



AC781x Reference Manual

Version: 1.2
Release date: 2022-01-06

© 2013 AutoