 500mA.
- A suspended device should draw a maximum of 500uA.

A device will go into the suspend state if there is no activity on the USB bus for more than 3ms. A suspended device wakes up, if RESUME signaling is detected.

USBD also supports software initiated remote wakeup. To initiate remote wakeup, the application software must enable all clocks and clear the suspend bit after MCU is waked up. This will cause the hardware to generate a remote wakeup signal upstream.

Setting the SETSPS bit to 1 enables the suspend mode, and it will disable the check of SOF reception. Setting the LOWM bit to 1 will shut down the static power consumption in the analog USB transceivers, but the RESUME signal is still able to be detected.

### Link Power Management (LPM) level L1

In order to optimize power consumption in SUSPEND/RESUME state, USB 2.0 has achieved Link Power Management (LPM). LPM includes 4 states from L0 to L3. LPM L1 state (sleep state) is the new power management state.

A device will go into the L1 state if the host sends a successful LPM transaction. L1 does not impose any specific power draw requirements (from VBUS) on the attached device.

For more details, please refer to [USB2\\_LinkPowerManagement\\_ECN](#).

### USB Interrupts

USBD has three interrupts: low-priority interrupt, high-priority interrupt and wakeup interrupt. Software can configure these interrupts to route the interrupt condition to these entries in the NVIC table. An interrupt will be generated when both the interrupt status bit and the corresponding interrupt enable bit are set. The interrupt status bit is set by hardware if the interrupt condition occurs (irrespective of the interrupt enable bit).

- Low-priority interrupt (Channel 20): triggered by all USB events.
- High-priority interrupt (Channel 19): triggered only by a correct transfer event for isochronous and double-buffer bulk transfer.
- Wakeup interrupt (Channel 42): triggered by the wakeup events.

#### 24.6.4. Operation guide

This section describes the operation guide for USBD.

##### USBD register initialization sequence

1. Clear the CLOSE bit in USBD\_CTL register, then clear the SETRST bit.

2. Clear USBD\_INTF register to remove any spurious pending interrupt.
3. Program USBD\_BADDR register to set endpoint buffer base address.
4. Set USBD\_CTL register to enable interrupts.
5. Wait for the reset interrupt (RSTIF).
6. In the reset interrupt, initialize default control endpoint 0 to start enumeration process and program USBD\_BADDR to set the device address to 0 and enable USB module function.
7. Configure endpoint 0 and prepare to receive SETUP packet.

### Endpoint initialization sequence

1. Program USBD\_EPxBTADDR or USBD\_EPxRBADDR registers with transmission or reception data buffer address.
2. Program the EP\_CTL and EP\_KCTL bits in USBD\_EPxCS register to set endpoint type and buffer kind according to the endpoint usage.
3. If the endpoint is a single buffer endpoint:
  - 1) Initialize the endpoint data toggle bit by programming the TX\_DTG or RX\_DTG bit in USBD\_EPxCS register, but endpoint 0 needs to set them to 1 and 0 respectively for control transfer.
  - 2) Configure endpoint status by programming the TX\_STA bit or RX\_STA bit in USBD\_EPxCS register, but both of them are set to '10 (NAK) if use endpoint 0 to initialize the control transfer.

If the endpoint is a double buffer endpoint:

- 1) Both transmission and reception toggle fields need to be programmed. If the endpoint is a Tx endpoint, clear the TX\_DTG and RX\_DTG bit in USBD\_EPxCS register, or if endpoint is a Rx endpoint, it needs to toggle TX\_DTG bit.
- 2) Program USBD\_EPxBTCNT and USBD\_EPxRBCNT register to set transfer data bit count.
- 3) Endpoint transmission and reception status both need to be configured. If the endpoint is a Tx endpoint, set the TX\_STA bit to be NAK and RX\_STA bit to be DISABLED, or the endpoint is a Rx endpoint, set the RX\_STA bit to be VALID and TX\_STA bit to be DISABLED.

### SETUP and OUT data transfers

1. Program USBD\_EPxRBCNT register to set BLKSIZ and EPRCNT filed, these filed defines the endpoint buffer length.
2. Configure the endpoint status to be VALID to enable the endpoint to receive data by programming USBD\_EPxCS register.

3. Wait for successful transfer interrupt (STIF).
4. In the interrupt handler, application can get the transaction type by reading the STEUP bit in USBD\_EPxCS register. Then application will read the data payload from the endpoint data buffer with the start address defined in USBD\_EPxRBAR register. Last application will interpret the data and process the corresponding transaction.

## IN data transfers

1. Program USBD\_EPxTBCNT register to set EPTXCNT filed, this filed defines the endpoint buffer length.
2. Configure the endpoint status to be VALID to enable the endpoint to transmit data by programming USBD\_EPxCS register.
3. Wait for successful transfer interrupt (STIF).
4. In the interrupt handler, application needs to update user buffer length and location pointer. Then application fill the endpoint buffer with user buffer data. Last application will configure the endpoint status to be VALID to start next transfer.

## 24.7. Registers definition

USBD base address: 0x4000 5C00

### 24.7.1. USBD control register (USBD\_CTL)

Address offset: 0x40

Reset value: 0x0003

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STIE	PMOUIE	ERRIE	WKUPIE	SPSIE	RSTIE	SOFIE	ESOFIE	L1REQIE	Reserved	L1RSRE Q	RSREQ	SETSPS	LOWM	CLOSE	SETRST

rw      rw

Bits	Fields	Descriptions
		Successful transfer interrupt enables.
15	STIE	0: Successful transfer interrupt disabled. 1: Interrupt generated when STIF bit in USBD_INTF register is set.
		Packet memory overrun/underrun interrupt enable.
14	PMOUIE	0: No interrupt generated when packet memory overrun / underrun. 1: Interrupt generated when PMOUIF bit in USBD_INTF register is set.
		Error interrupt enable.
13	ERRIE	0: Error interrupt disabled 1: Interrupt generated when ERRIF bit in USBD_INTF register is set.
		Wakeup interrupt enable
12	WKUPIE	0: Wakeup interrupt disabled 1: Interrupt generated when WKUPIF bit in USBD_INTF register is set.
		Suspend state interrupt enable
11	SPSIE	0: Suspend state interrupt disabled 1: Interrupt generated when SPSIF bit in USB_IFR register is set.
		USB reset interrupt enable.
10	RSTIE	0: USB reset interrupt disabled 1: Interrupt generated when RSTIF bit in USBD_INTF register is set.
		Start of frame interrupt enable
9	SOFIE	0: Start of frame interrupt disabled

		1: Interrupt generated when SOFIF bit in USBD_INTF register is set.
		Expected start of frame interrupt enable
8	ESOFIE	0: Expected start of frame interrupt disabled 1: Interrupt generated when ESOFIF bit in USBD_INTF register is set.
		LPM L1 state request interrupt enable
7	L1REQIE	0: LPM L1 state request interrupt disabled 1: Interrupt generated when L1REQ bit in USBD_INTF register is set.
		Must be kept at reset value
6	Reserved	LPM L1 resume request
5	L1RSREQ	MCU can set this bit to send a LPM L1 resume signal to the host. After the signaling ends, this bit is cleared by hardware.  Resume request
		The software set a resume request to the USB host, and the USB host should drive the resume sequence according the USB specifications
4	RSREQ	0: No resume request 1: Send resume request.  Set suspend
		The software should set suspend state when SPSIF bit in USBD_INTF register is set.
3	SETSPS	0: Not set suspend state. 1: Set suspend state.  Low-power mode
		When set this bit, the USB goes to low-power mode at suspend state. If resume from suspend state, the hardware reset this bit.
2	LOWM	0: No effect 1: Go to low-power mode at suspend state.  Close state
		When this bit is set, the USBD goes to close state, and completely close the USBD and disconnected from the host.
1	CLOSE	0: Not in close state 1: In close state.  Set reset
0	SETRST	When this bit is set, the USBD peripheral should be reset. 0: No reset

1: A reset generated.

### 24.7.2. USBD interrupt flag register (USBD\_INTF)

Address offset: 0x44

Reset value: 0x0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STIF	PMOUIF	ERRIF	WKUPIF	SPSIF	RSTIF	SOFIF	ESOFIF	L1REQ	Reserved	DIR		EPNUM[3:0]			

rc\_w0      rc\_w0      rc\_w0      rc\_w0      rc\_w0      rc\_w0      rc\_w0      rc\_w0      rc\_w0      r      r

Bits	Fields	Descriptions
15	STIF	Successful transfer interrupt flag This bit set by hardware when a successful transaction completes
14	PMOUIF	Packet memory overrun/underrun interrupt flag This bit set by hardware to indicate that the packet memory is inadequate to hold transfer data. The software writes 0 to clear this bit.
13	ERRIF	Error interrupt flag This bit set by hardware when an error happens during transaction. The software writes 0 to clear this bit.
12	WKUPIF	Wakeup interrupt flag This bit set by hardware in the SUSPEND state to indicate that activity is detected. The software writes 0 to clear this bit.
11	SPSIF	Suspend state interrupt flag When no traffic happen in 3ms, hardware set this bit to indicate a SUSPEND request. The software writes 0 to clear this bit.
10	RSTIF	USB reset interrupt flag Set by hardware when the USB RESET signal is detected. The software writes 0 to clear this bit.
9	SOFIF	Start of frame interrupt flag Set by hardware when a new SOF packet arrives, The software writes 0 to clear this bit.
8	ESOFIF	Expected start of frame interrupt flag Set by the hardware to indicate that a SOF packet is expected but not received. The software writes 0 to clear this bit.
7	L1REQ	Set by the hardware when LPM L1 transaction is successfully received and acknowledged. The software writes 0 to clear this bit.

---

6:5	Reserved	Must be kept at reset value
		Direction of transaction
4	DIR	Set by the hardware to indicate the direction of the transaction 0: IN type 1: OUT type
3:0	EPNUM[3:0]	Endpoint Number Set by the hardware to identify the endpoint which the transaction is directed to

### 24.7.3. USBD status register (USBD\_STAT)

Address offset: 0x48

Reset value: 0x0XXX where X is undefined

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RX_DP	RX_DM	LOCK		SOFLN[1:0]						FCNT[10:0]					
r	r	r	r							r					

Bits	Fields	Descriptions
15	RX_DP	Receive data + line status Represent the status on the DP line
14	RX_DM	Receive data - line status Represent the status on the DM line
13	LOCK	Locked the USB Set by the hardware indicate that at least two consecutive SOF have been received
12:11	SOFLN[1:0]	SOF lost number Increment every ESOFIF happens by hardware Cleared once the reception of SOF
10:0	FCNT[10:0]	Frame number counter The Frame number counter incremented every SOF received.

### 24.7.4. USBD device address register (USBD\_DADDR)

Address offset: 0x4C

Reset value: 0x0000

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

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

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
15:8	Reserved	Must be kept at reset value
		USB device enable
		Set by software to enable the USB device
7	USBEN	0: The USB device disabled. No transactions handled. 1: The USB device enabled.
		USBD device address
6:0	USBDAR[6:0]	After bus reset, the address is reset to 0x00. If the enable bit is set, the device will respond on packets for function address DEV_ADDR

#### 24.7.5. USBD buffer address register (USBD\_BADDR)

Address offset: 0x50

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
BAR[12:0]														Reserved	
rw															

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
15:3	BAR[12:0]	Buffer address Start address of the allocation buffer(512byte on-chip SRAM), used for buffer descriptor table, packet memory
2:0	Reserved	Must be kept at reset value

#### 24.7.6. USBD endpoint x control and status register (USBD\_EPxCS), x=[0..7]

Address offset: 0x00 to 0x1C

Reset value: 0x0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RX_ST	RX_DTG	RX_STA[1:0]	SETUP	EP_CTL[1:0]	EP_KCTL	TX_ST	TX_DTG	TX_STA[1:0]						EP_ADDR[3:0]	
rc_w0	t	t	r	rw	rw	rc_w0	t	t						rw	

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
15	RX_ST	Reception successful transferred

		Set by hardware when a successful OUT/SETUP transaction complete Cleared by software by writing 0
14	RX_DTG	<p>Reception data PID toggle</p> <p>This bit represent the toggle data bit (0=DATA0,1=DATA1)for non-isochronous endpoint</p> <p>Used to implement the flow control for double-buffered endpoint</p> <p>Used to swap buffer for isochronous endpoint</p>
13:12	RX_STA[1:0]	<p>Reception status bits</p> <p>Toggle by writing 1 by software</p> <p>Remain unchanged by writing 0</p> <p>Refer to the table below</p>
11	SETUP	<p>Setup transaction completed</p> <p>Set by hardware when a SETUP transaction completed.</p>
10:9	EP_CTL[1:0]	<p>Endpoint type control</p> <p>Refer to the table below</p>
8	EP_KCTL	<p>Endpoint kind control</p> <p>The exact meaning depends on the endpoint type</p> <p>Refer to the table below</p>
7	TX_ST	<p>Transmission successful transfer</p> <p>Set by hardware when a successful IN transaction complete</p> <p>Clear by software</p>
6	TX_DTG	<p>Transmission data PID toggle</p> <p>This bit represent the toggle data bit (0=DATA0,1=DATA1)for non-isochronous endpoint</p> <p>Used to implement the flow control for double-buffered endpoint</p> <p>Used to swap buffer for isochronous endpoint</p>
5:4	TX_STA[1:0]	<p>Status bits, for transmission transfers</p> <p>Refer to the table below</p>
3:0	EP_ADDR	<p>Endpoint address</p> <p>Used to direct the transaction to the target endpoint</p>

**Table 24-4. Reception status encoding**

<b>RX_STA[1:0]</b>	<b>Meaning</b>
00	DISABLED: ignore all reception requests of this endpoint
01	STALL: STALL handshake status
10	NAK: NAK handshake status
11	VALID: enable endpoint for reception

**Table 24-5. Endpoint type encoding**

EP_CTL[1:0]	Meaning
00	BULK: bulk endpoint
01	CONTROL: control endpoint
10	ISO: isochronous endpoint
11	INTERRUPT: interrupt endpoint

**Table 24-6. Endpoint kind meaning**

EP_KCTL[1:0]		EP_KCTL Meaning
00	BULK	DBL_BUF
01	CONTROL	STATUS_OUT

**Table 24-7. Transmission status encoding**

TX_STA[1:0]	Meaning
00	DISABLED: ignore all transmission requests of this endpoint
01	STALL: STALL handshake status
10	NAK: NAK handshake status
11	VALID: enable endpoint for transmission

#### 24.7.7. USBD endpoint x transmission buffer address register

##### (USBD\_EPxTBADDR), x can be in [0..7]

Address offset: [USBD\_BADDR] + x \* 16

USB local address: [USBD\_BADDR] + x \* 8

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	
EPTXBAR[15:1]															EPTXBA R[0]	
rw															rw	

Bits	Fields	Descriptions
15:1	EPTXBAR[15:1]	Endpoint transmission buffer address Start address of the packet buffer containing data to be sent when receive next IN token
0	EPTXBAR[0]	Must be set to 0

#### 24.7.8. USBD endpoint x transmission buffer byte count register

##### (USBD\_EPxTBCNT), x can be in [0..7]

Address offset: [USBD\_BADDR] + x \* 16 + 4

USB local Address: [USBD\_BADDR] + x \* 8 + 2

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								EPTXCNT[9:0]							

rw

Bits	Fields	Descriptions
15:10	Reserved	Must be kept at reset value
9:0	EPTXCNT[9:0]	Endpoint transmission byte count The number of bytes to be transmitted at next IN token

#### 24.7.9. **USBD endpoint x reception buffer address register (USBD\_EPxRBADDR), x can be in [0..7]**

Address offset: [USBD\_BADDR] + x \* 16 + 8

USB local Address: [USB\_BADDR] + x \* 8 + 4

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

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

rw

rw

Bits	Fields	Descriptions
15:1	EPRBAR[15:1]	Endpoint reception buffer address Start address of packet buffer containing the data received by the endpoint at the next OUT/SETUP token
0	EPRBAR[0]	Must be set to 0

#### 24.7.10. **USBD endpoint x reception buffer byte count register (USBD\_EPxRBCNT), x can be in [0..7]**

Address offset: [USBD\_BADDR] + x \* 16 + 12

USB local Address: [USBD\_BADDR] + x \* 8 + 6

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
BLKSIZ	BLKNUM[4:0]					EPRCNT[9:0]									

rw

rw

r

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
15	BLKSIZ	Block size 0: block size is 2 bytes 1: block size is 32 bytes
14:10	BLKNUM[4:0]	Block number The number of blocks allocated to the packet buffer
9:0	EPRCNT[9:0]	Endpoint reception byte count The number of bytes to be received at next OUT/SETUP token

#### 24.7.11. USBD LPM control and status register (USBD\_LPMCS)

Address offset: 0x54

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				BLSTAT[3:0]				r		r		rw		rw	

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
15:8	Reserved	Must be kept at reset value
7:4	BLSTAT[3:0]	bLinkState value  This field contain the bLinkState value received with last ACKed LPM token.
3	REMWK	bRemoteWake value  This bit contains the bRemoteWake value received with last ACKed LPM token
2	Reserved	Must be kept at reset value
1	LPMACK	LPM token acknowledge enable  0: the valid LPM token will be NYETed. 1: the valid LPM token will be ACKed.  The NYET/ACK will be returned only on a successful LPM transaction: No errors in both the EXT token and the LPM token (else ERROR). A valid bLinkState = 0001B (L1) is received (else STALL)
0	LPMEN	LPM support enable  This bit is set by the software to enable the LPM support within the USB device. If this bit is set to 0, no LPM transactions are handled.

## 25. Universal serial bus full-speed interface (USBFS)

The USBFS is available on GD32F305 and GD32F307 series.

### 25.1. Overview

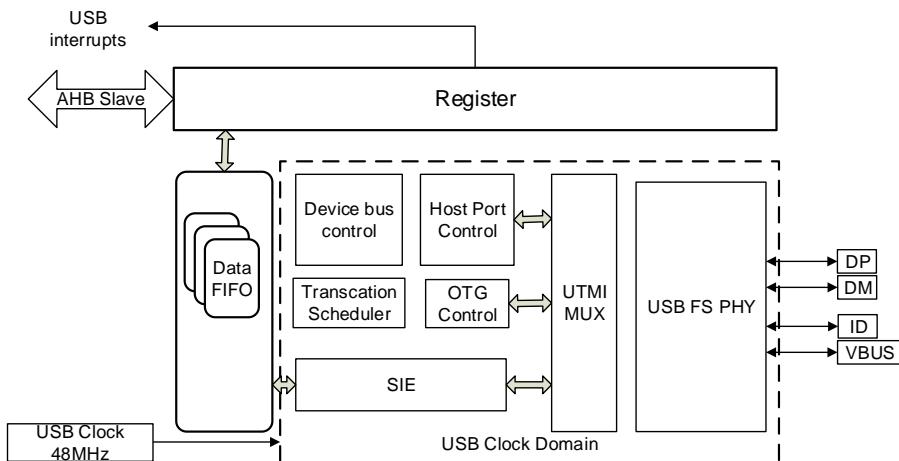
USB Full-Speed (USBFS) controller provides a USB-connection solution for portable devices. USBFS supports host and device modes, as well as OTG mode with HNP (Host Negotiation Protocol) and SRP (Session Request Protocol). USBFS contains a full-speed internal USB PHY and external PHY chip is not contained. USBFS supports all the four types of transfer (control, bulk, Interrupt and isochronous) which are defined in USB 2.0 protocol.

### 25.2. 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.

## 25.3. Block diagram

**Figure 25-1. USBFS block diagram**



## 25.4. Signal description

**Table 25-1. 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

## 25.5. Function overview

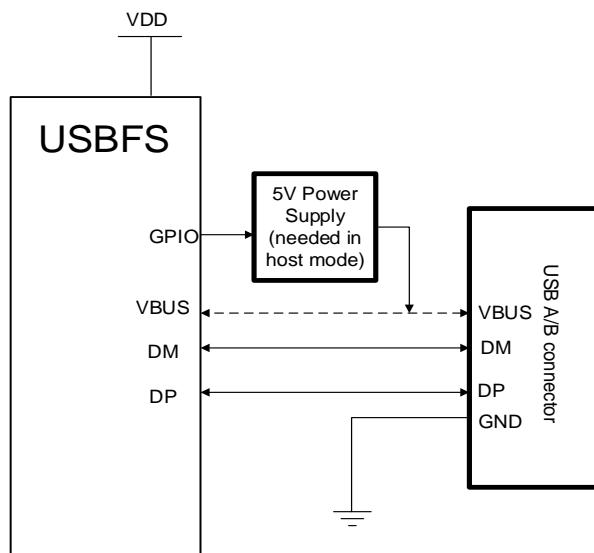
### 25.5.1. USBFS clocks and working modes

USBFS could be operated 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 25-2. Connection with host or device mode.](#)

Figure 25-2. Connection with host or device mode

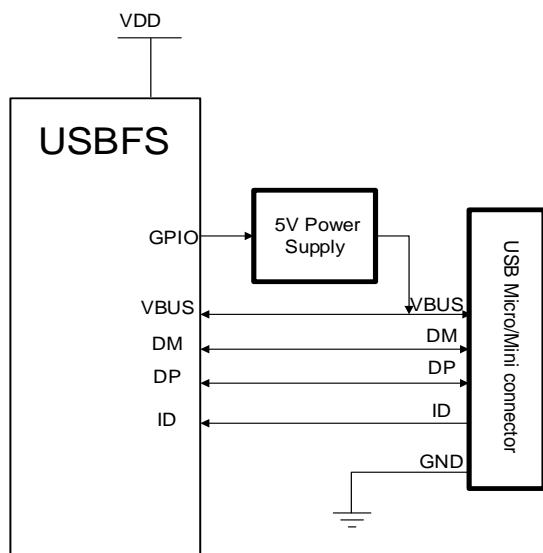


When USBFS works in host mode (FHM bit is set and FDM bit is cleared), the VBUS is 5V power, and detecting pin which is using for voltage detection is defined in USB protocol. The internal PHY cannot supply 5V VBUS power and only has some voltage comparers, charge and dis-charge circuits on VBUS line. Thus, 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 configured by VBUSIG bit in USBFS\_GCCFG register. So if the device is not necessary to detect the voltage on VBUS pin, it could be configured by setting the VBUSIG bit, then the VBUS pin can be freed for other uses. Otherwise, the VBUS connection cannot be omitted, and USBFS continuously monitor the VBUS voltage. It will immediately switch off the pull-up resistor on DP line once that the VBUS voltage falls below the needed valid value, then lead to be disconnection.

The OTG mode connection is described in the [Figure 25-3. 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 which is 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 25-3. Connection with OTG mode

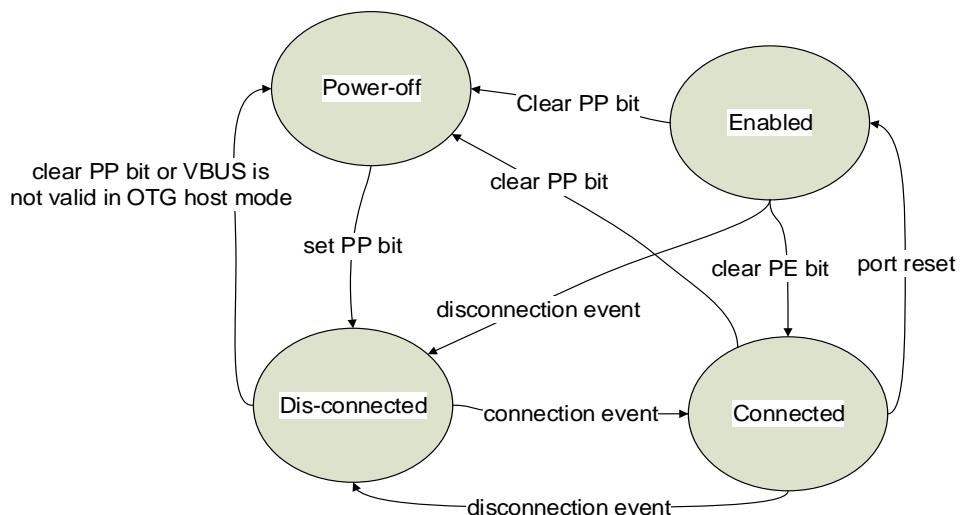


## 25.5.2. USB host function

### 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 25-4. 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 describing 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 into suspend state. In suspend state, USBFS stops sending SOFs on USB bus, and it will lead the connected USB device to enter into suspend state after 3ms. Application can set the PREM bit in USBFS\_HPCS register to start a resume sequence, so as 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 then 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 describing in USB 2.0 protocol, SOF packets are generated (by the host controller or hub transaction translator) at each 1ms in full-speed links.

Once that USBFS entered 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 for each SOF packet and output it to a pin. The pulse length is 12 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 wrote 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.

### 25.5.3. 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 or setting VBUSIG bit in USBFS\_GCCFG register, USBFS enters into powered state. USBFS begins to switch on the pull-up resistor on DP line, thus, host side will detect a connection event.

#### 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 describing in 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 if 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 by 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.

## 25.5.4. 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 about 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 changing the role to be a host is described in this section. This protocol eliminates the necessity of switching the cable connection for the roles change 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 into suspend status. Then, the B-Device will enter into 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. In additional, 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 save 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 is described in OTG protocol, an OTG device must compare VBUS voltage with several threshold values, and the compared 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. 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.

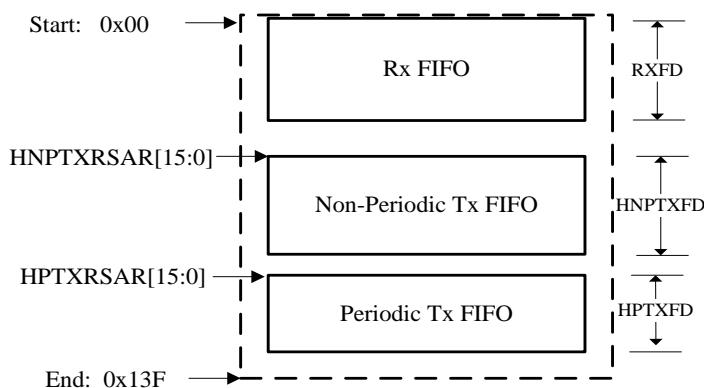
## 25.5.5. Data FIFO

The USBFS contains a 1.25K bytes data FIFO for packet data storing. The data FIFO is implemented by using an internal SRAM in USBFS.

### Host Mode

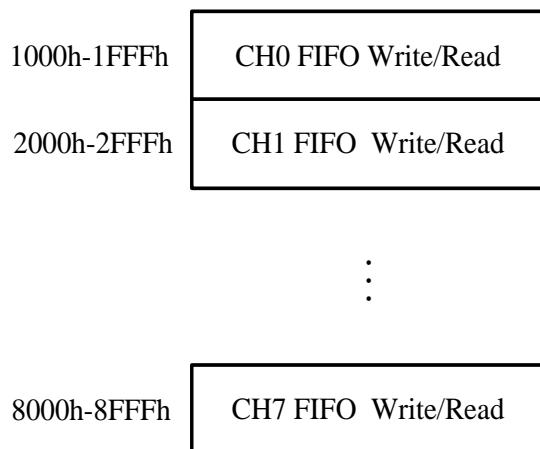
In host mode, the data FIFO space is divided into 3 parts: Rx FIFO for received packet, Non-Periodic Tx FIFO for non-period transmission packet and Periodic Tx FIFO for periodic transmission packet. All IN channels shares the Rx FIFO for packets reception. All the periodic OUT channels share the periodic Tx FIFO to packets transmission. All the non-periodic OUT channels share the non-Periodic 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 25-5. 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 25-5. HOST mode FIFO space in SRAM



USBFS provides a special register area for the internal data FIFO reading and writing. [Figure 25-6. Host mode FIFO access register map](#) describes the register memory area that the data FIFO can access. 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 by using USBFS\_GRSTATR/ USBFS\_GRSTATP register.

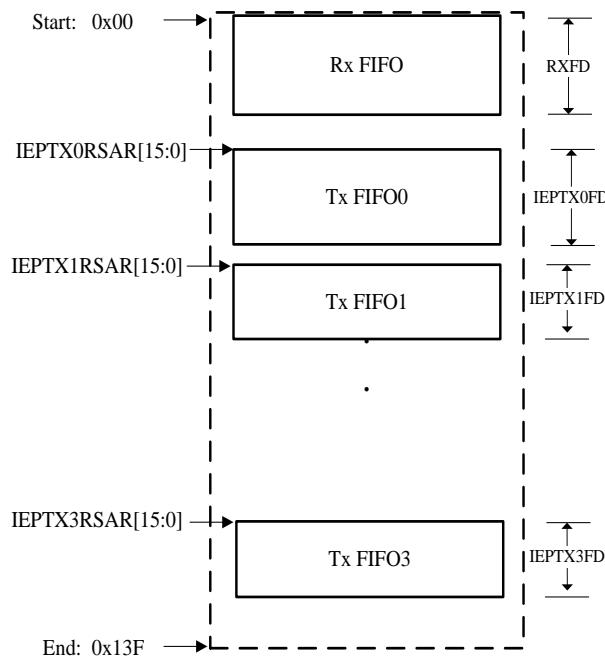
Figure 25-6. Host mode FIFO access register map



#### Device mode

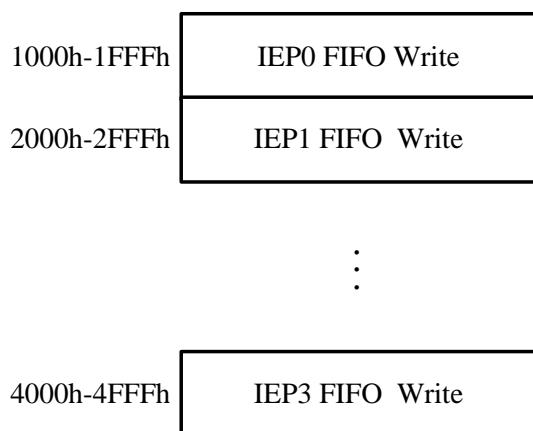
In device mode, the data FIFO is divided into several parts: one Rx FIFO, and 4 Tx FIFOs (one for each IN endpoint). All the OUT endpoints share the Rx FIFO for receiving packets. The size and start offset of these data FIFOs should be configured by using USBFS\_GRFLEN and USBFS\_DIEPxTFLEN ( $x=0\dots3$ ) registers. [Figure 25-7. 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 25-7. Device mode FIFO space in SRAM**



USBFS provides a special register area for the internal data FIFO reading and writing. [\*\*Figure 25-8. Device mode FIFO access register map\*\*](#) describes the register memory area where the data FIFO can access. 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 by using USBFS\_GRSTATR/USBFS\_GRSTATP register.

**Figure 25-8. Device mode FIFO access register map**



## 25.5.6. 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 byte 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 finishing, TLEN could 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 will be 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. Then software should handle the Rx FIFO not empty event: read and pop this status entry, and 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 is finished 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 have been 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 has been finished.
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 has been finished on USB bus, PCNT in USBFS\_HCHxLEN register is decreased by 1. If the transaction is finished 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 byte 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 could 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. At any 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 report the transaction result.
5. After all the data packets in a transfer have been successfully sent on USB bus, USBFS generates TF flag to indicate that the transfer successfully is finished and the IN endpoint is disabled.

#### **OUT transfers operation sequence**

1. Initialize USBFS global registers.
2. Initialize the endpoint and enable the endpoint.
3. When an OUT token is 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 is finished 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 is finished and the IN endpoint is disabled.

## 25.6. 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 25-2. USBFS global interrupt](#).

**Table 25-2. USBFS global interrupt**

Interrupt Flag	Description	Operation Mode
SESIF	Session interrupt	Host or device mode
DISCIF	Disconnect interrupt flag	Host Mode
IDPSC	ID pin status change	Host or device 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/PXNCIF	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
OEPIF	OUT endpoint interrupt flag	Device mode
IEPIF	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

Interrupt Flag	Description	Operation 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 if when the USBFS's clocks are stopped. The source of the wake-up interrupt is WKUPIF bit in USBHS\_GINTF register.

## 25.7. Register definition

USBFS base address: 0x5000 0000

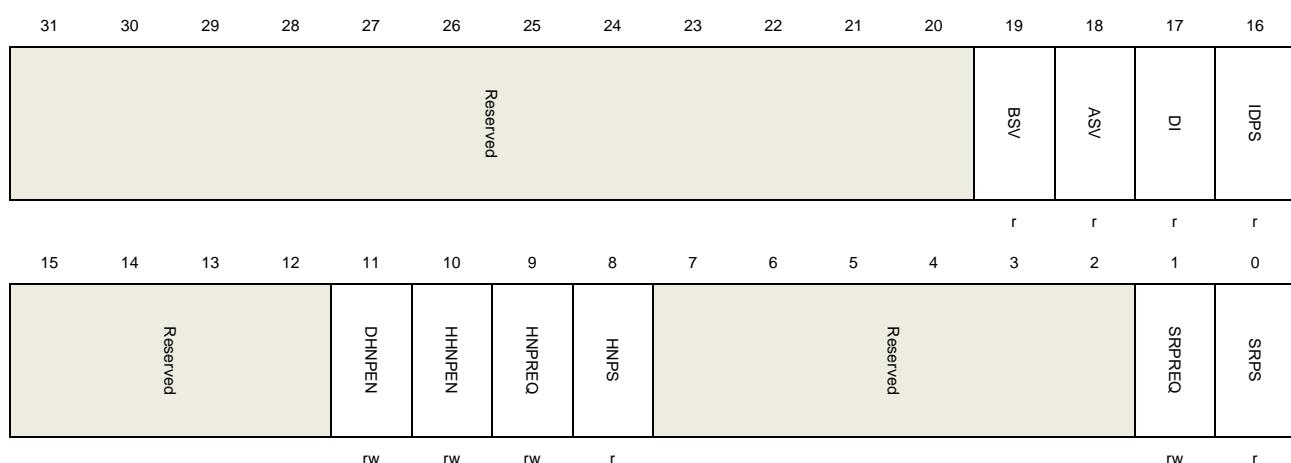
### 25.7.1. Global control and status registers

#### 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)



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 not below VBSESSVLD <b>Note:</b> 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 below VASESSVLD The A-Device is the default host at the start of a session. <b>Note:</b> 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.
		<b>Note:</b> 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><b>Note:</b> 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><b>Note:</b> Only accessible in device mode.</p>
10	HHNPEN	<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><b>Note:</b> 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><b>Note:</b> 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><b>Note:</b> 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	

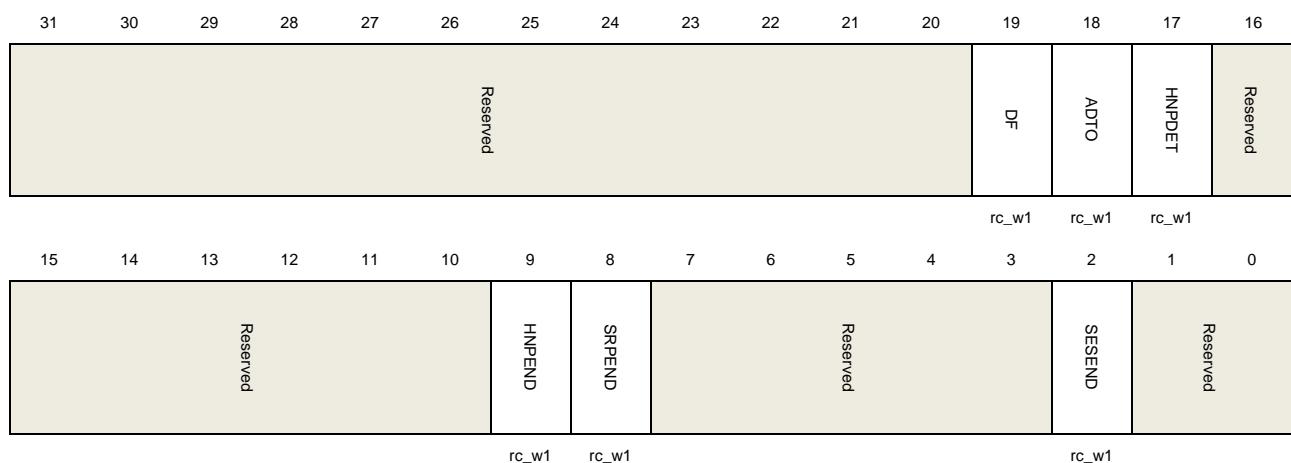
**Note:** Only accessible in device mode.

### Global OTG interrupt flag register (USBFS\_GOTGINTR)

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. <b>Note:</b> 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. <b>Note:</b> Accessible in both device and host modes.
17	HNPDET	Host negotiation request detected Set by USBFS when A-Device detects a HNP request. <b>Note:</b> 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.

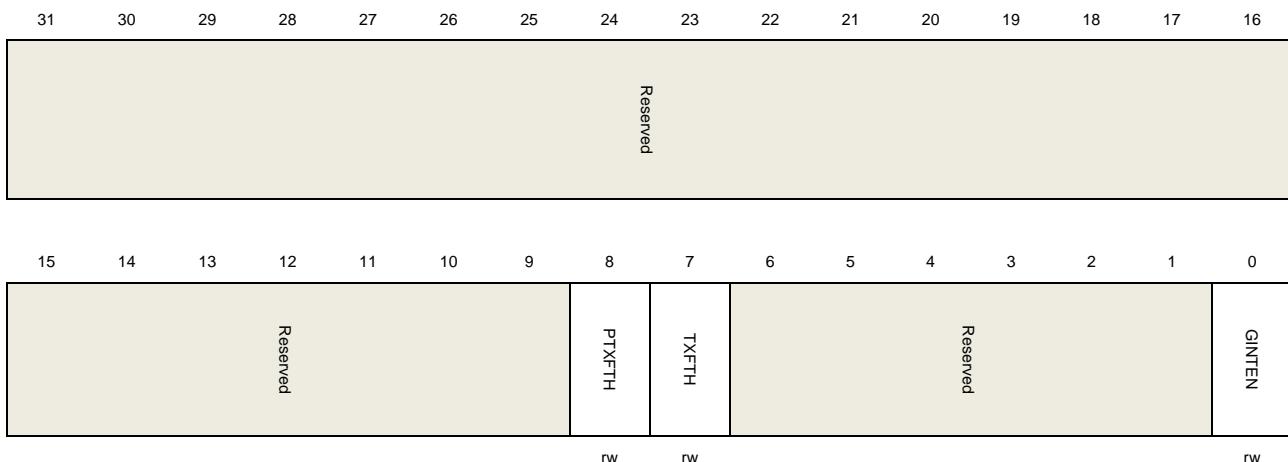
8	SRPEND	SRPEND
		Set by the core when a SRP ends. Read the SRPS in USBFS_GOTGCS register to get the result of SRP.
		<b>Note:</b> Accessible in both device and host modes.
7:3	Reserved	Must be kept at reset value
2	SESEND	Session end Set by the core when VBUS voltage is below Vb_ses_vld.
1:0	Reserved	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	PTXFTH	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 <b>Note:</b> Only accessible in host mode.
7	TXFTH	Tx FIFO threshold <b>Device mode:</b> 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)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	FDM	FHM													
	rw	rw													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		UTT[3:0]			HNP[CEN]	SRC[CEN]			Reserved						rw
		rw			r/rw	r/rw									

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.  <b>Note:</b> 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.
		<b>Note:</b> 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. <b>Note:</b> Only accessible in device mode.
9	HNPSEN	HNP capability enable Controls whether the HNP capability is enabled 0: HNP capability is disabled 1: HNP capability is enabled <b>Note:</b> Accessible in both device and host modes.
8	SRPCEN	SRP capability enable Controls whether the SRP capability is enabled 0: SRP capability is disabled 1: SRP capability is enabled <b>Note:</b> 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
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Reserved

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>Tx FIFO number</p> <p>Indicates which Tx FIFO will be flushed when TXFF bit in the same register is set.</p> <p>Host Mode:</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>Device Mode:</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>Tx FIFO flush</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><b>Note:</b> Accessible in both device and host modes.</p>
4	RXFF	<p>Rx FIFO flush</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><b>Note:</b> Accessible in both device and host modes.</p>
3	Reserved	Must be kept at reset value
2	HFCRST	<p>Host frame counter reset</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><b>Note:</b> 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	SEEIF	DISCIF	IDPSC	Reserved.	P1XFEIF	HCFIF	HPIF	Reserved		PXNCIF/ ISOONCIF	ISOINCIF	OEPIF	IEPIF	Reserved	
rc_w1	rc_w1	rc_w1	rc_w1	r	r	r	r	rc_w1	rc_w1	r	r	r	r	r	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EOPFIF	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	r	r	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.  <b>Note:</b> Accessible in both device and host modes.
30	SEEIF	Session interrupt flag  This interrupt is triggered when a SRP is detected (in A-Device mode) or VBUS becomes valid for a B- Device (in B-Device mode).  <b>Note:</b> Accessible in both device and host modes.
29	DISCIF	Disconnect interrupt flag  This interrupt is triggered after a device disconnection.  <b>Note:</b> Only accessible in host mode.
28	IDPSC	ID pin status change  Set by the core when ID status changes.  <b>Note:</b> 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><b>Note:</b> 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.</p> <p>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><b>Note:</b> 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.</p> <p>Software should read USBFS_HPCS register to get the source of this interrupt.</p> <p>This bit will be automatically cleared after the flags that causing a port interrupt are cleared.</p> <p><b>Note:</b> 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><b>Note:</b> 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> <p><b>Note:</b> Only accessible in device mode.</p>

18	IEPIF	IN endpoint interrupt flag  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.  <b>Note:</b> Only accessible in device mode.
17:16	Reserved	Must be kept at reset value
15	EOPFIF	End of periodic frame interrupt flag  When USB bus time in a frame reaches the value defined by EOPFT [1:0] bits in USBFS_DCFG register, USBFS sets this flag.  <b>Note:</b> Only accessible in device mode.
14	ISOOPDIF	Isochronous OUT packet dropped interrupt flag  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.  <b>Note:</b> Only accessible in device mode.
13	ENUMF	Enumeration finished  USBFS sets this bit after the speed enumeration finishes. Read USBFS_DSTAT register to get the current device speed.  <b>Note:</b> Only accessible in device mode.
12	RST	USB reset  USBFS sets this bit when it detects a USB reset signal on bus.  <b>Note:</b> Only accessible in device mode.
11	SP	USB suspend  USBFS sets this bit when it detects that the USB bus is idle for 3 ms and enters suspend state.  <b>Note:</b> Only accessible in device mode.
10	ESP	Early suspend  USBFS sets this bit when it detects that the USB bus is idle for 3 ms.  <b>Note:</b> Only accessible in device mode.
9:8	Reserved	Must be kept at reset value
7	GONAK	Global OUT NAK effective  Write 1 to SGONAK bit in the USBFS_DCTL register and USBFS will set GONAK flag after the writing to SGONAK takes effect.  <b>Note:</b> Only accessible in device mode.
6	GNPINAK	Global Non-Periodic IN NAK effective  Write 1 to SGINAK bit in the USBFS_DCTL register and USBFS will set GNPINAK flag after the writing to SGINAK takes effect.

		<b>Note:</b> 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>
		<b>Note:</b> Only accessible in host mode.
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>
		<b>Note:</b> Accessible in both host and device modes.
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>
		<b>Note:</b> Accessible in both host and device modes.
2	OTGIF	<p>OTG interrupt flag</p> <p>USBFS sets this bit when the flags in USBFS_GOTGINTR register generate an interrupt. Software should read USBFS_GOTGINTR register to get the source of this interrupt. This bit is cleared after the flags in USBFS_GOTGINTR causing this interrupt are cleared.</p>
		<b>Note:</b> Accessible in both host and device modes.
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>
		<b>Note:</b> Accessible in both host and device modes.
0	COPM	<p>Current operation mode</p> <p>0: Device mode</p> <p>1: Host mode</p>
		<b>Note:</b> Accessible in both host and device modes.

## 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	SESIE	DISCIE	IDPSCIE	Reserved.	PTXFEIE	HClIE	HPIE	Reserved	ISOONClIE	PXNCIE/	ISOINClIE	OEPIE	IEPIE	Reserved	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
EOPPIE	ISOOPDIE	ENUMFIE	RSTIE	SPIE	ESPIE	Reserved	GONAKIE	GNPINAKIE	NPTXFEIE	RXFNEIE	SOFIE	OTGIE	MFIE	Reserved	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 <b>Note:</b> Accessible in both host and device modes.
30	SESIE	Session interrupt enable 0: Disable session interrupt 1: Enable session interrupt <b>Note:</b> Accessible in both host and device modes.
29	DISCIE	Disconnect interrupt enable 0: Disable disconnect interrupt 1: Enable disconnect interrupt <b>Note:</b> 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 <b>Note:</b> 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 <b>Note:</b> Only accessible in host mode.
25	HClIE	Host channels interrupt enable 0: Disable host channels interrupt 1: Enable host channels interrupt <b>Note:</b> Only accessible in host mode.

---

24	HPIE	Host port interrupt enable 0: Disable host port interrupt 1: Enable host port interrupt <b>Note:</b> 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 <b>Note:</b> 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 <b>Note:</b> 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 <b>Note:</b> Only accessible in device mode.
19	OEPIE	OUT endpoints interrupt enable 0: Disable OUT endpoints interrupt 1: Enable OUT endpoints interrupt <b>Note:</b> Only accessible in device mode.
18	IEPIE	IN endpoints interrupt enable 0: Disable IN endpoints interrupt 1: Enable IN endpoints interrupt <b>Note:</b> 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 <b>Note:</b> 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 <b>Note:</b> Only accessible in device mode.
13	ENUMFIE	Enumeration finish enable 0: Disable enumeration finish interrupt 1: Enable enumeration finish interrupt <b>Note:</b> Only accessible in device mode.

---

12	RSTIE	USB reset interrupt enable 0: Disable USB reset interrupt 1: Enable USB reset interrupt <b>Note:</b> Only accessible in device mode.
11	SPIE	USB suspend interrupt enable 0: Disable USB suspend interrupt 1: Enable USB suspend interrupt <b>Note:</b> Only accessible in device mode.
10	ESPIE	Early suspend interrupt enable 0: Disable early suspend interrupt 1: Enable early suspend interrupt <b>Note:</b> 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 <b>Note:</b> 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 <b>Note:</b> 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 <b>Note:</b> 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 <b>Note:</b> 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 <b>Note:</b> Accessible in both device and host modes.
2	OTGIE	OTG interrupt enable 0: Disable OTG interrupt 1: Enable OTG interrupt <b>Note:</b> 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

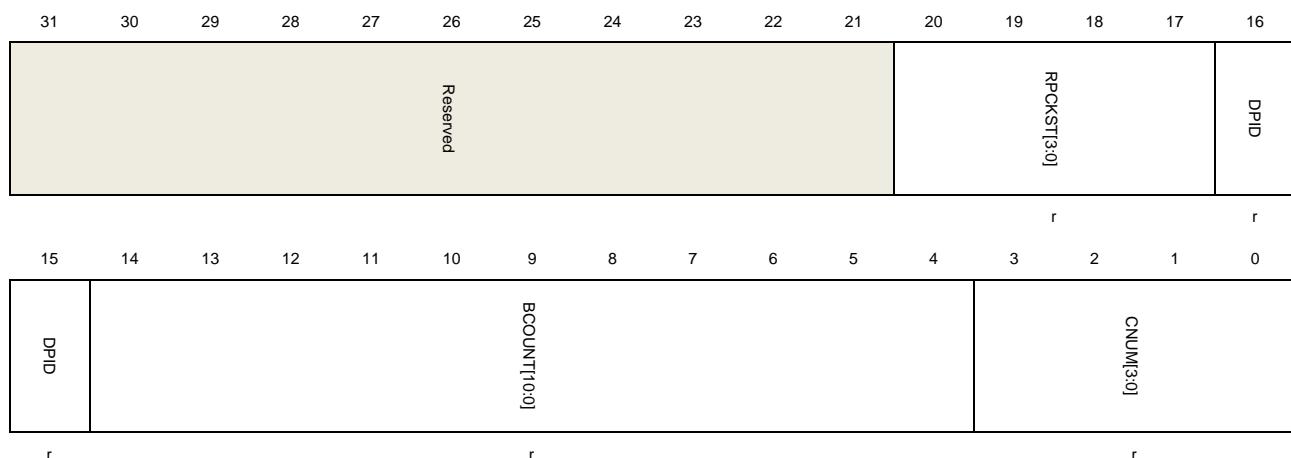
Reset value: 0x0000 0000

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 Rx FIFO 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]	Received packet status 0010: IN data packet received 0011: IN transfer completed (generates an interrupt if popped) 0101: Data toggle error (generates an interrupt if popped) 0111: Channel halted (generates an interrupt if popped) Others: Reserved
16:15	DPID[1:0]	Data PID The Data PID of the received packet 00: DATA0

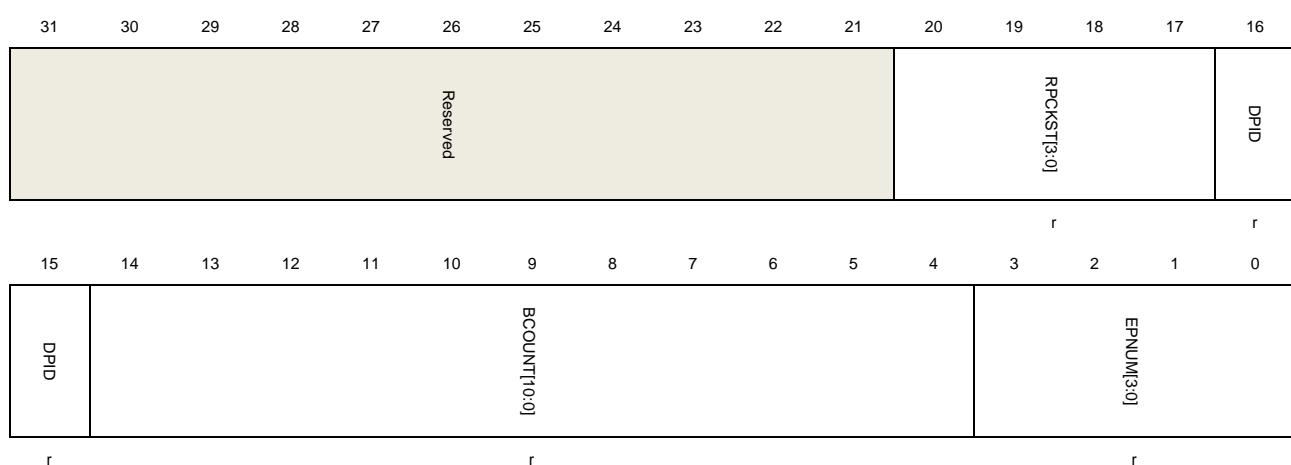
10: DATA1

Others: Reserved

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:**



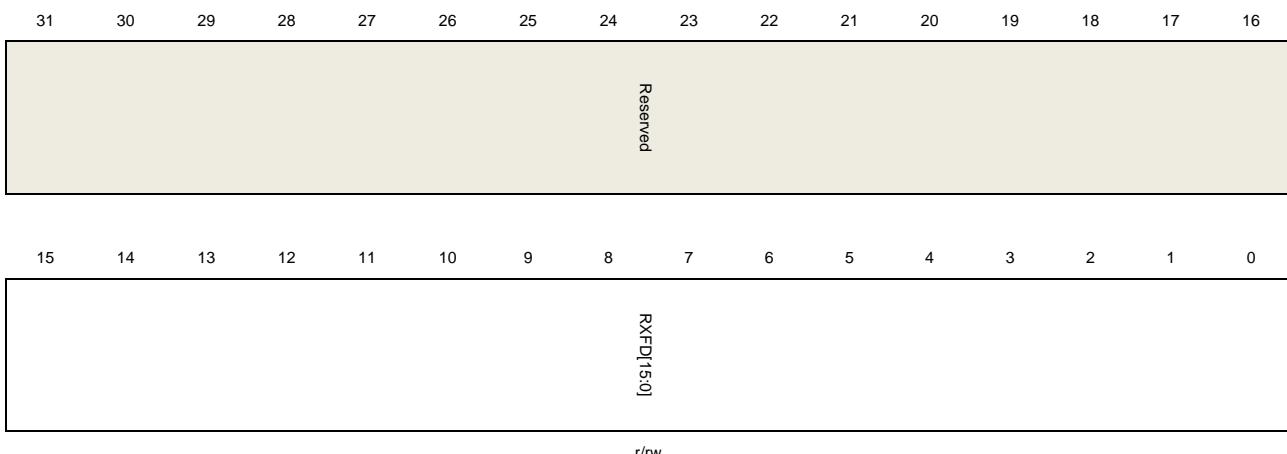
<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:21	Reserved	Must be kept at reset value
20:17	RPCKST[3:0]	Received packet status 0001: Global OUT NAK (generates an interrupt) 0010: OUT data packet received 0011: OUT transfer completed (generates an interrupt) 0100: SETUP transaction completed (generates 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 Others: Reserved
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)



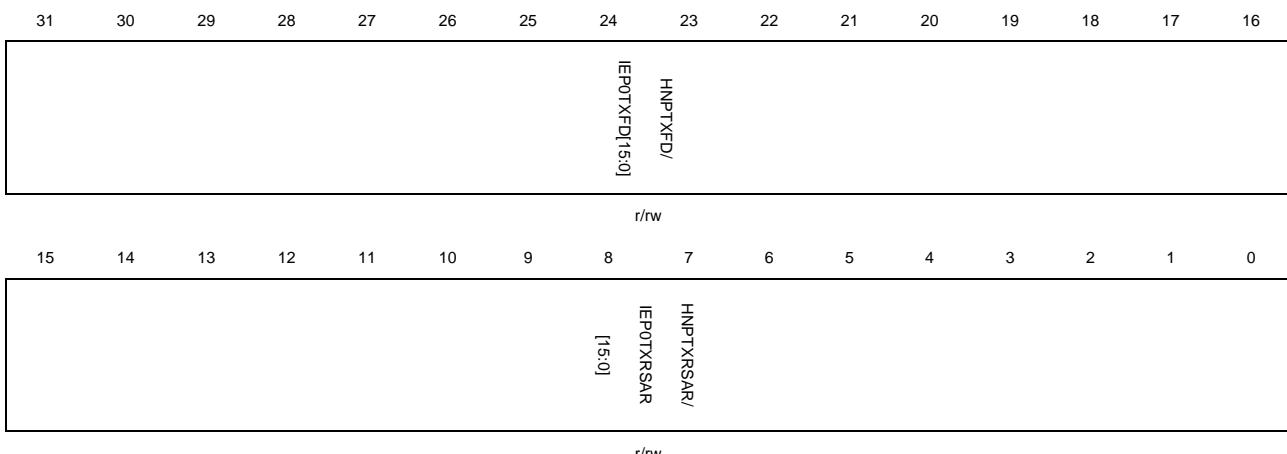
Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	RXFD[15:0]	Rx FIFO depth  In terms of 32-bit words.  1≤RXFD≤1024

### **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)**

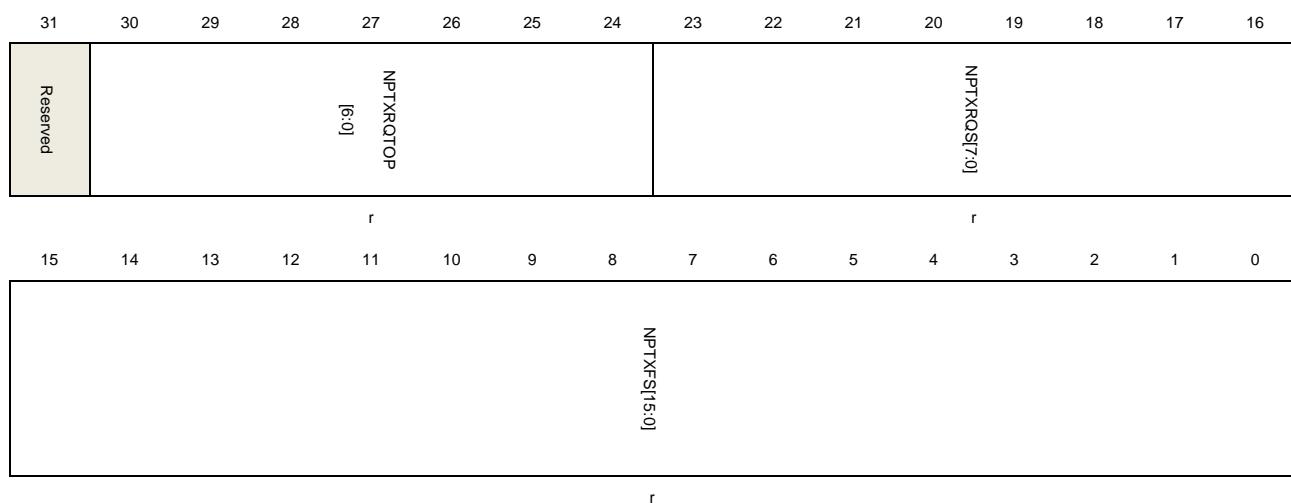
Address offset: 0x002C

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:
		– 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	NPTXRQS[7:0]	<p>Non-periodic Tx request queue space</p> <p>The remaining space of the non-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	NPTXFS[15:0]	<p>Non-periodic Tx FIFO space</p> <p>The remaining space of the non-periodic transmit FIFO.</p> <p>In terms of 32-bit words.</p> <p>0: Non-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 NPTXFD</math>)</p> <p>Others: Reserved</p>

### 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	VBUSACEN		Reserved	PWRON
										RW	RW	RW	RW		RW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:22	Reserved	Must be kept at reset value
21	VBUSIG	<p><math>V_{BUS}</math> ignored</p> <p>When this bit is set, USBFS doesn't monitor the voltage on <math>V_{BUS}</math> pin and always consider <math>V_{BUS}</math> voltage as valid both in host mode and in device mode, then free the <math>V_{BUS}</math> pin for other usage.</p> <p>0: <math>V_{BUS}</math> is not ignored.</p> <p>1: <math>V_{BUS}</math> is ignored and always consider <math>V_{BUS}</math> 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 <math>V_{BUS}</math> B-device Comparer enable</p> <p>0: <math>V_{BUS}</math> B-device comparer disabled</p> <p>1: <math>V_{BUS}</math> B-device comparer enabled</p>
18	VBUSACEN	<p>The <math>V_{BUS}</math> A-device Comparer enable</p> <p>0: <math>V_{BUS}</math> A-device comparer disabled</p> <p>1: <math>V_{BUS}</math> 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



rw

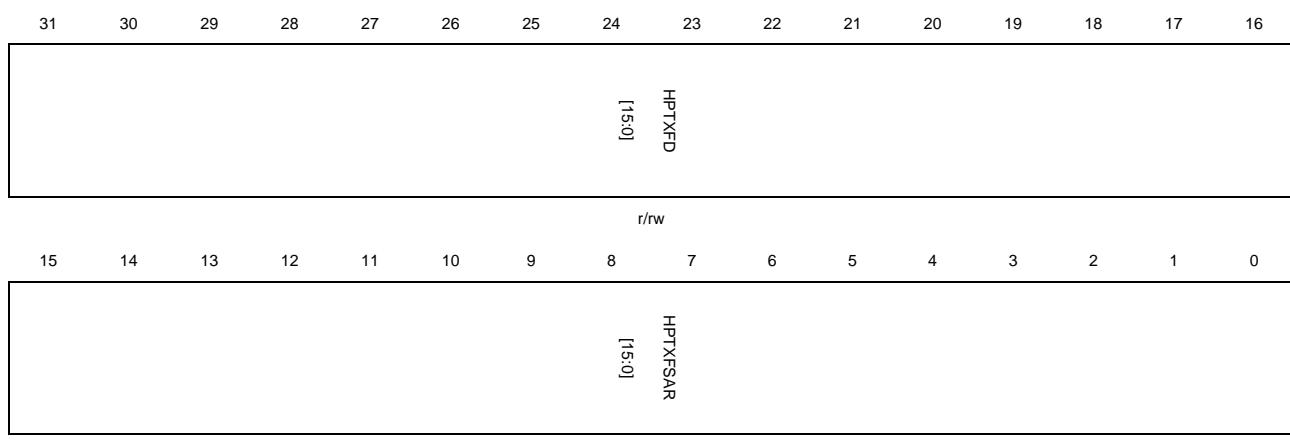
<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:0	CID	Core ID Software can write or read this field and uses this field as a unique ID for its application

### **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)



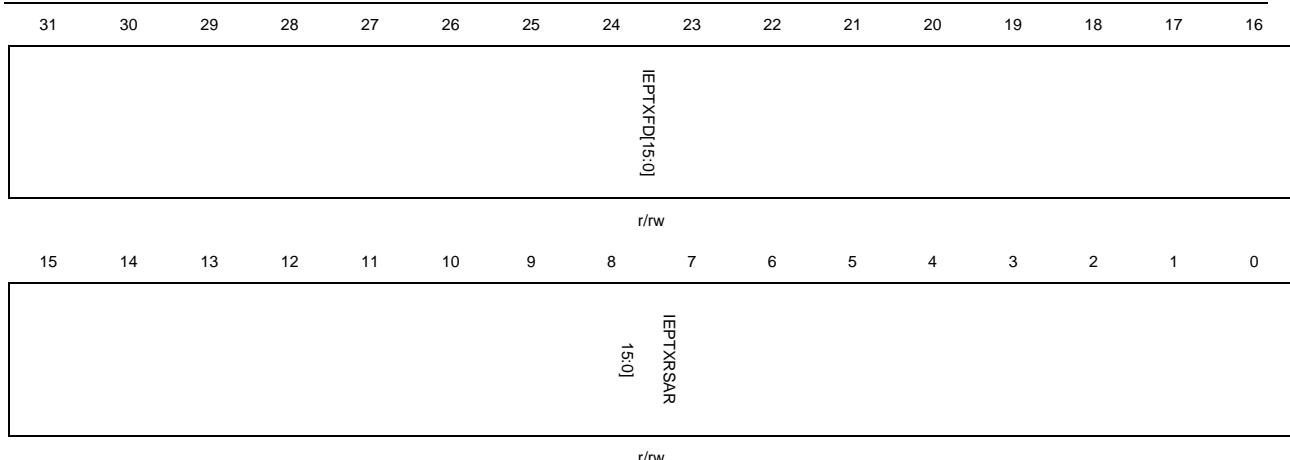
<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:16	HPTXFD[15:0]	Host Periodic Tx FIFO depth In terms of 32-bit words. $1 \leq HPTXFD \leq 1024$
15:0	HPTXFSAR[15:0]	Host periodic Tx FIFO RAM start address The start address for host periodic transmit FIFO RAM is in term of 32-bit words.

### **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)



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

## 25.7.2. Host control and status registers

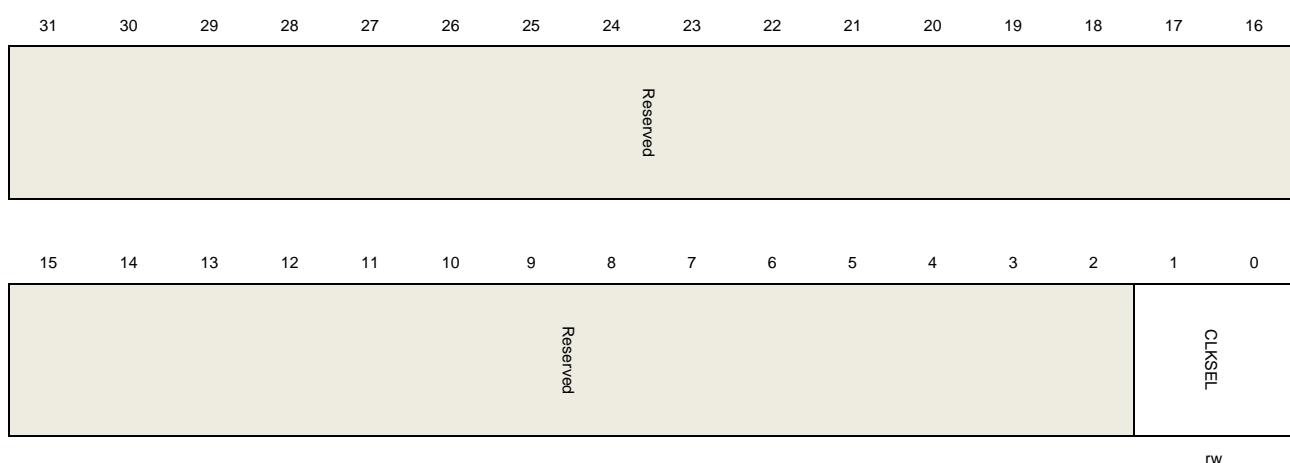
### Host control register (USBFS\_HCTL)

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	Clock select for usbclock. 01: 48MHz clock others: reserved

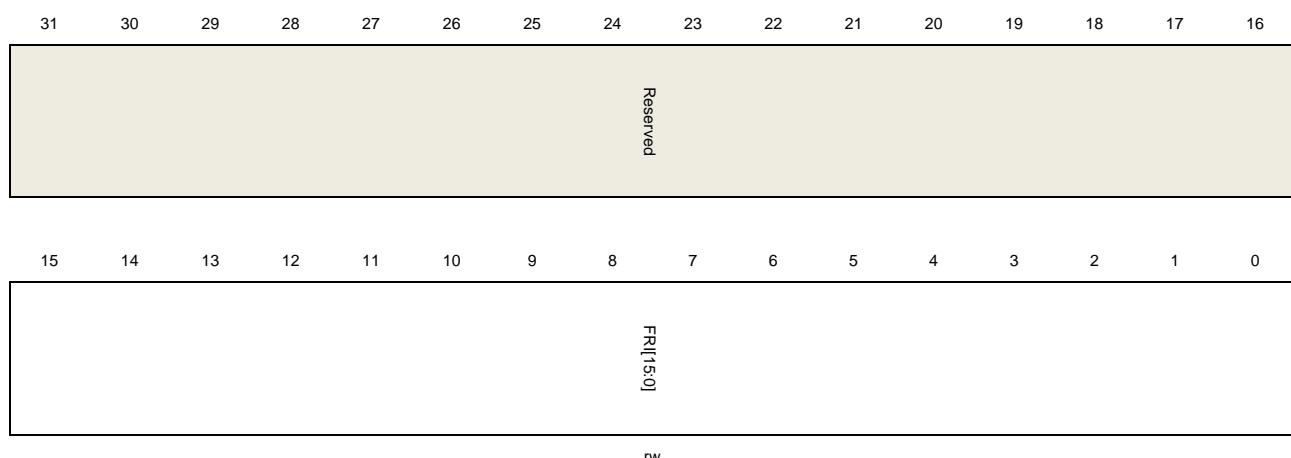
### **Host frame interval register (USBFS\_HFT)**

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)



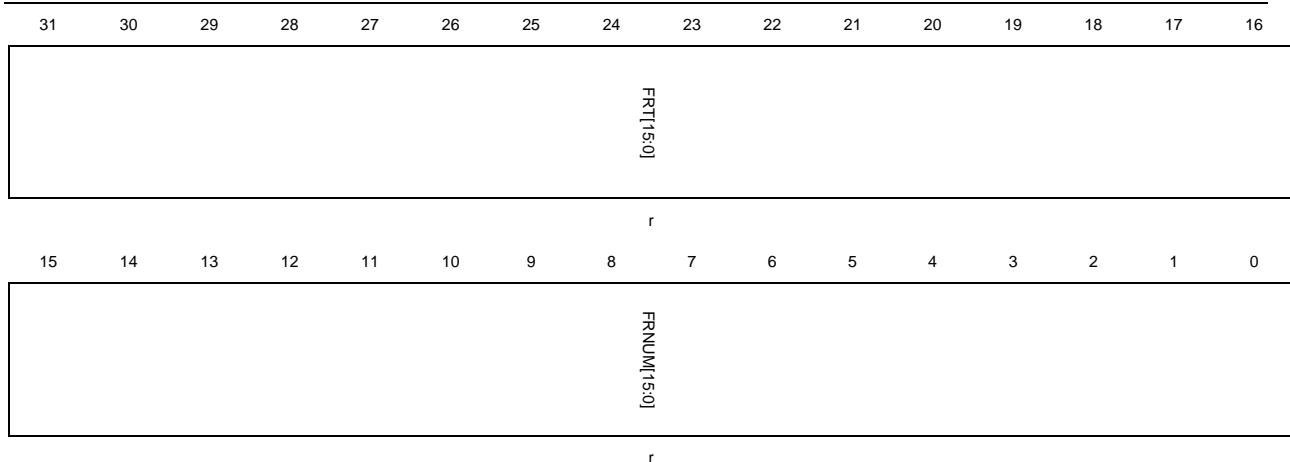
<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:16	Reserved	Must be kept at reset value
15:0	FRI[15:0]	Frame interval  This value describes the frame time in terms of PHY clocks. Each time when port is enabled after a port reset operation, USBFS use a proper value according to the current speed, and 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.

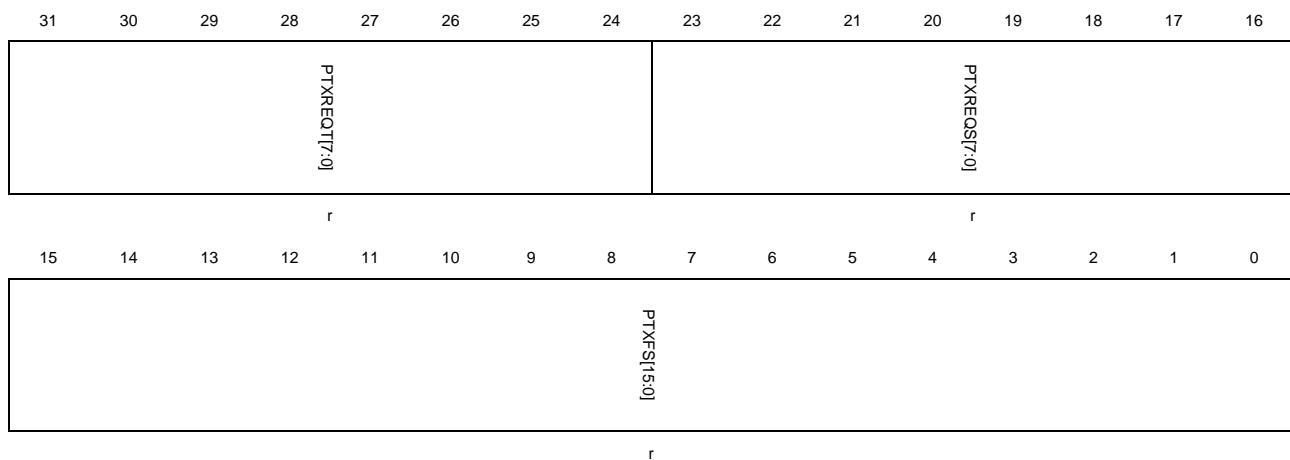
### **Host periodic transmit FIFO/queue status register (USBFS\_HPTFQSTAT)**

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>

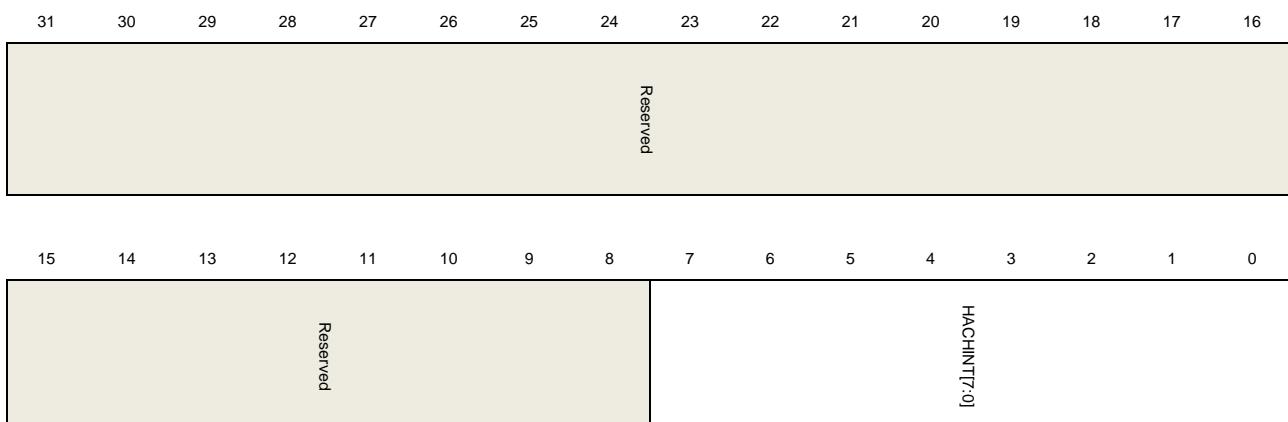
### Host all channels interrupt register (USBFS\_HACHINT)

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.

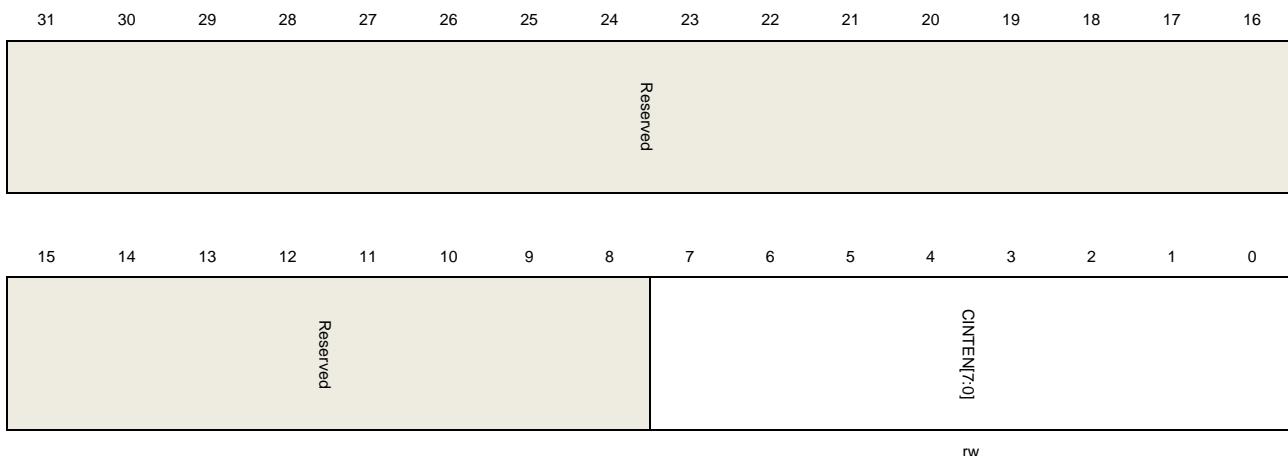
### **Host all channels interrupt enable register (USBFS\_HACHINTEN)**

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.

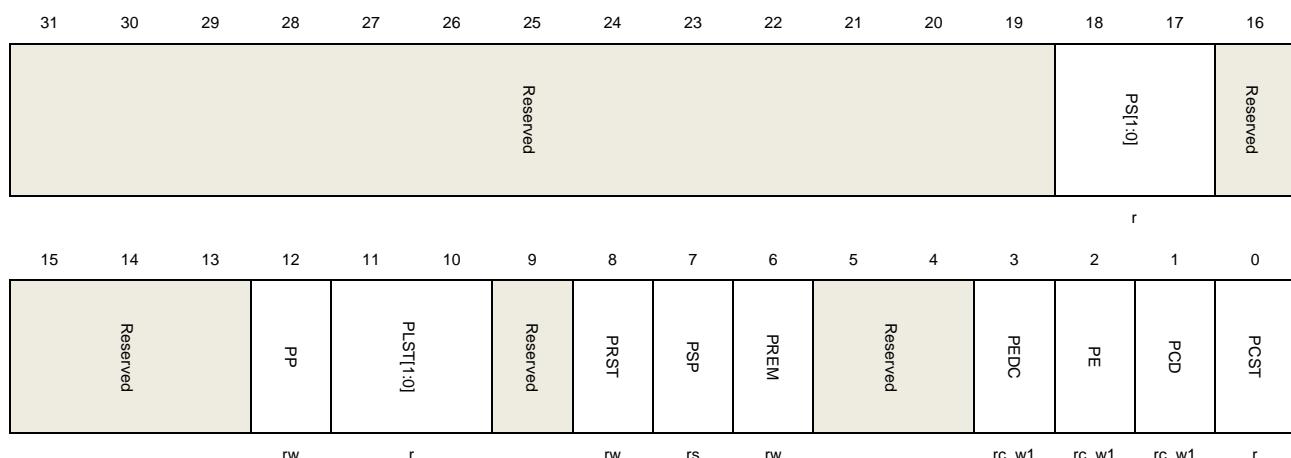
### **Host port control and status register (USBFS\_HPCS)**

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)



Bits	Fields	Descriptions
31:19	Reserved	Must be kept at reset value
18:17	PS	<p>Port speed</p> <p>Report the enumerated speed of the device attached to this port.</p> <p>01: Full speed</p> <p>10: Low speed</p> <p>Others: Reserved</p>
16:13	Reserved	Must be kept at reset value
12	PP	<p>Port power</p> <p>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.</p> <p>Software should ensure the true power supply on VBUS before setting this bit.</p> <p>0: Port is powered off</p> <p>1: Port is powered on</p>
11:10	PLST	<p>Port line status</p> <p>Report the current state of USB data lines</p> <p>Bit 10: State of DP line</p> <p>Bit 11: State of DM line</p>
9	Reserved	Must be kept at reset value
8	PRST	<p>Port reset</p> <p>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.</p> <p>0: Port is not in reset state</p> <p>1: Port is in reset state</p>
7	PSP	<p>Port suspend</p> <p>Application sets this bit to put port into suspend state. When this bit is set the port</p>

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: <ul style="list-style-type: none"> <li>– A disconnect condition</li> <li>– Software clearing this bit</li> </ul> 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)

CEN	CDIS	ODDFRM	DAR[6:0]						Reserved	EPTYPE[1:0]	LSD	Reserved	
rs	rs	rw			rw					rw	rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	
EPDIR	EPNUM[3:0]			MPI[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	<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	<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      Endpoint number

The number of the endpoint that this channel wants to communicate with.

10:0      MPL      Maximum packet length

The target endpoint's maximum packet length.

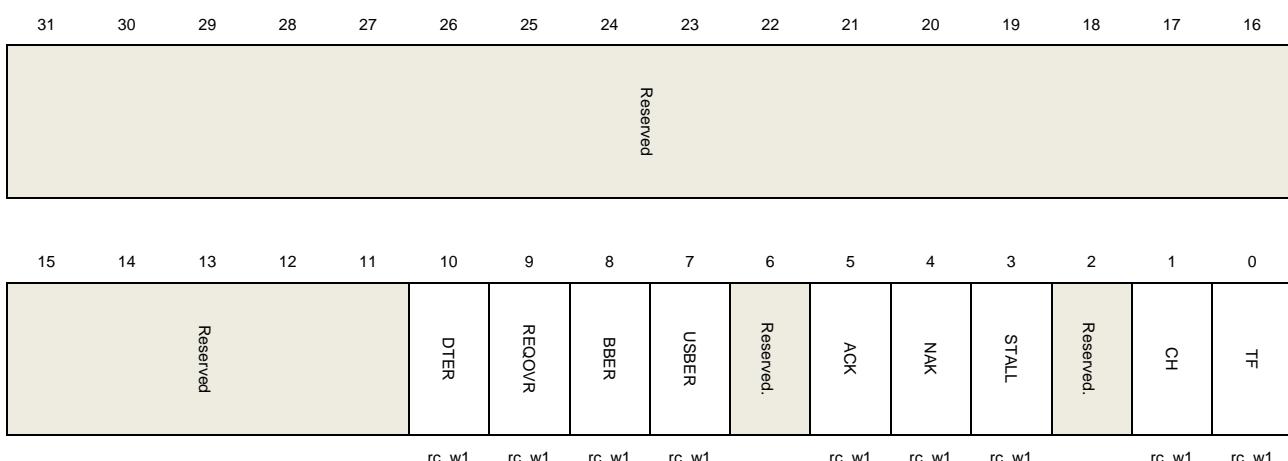
### **Host channel-x interrupt flag register (USBFS\_HCHxINTF) (x = 0..7 where x = channel number)**

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)



<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
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:
		<ul style="list-style-type: none"> <li>- A received packet has a wrong CRC field</li> <li>- A stuff error detected on USB bus</li> <li>- Timeout when waiting for a response packet</li> </ul>
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)

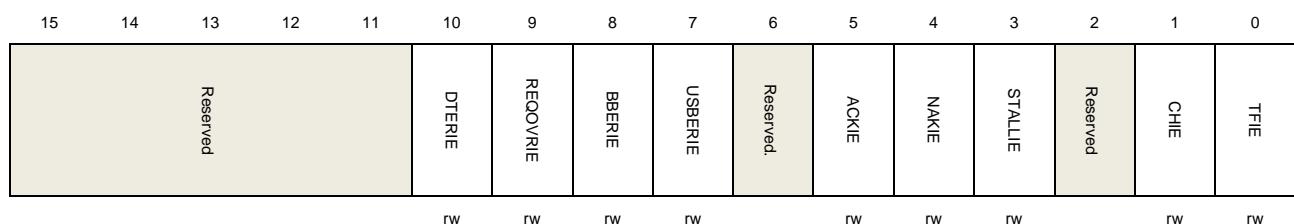
Reset value: 0x0000 0000

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)

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

Reserved



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

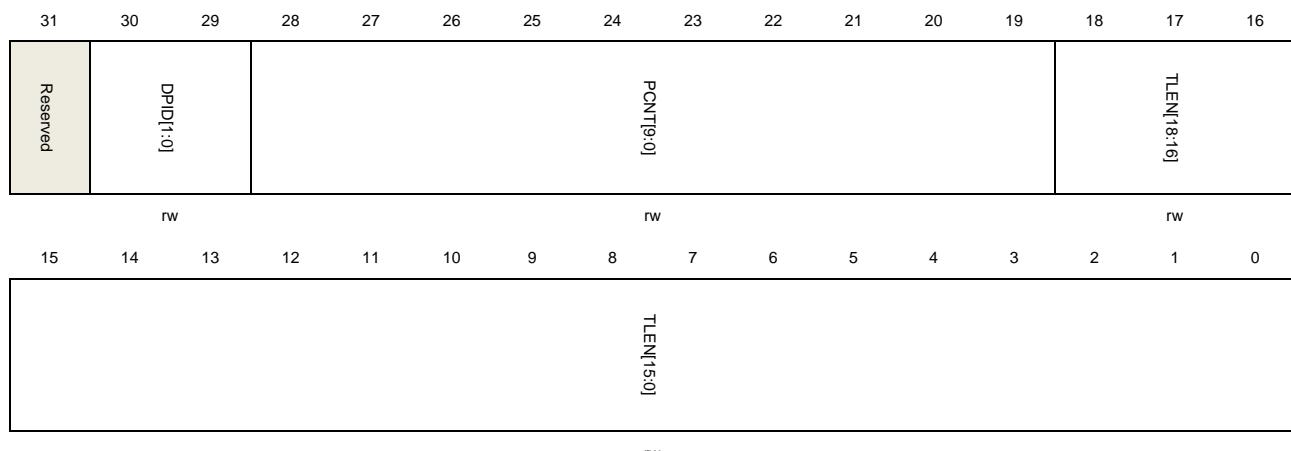
1: Enable transfer finished interrupt

### **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]	<p>Data PID</p> <p>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.</p> <p>00: DATA0</p> <p>10: DATA1</p> <p>11: SETUP (For control transfer only)</p> <p>01: Reserved</p>
28:19	PCNT[9:0]	<p>Packet count</p> <p>The number of data packets desired to be transmitted (OUT) or received (IN) in a transfer.</p> <p>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.</p>
18:0	TLEN[18:0]	<p>Transfer length</p> <p>The total data byte number of a transfer.</p> <p>For OUT transfers, this field is the total data bytes of all the data packets desired</p>

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.

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.

### 25.7.3. Device control and status registers

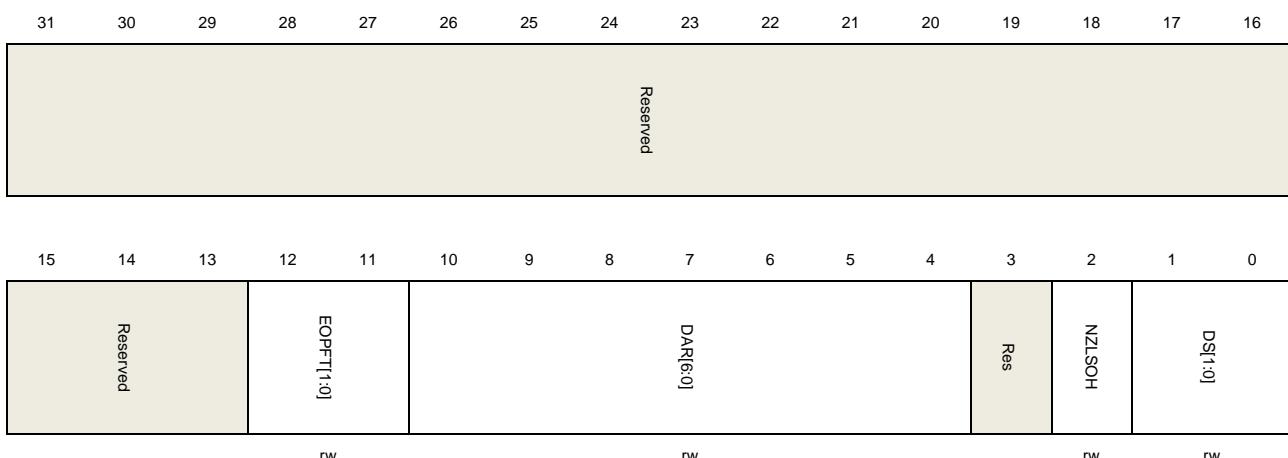
#### Device configuration register (USBFS\_DCFG)

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> <p>10: 90% of the frame time</p> <p>11: 95% of the frame time</p>
10:4	DAR[6:0]	<p>Device address</p> <p>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.</p>

---

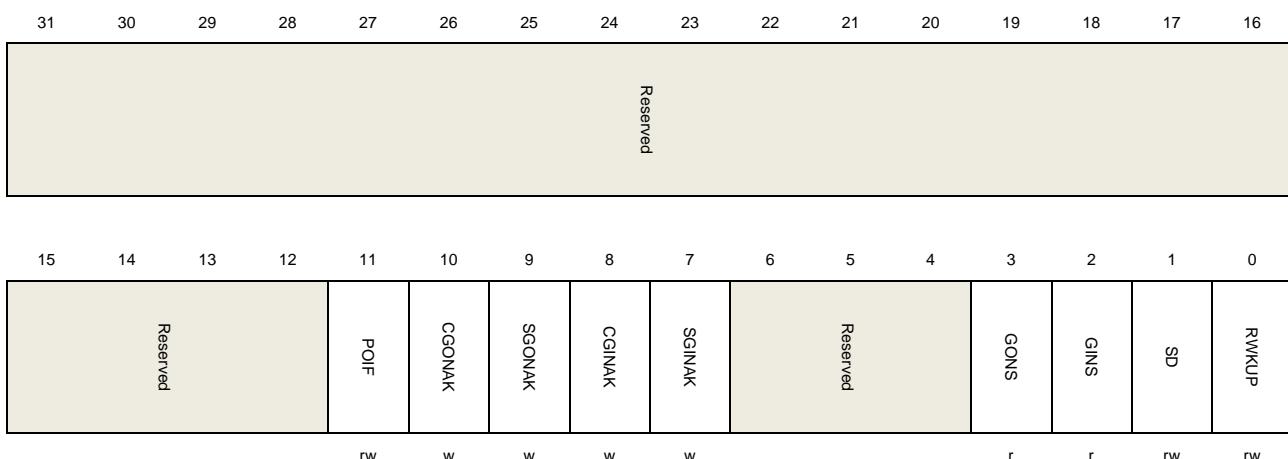
3	Reserved	Must be kept at reset value
2	NZLSOH	<p>Non-zero-length status OUT handshake</p> <p>When a USB device receives a non-zero-length data packet during status OUT stage, this field controls that either USBFS should receive this packet or reject this packet with a STALL handshake.</p> <p>0: Treat this packet as a normal packet and response according to the status of NAKS and STALL bits in USBFS_DOEPxCTL register.</p> <p>1: Send a STALL handshake and don't save the received OUT packet.</p>
1:0	DS[1:0]	<p>Device speed</p> <p>This field controls the device speed when the device connected to a host.</p> <p>11: Full speed</p> <p>Others: Reserved</p>

### **Device control register (USBFS\_DCTL)**

Address offset: 0x0804

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	POIF	<p>Power-on initialization finished</p> <p>Software should set this bit to notify USBFS that the registers are initialized after waking up from power down state.</p>
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</p>

		USBFS_GINTF register triggered after a while. Software should clear the GONAK flag before writing this bit again.
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.</p> <p>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>

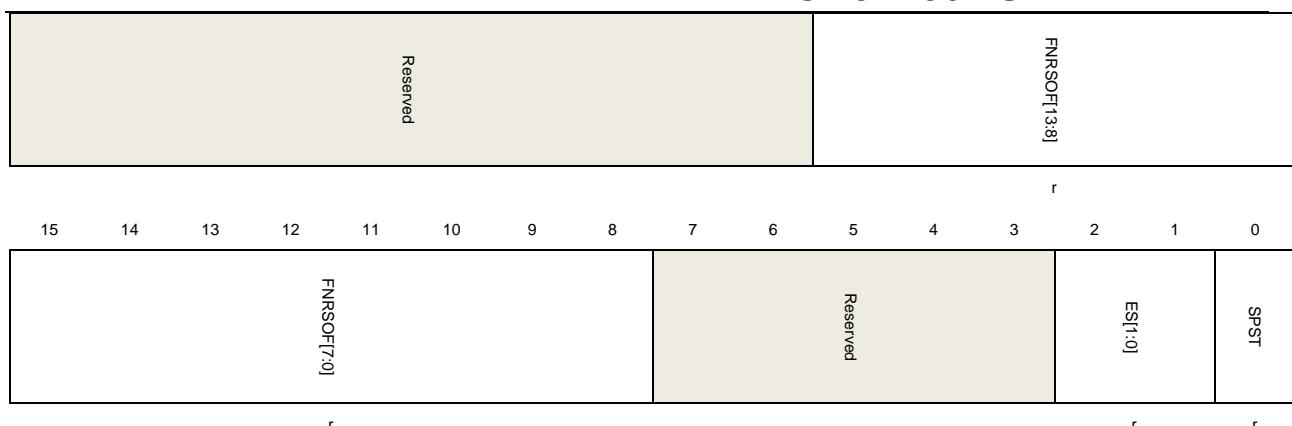
### **Device status register (USBFS\_DSTAT)**

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 in suspend state.  1: Device is not in suspend state.

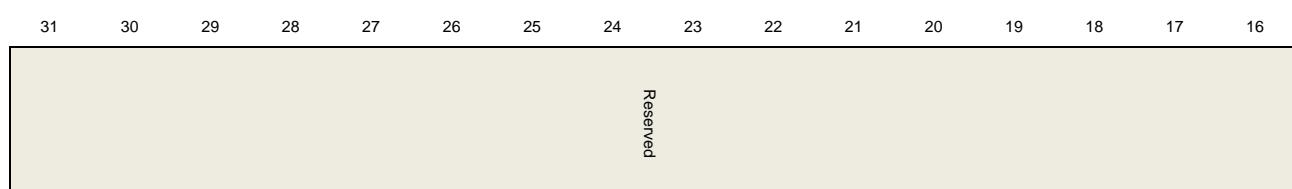
## Device IN endpoint common interrupt enable register (USBFS\_DIEPINTEN)

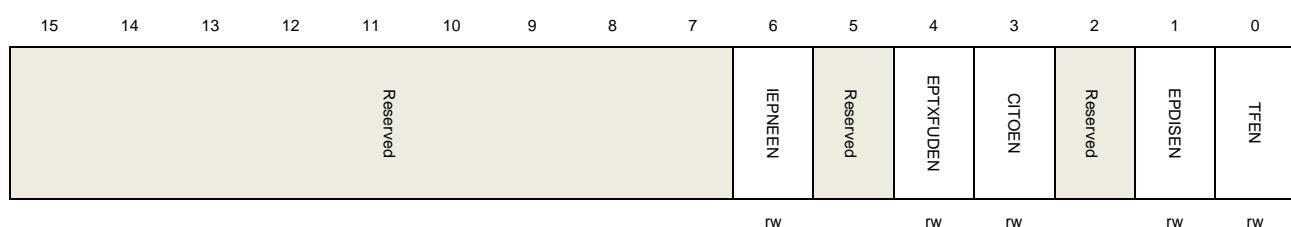
Address offset: 0x810

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

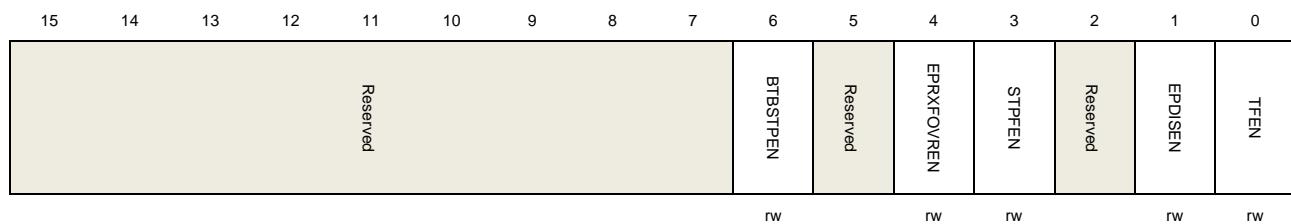
### Device OUT endpoint common interrupt enable register (USBFS\_DOEPINTEN)

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	BTBSPEN	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

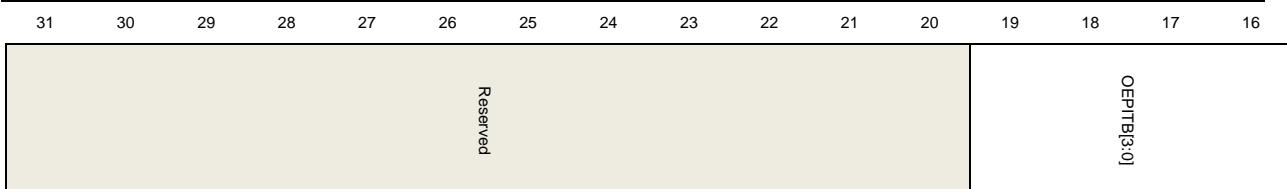
### Device all endpoints interrupt register (USBFS\_DAEPIR)

Address offset: 0x0818

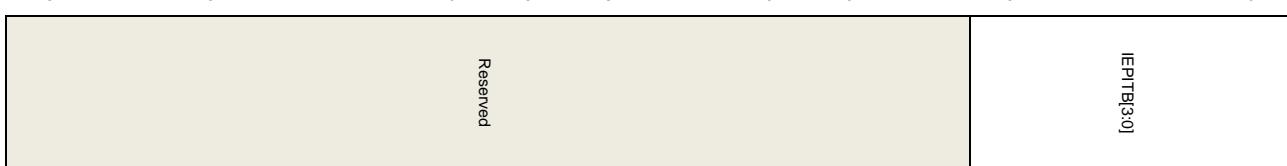
Reset value: 0x0000 0000

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)



r



r

Bits	Fields	Descriptions
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\_DAEPISEN)

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)



rw



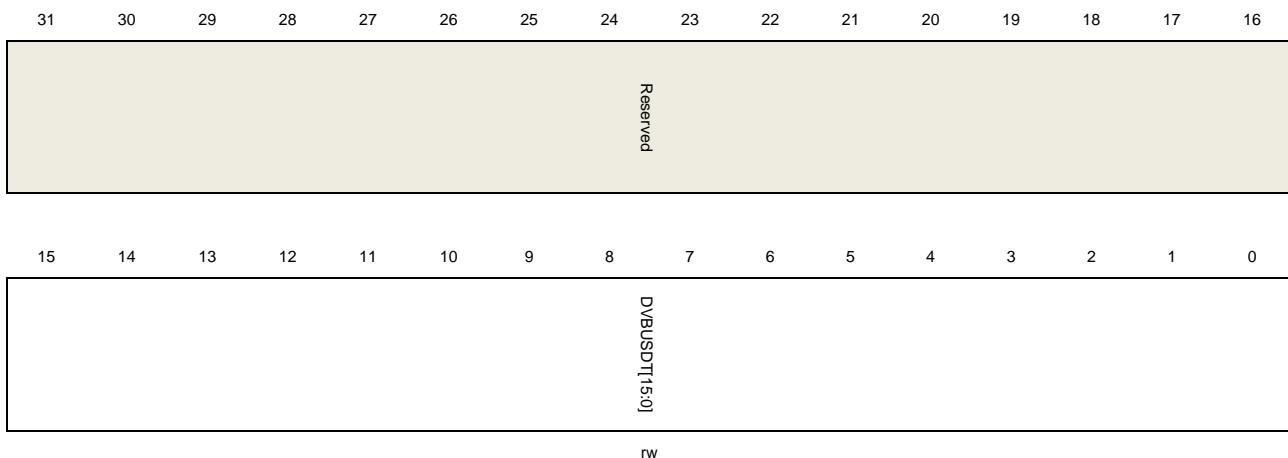
<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:20	Reserved	Must be kept at reset value
19:16	OEPIE[3:0]	Out endpoint interrupt enable 0: Disable OUT endpoint-n interrupt 1: Enable OUT endpoint-n interrupt Each bit represents an OUT endpoint: Bit 16 for OUT endpoint 0, bit 19 for OUT endpoint 3.
15:4	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)



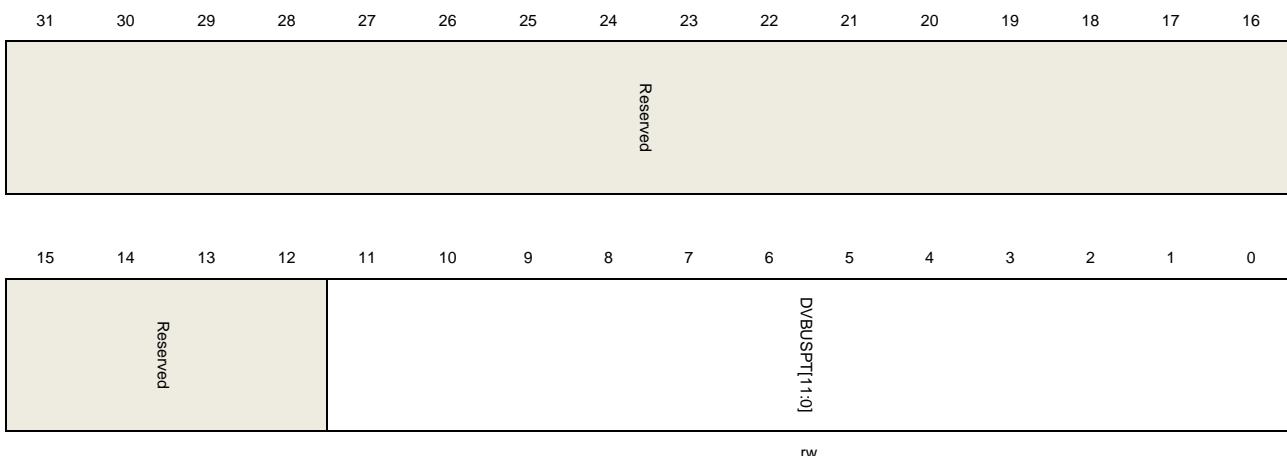
<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
31:16	Reserved	Must be kept at reset value
15:0	DVBUSDT[15:0]	Device VBUS discharge time There is a discharge process after V <sub>BUS</sub> pulsing in SRP protocol. This field defines the discharge time of V <sub>BUS</sub> . The true discharge time is 1024 * DVBUSDT[15:0] *T <sub>USB</sub> CLOCK, where T <sub>USB</sub> CLOCK 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)



Bits	Fields	Descriptions
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 $V_{BUS}$ . The true pulsing time is $1024 \times DVBUSPT[11:0] \times T_{USBCLOCK}$ , where $T_{USBCLOCK}$ is the period time of USB clock.

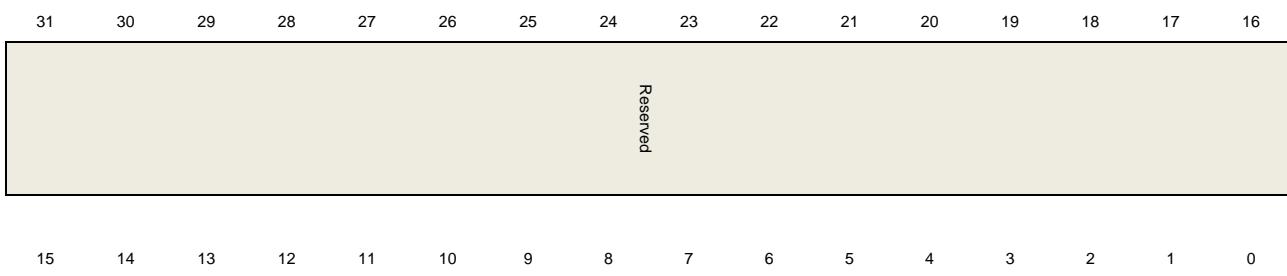
**Device IN endpoint FIFO empty interrupt enable register (USBFS\_DIEPFEINTEN)**

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)



Reserved	IEPTXFEIE[3:0]
rw	

Bits	Fields	Descriptions
31:4	Reserved	Must be kept at reset value
3:0	IEPTXFEIE[3:0]	<p>IN endpoint Tx FIFO empty interrupt enable bits</p> <p>This field controls whether the TXFE bits in USBFS_DIEPxINTF registers are able to generate an endpoint interrupt bit in USBFS_DAEPINT register.</p> <p>Bit 0 for IN endpoint 0, bit 3 for IN endpoint 3</p> <p>0: Disable FIFO empty interrupt</p> <p>1: Enable FIFO empty interrupt</p>

### 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]		NAKS	Reserved
rs	rs		w	w		rw			rs	r	r	r	r		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EPACT								Reserved				MPL[1:0]			
r															rw

Bits	Fields	Descriptions
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]	<p>Maximum packet length</p> <p>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:</p> <ul style="list-style-type: none"> <li>00: 64 bytes</li> <li>01: 32 bytes</li> <li>10: 16 bytes</li> <li>11: 8 bytes</li> </ul>

### **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 <sup>SD1PID</sup>	SEVENFRM <sup>RM</sup>	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]				

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.
	SD1PID	Software sets this bit to set EOFRM bit in this register. 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]	<p>Tx FIFO number</p> <p>Defines the Tx FIFO number of this IN endpoint.</p>
21	STALL	<p>STALL handshake</p> <p>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.</p> <p>For control IN 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 IN endpoint:</p> <p>Only software can clear this bit</p>
20	Reserved	Must be kept at reset value
19:18	EPTYPE[1:0]	<p>Endpoint type</p> <p>This field defines the transfer type of this endpoint:</p> <ul style="list-style-type: none"> <li>00: Control</li> <li>01: Isochronous</li> <li>10: Bulk</li> <li>11: Interrupt</li> </ul>
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> <ul style="list-style-type: none"> <li>0: USBFS sends data or handshake packets according to the status of the endpoint's Tx FIFO.</li> <li>1: USBFS always sends NAK handshake to the IN token.</li> </ul> <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 IN endpoints)</p> <p>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.</p> <ul style="list-style-type: none"> <li>0: Only sends data in even frames</li> <li>1: Only sends data in odd frames</li> </ul>
	DPID	<p>Endpoint data PID (For interrupt/bulk IN endpoints)</p> <p>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.</p> <ul style="list-style-type: none"> <li>0: Data packet's PID is DATA0</li> </ul>

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	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	Reserved	Must be kept at reset value

---

21	STALL	STALL handshake  Set this bit to make USBFS send STALL handshake during an OUT transaction. 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.
20	SNOOP	Snoop mode  This bit controls the snoop mode of an OUT endpoint. In snoop mode, USBFS doesn't check the received data packet's CRC value. 0: Snoop mode disabled 1: Snoop mode enabled
19:18	EPTYPE[1:0]	Endpoint type  This field is fixed to '00' for control endpoint.
17	NAKS	NAK status  This bit controls the NAK status of USBFS when both STALL bit in this register and GONS bit in USBFS_DCTL register are cleared: 0: USBFS sends data or handshake packets according to the status of the endpoint's Rx FIFO. 1: USBFS always sends NAK handshake for the OUT token.  This bit is read-only and software should use CNAK and SNAK in this register to control this bit.
16	Reserved	Must be kept at reset value
15	EPACT	Endpoint active  This field is fixed to '1' for endpoint 0.
14:2	Reserved	Must be kept at reset value
1:0	MPL[1:0]	Maximum packet length  This is a read-only field, and its value comes from the MPL field of USBFS_DIEP0CTL register: 00: 64 bytes 01: 32 bytes 10: 16 bytes 11: 8 bytes

**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	EOFRMDPID	
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				MP1[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	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.
	SD1PID	Set DATA1 PID (For interrupt/bulk OUT endpoints) Software sets this bit to set DPID bit in this register.
28	SEVENFRM	Set even frame (For isochronous OUT endpoints) Software sets this bit to clear EOFRM bit in this register.
	SD0PID	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	Clear NAK Software sets this bit to clear NAKS bit in this register.
25:22	Reserved	Must be kept at reset value
21	STALL	STALL handshake 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.

For control OUT 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 OUT endpoint:

Only software can clear this bit.

20	SNOOP	Snoop mode  This bit controls the snoop mode of an OUT endpoint. In snoop mode, USBFS doesn't check the received data packet's CRC value.  0: Snoop mode disabled 1: Snoop mode enabled
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 GONS bit in USBFS_DCTL register are cleared:  0: USBFS sends handshake packets according to the status of the endpoint's Rx FIFO. 1: USBFS always sends NAK handshake to the OUT 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 OUT endpoints)  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.  0: Only sends data in even frames 1: Only sends data in odd frames
	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  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.

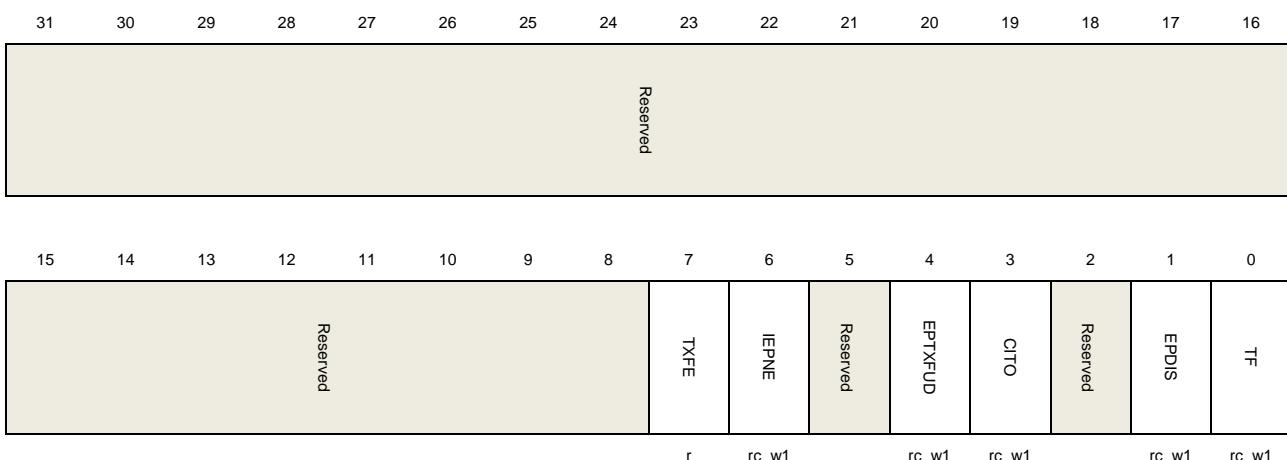
### **Device IN endpoint-x interrupt flag register (USBFS\_DIEPxINTF) (x = 0..3, where x = endpoint\_number)**

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.

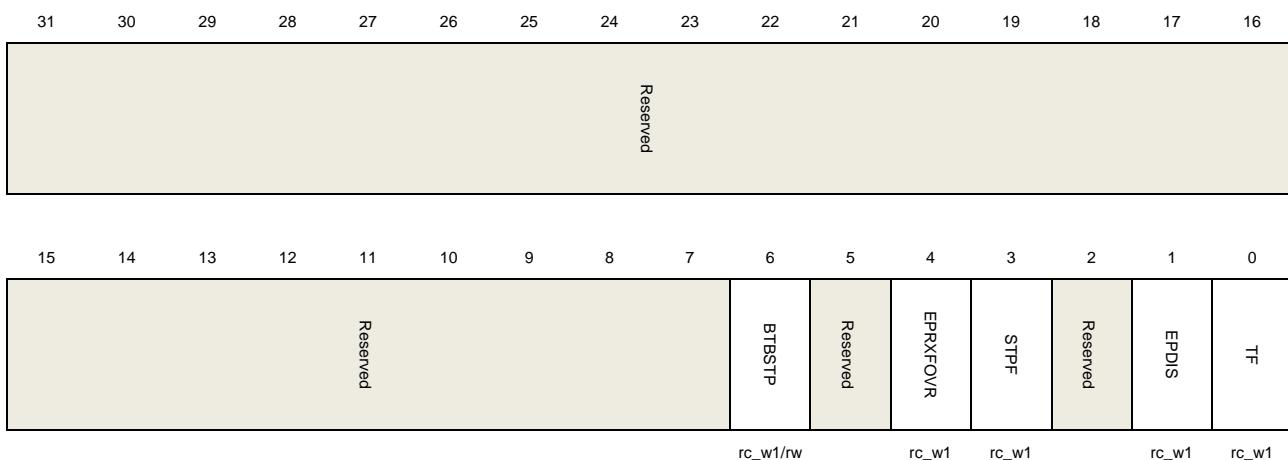
**Device OUT endpoint-x interrupt flag register (USBFS\_DOEPxINTF) (x = 0..3,  
where x = endpoint\_number)**

Address offset: 0xB08 + (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.

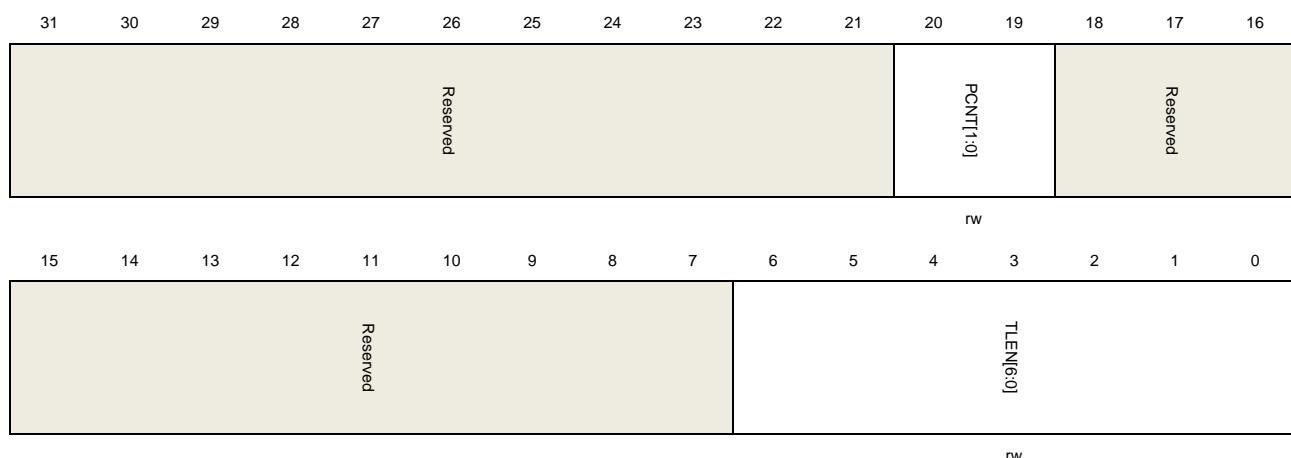
3	STPF	SETUP phase finished (Only for control OUT endpoint) This flag is triggered when a setup phase finished, i.e. USBFS receives an IN or OUT token after a setup token.
2	Reserved	Must be kept at reset value
1	EPDIS	Endpoint disabled This flag is triggered when an endpoint is disabled by the software's 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]	Transfer length

The total data byte number of a transfer.

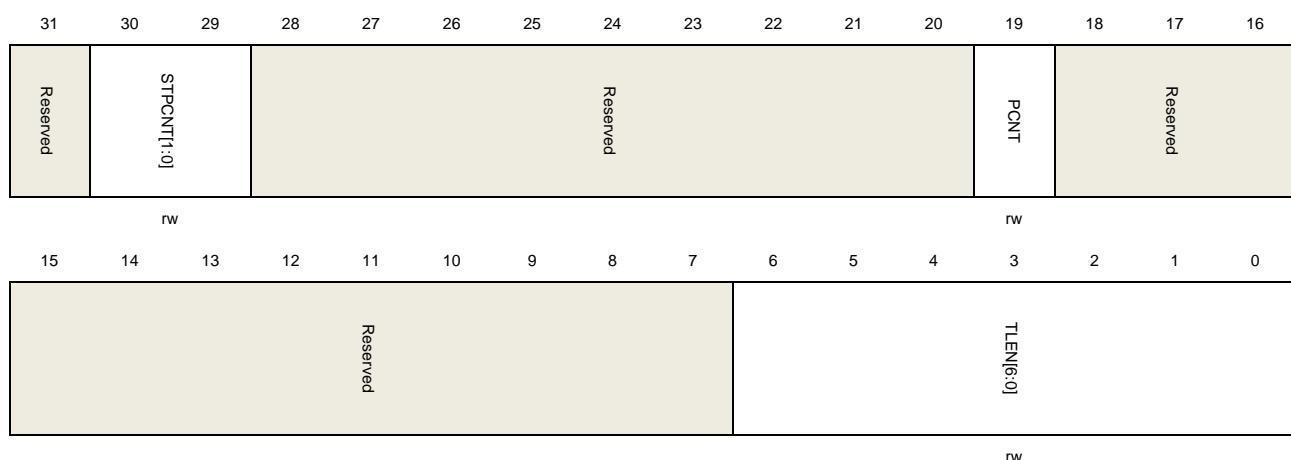
This field is the total data bytes of all the data packets desired to be transmitted in an IN transfer. Program this field before the endpoint is enabled. When software successfully writes a packet into the endpoint's Tx FIFO, this field is decreased by the byte size of the packet.

### **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)



<b>Bits</b>	<b>Fields</b>	<b>Descriptions</b>
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 BTBSTOP flag in USBFS_DOEP0INTF register will be triggered.</p> <p>00: 0 packet 01: 1 packet 10: 2 packets 11: 3 packets</p>
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</p>

on bus.

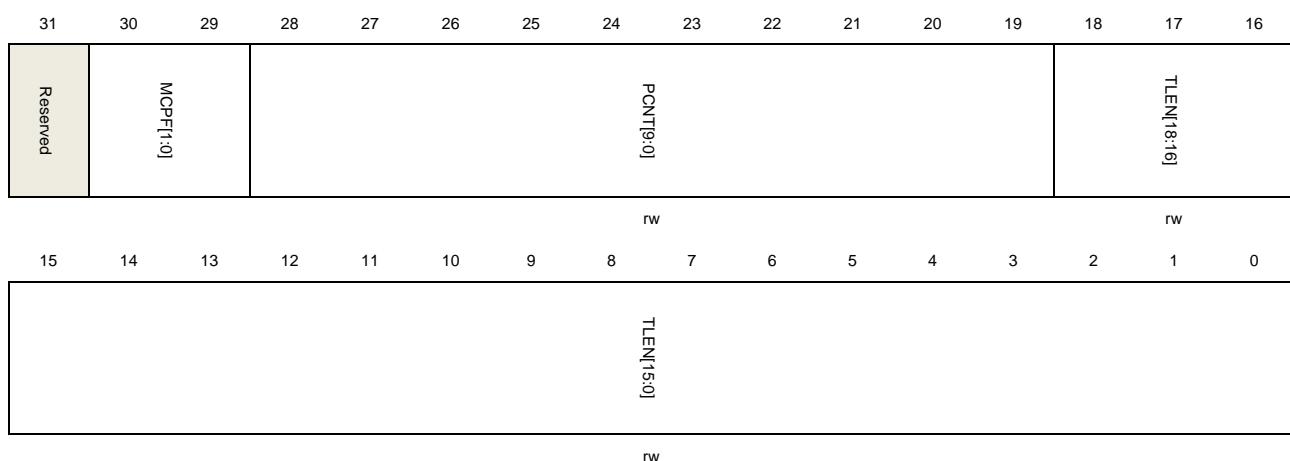
18:7	Reserved	Must be kept at reset value
6:0	TLEN[6:0]	<p>Transfer length</p> <p>The total data byte number of a transfer.</p> <p>This field is the total data bytes of all the data packets desired to receive in an OUT transfer. Program this field before the endpoint is enabled. Each time software reads 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> <p>10: 2 packets</p> <p>11: 3 packets</p>
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</p>

transmission.

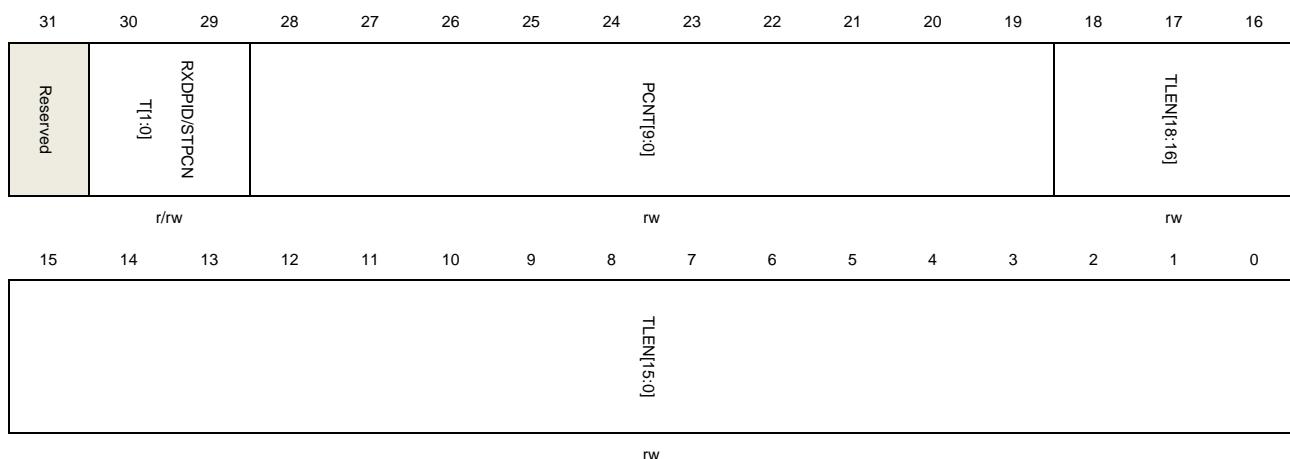
18:0	TLEN[18:0]	Transfer length The total data byte number of a transfer. This field is the total data bytes of all the data packets desired to be transmitted in an IN transfer. Program this field before the endpoint is enabled. When software successfully writes a packet into the endpoint's Tx FIFO, this field is decreased by the byte size of the packet.
------	------------	--

### 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]	Received data PID (For isochronous OUT endpoints) This field saves the PID of the latest received data packet on this endpoint. 00: DATA0 10: DATA1 Others: Reserved
	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]	<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>
18:0	TLEN[18:0]	<p>Transfer length</p> <p>The total data byte 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 after software reads out a packet from the RxFIFO, this field is decreased by the byte size of the packet.</p>

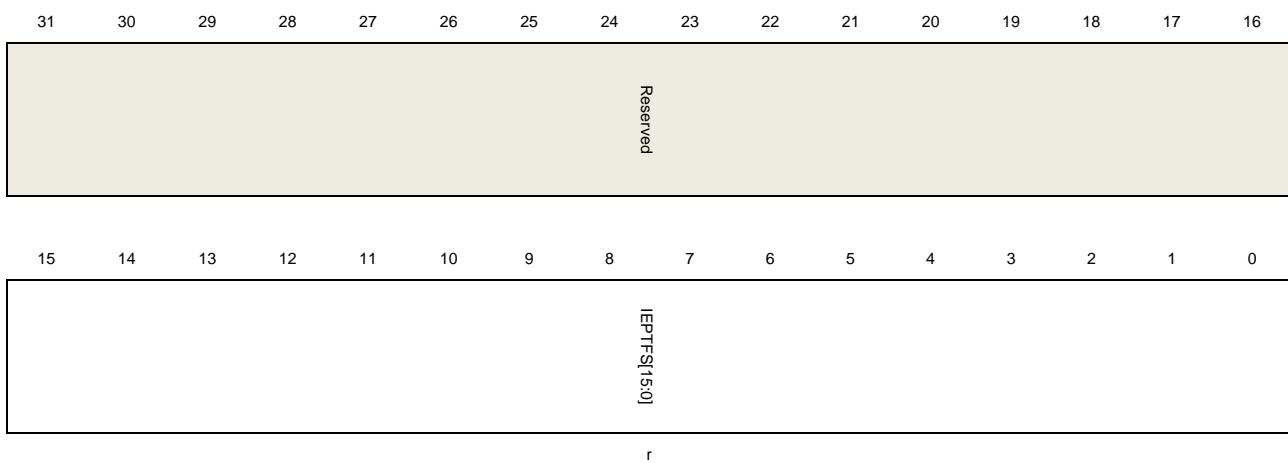
**Device IN endpoint-x transmit FIFO status register (USBFS\_DIEPxTFSTAT) (x = 0..3, where x = endpoint\_number)**

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)



Bits	Fields	Descriptions
31:16	Reserved	Must be kept at reset value
15:0	IEPTFS[15:0]	<p>IN endpoint's Tx FIFO space remaining</p> <p>IN endpoint's Tx FIFO space remaining in 32-bit words:</p> <p>0: FIFO is full</p> <p>1: 1 word available</p> <p>...</p>

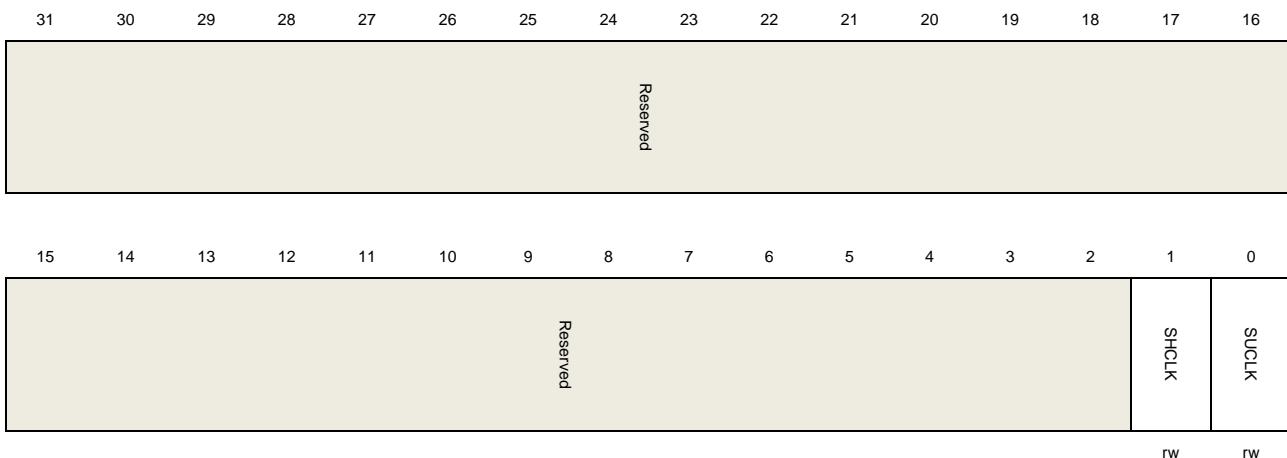
n: n words available

### 25.7.4. Power and clock control register (USBFS\_PWRCLKCTL)

Address offset: 0x0E00

Reset value: 0x0000 0000

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



Bits	Fields	Descriptions
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

## 26. Revision history

**Table 26-1. Revision history**

Revision No.	Description	Date
1.0	Initial Release	Feb.10, 2017
1.1	1. Revise ADC/DAC/DMA/RCU/USART	Oct. 17, 2017
2.0	1. Adapt To New Document Specification	Dec.14, 2018
2.1	1. Add Operation process of ADC Temperature Sensor, refer to <a href="#"><u>Temperature sensor, and internal reference voltage VREFINT</u></a>	Jun.10, 2019
2.2	1. In PMU chapter 3.3, update block diagram of PMU, refers to <a href="#"><u>Figure 3-1. Power supply overview</u></a> , Update description of PMU about ADC\DAC\VREF, refers to <a href="#"><u>VDDA domain</u></a> , Update description of sleep mode about WFE wakeup, refers to <a href="#"><u>Sleep mode</u></a> 2. In WDGT chapter 14.1.4, modify the register attribute description from 'ro, wo' to 'r, w', refers to <a href="#"><u>Register definition</u></a> 3. In ENET chapter 23.3.6, update description of ERSS bit about receive descriptor, refers to <a href="#"><u>RxDMA descriptors in normal mode</u></a>	Oct.10, 2019
2.3	1. In USB chapter 24.7.2, modify the description of USBD_INTF register DIR bit, refers to <a href="#"><u>USBD interrupt flag register (USBD INTF)</u></a> 2. In I2C chapter 18.3.6, Update figure 18-6, refers to <a href="#"><u>Figure 18-6. I2C communication flow with 7-bit address</u></a>	Mar.16, 2020
2.4	1. In CAN chapter 22.3.7, modify CAN baud rate calculation formula, refers to <a href="#"><u>Baud rate</u></a> 2. In I2C chapter 18.3.11, change SMBTYPE to SMBSEL, refers to <a href="#"><u>SMBus support</u></a> 3. In WDGT chapter 14.1.3, add notes for entering deep sleep or standby mode immediately after feeding the dog, refers to <a href="#"><u>Function overview</u></a> 4. In chapter one, update table 1-2, integrate the boot loader address together, refers to <a href="#"><u>Table 1-2. Memory map of GD32F30x devices</u></a> 5. In ADC chapter 12.4.3, add notes about the delay after ADC startup, refers to <a href="#"><u>ADCON switch</u></a>	Jun.30, 2020
2.5	1. In SDIO chapter 20.5.5, correction of clerical errors in table 20-27 from '4MB1' to '4MB', refers to <a href="#"><u>Table 20-27. Maximum AU size</u></a> 2. In TIMERx chapter 16.2.4, update figure 16-40, refers to	Dec.16, 2020

	<p><b><u>Figure 16-37.</u></b>, TIMERx(x = 1,2,3,4) registers, modify the description of bits[9:8], refers to <b><u>Channel control register 0 (TIMERx CHCTL0)</u></b> and <b><u>Channel control register 1 (TIMERx CHCTL1)</u></b>, TIMERx(x = 8,11) registers modified, refers to <b><u>Channel control register 0 (TIMERx CHCTL0)</u></b>, TIMERx(x = 0~4,7,8,11) registers, modify the description of bits[2:0], refers to <b><u>Slave mode configuration register (TIMERx SMCFG)</u></b></p> <p>3. In PMU chapter 3.3.2 VDDA domain, when the value of VDDA and VDD is different, VDDA should be no more than 0.3V higher than VDD, refers to <b><u>VDDA domain</u></b></p> <p>4. In USB chapter 25.5.2, The width of pulse signal generated by USB for each SOF packet is changed from 16 HCLK to 12 HCLK, refers to <b><u>USB host function</u></b></p>	
2.6	<p>1. In chapter 18.3.10, correct the condition of bit set for auto send PEC in DMA mode, refers to <b><u>Packet error checking</u></b></p> <p>2. In chapter 22.4.17-22.4.22, add the description of fliter register only be used by CAN0, refers to <b><u>22.4.17-22.4.22</u></b></p> <p>3. In chapter 23.3.5, the method of modifying the enable PTP output signal is described as setting bit 30 of the AFIO PCF0 register, refers to <b><u>PTP pulse-per-second (PPS) output signal</u></b></p> <p>4. Modify the description of SMBALT bit in I2C_STAT0 register</p> <p>5. In chapter 18.3.6, correct the word in <b><u>Figure 18-7</u></b> and <b><u>Figure 18-8</u></b></p> <p>6. In chapter 21.3, modify the <b><u>Figure 21-21</u></b>, <b><u>Figure 21-22</u></b> and <b><u>Figure 21-23</u></b></p>	Jun.30, 2021
2.7	<p>1. Modify the value of max timeout and min timeout in <b><u>Table 14-1</u></b></p> <p>2. Modify reset value of ENET_PTP_TSCTL, from 0x0000 0000 to 0x0000 2000</p> <p>3. Delete the description of ETM</p> <p>4. Modify the description of the note below <b><u>Table 8-10</u></b></p> <p>5. Modify the description of bit15 in the AFIO_PCF0</p> <p>6. Modify the description of system memory is map to 0x0000 0000</p> <p>7. Modify the description of external crystals parameter from 3-25M to 4-32M</p> <p>8. Modify the description of ADDSEND bit and MASTER bit in the I2C_STAT0 and I2C_STAT1</p> <p>9. Add the description of HD,XD,CL to system chapter</p>	Dec.30, 2021
2.8	1. Correct the description of register of RCU_DSV	Jul.14, 2022

- |  |   |  |
|--|---|--|
|  | <ul style="list-style-type: none"><li>2. Update for consistency of description across series</li><li>3. Modify the register name of ENET_MAC_PHY_DATA</li><li>4. Delete bit field of TRACE_MODE</li><li>5. Modify context from ADC to ADCs in the system diagram</li><li>6. Modify flash range of GD32F30X_HD from 128KB to 512KB</li></ul> |  |
|--|---|--|

- 2. Update for consistency of description across series
- 3. Modify the register name of ENET\_MAC\_PHY\_DATA
- 4. Delete bit field of TRACE\_MODE
- 5. Modify context from ADC to ADCs in the system diagram
- 6. Modify flash range of GD32F30X\_HD from 128KB to 512KB

## 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 it's 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 it's 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.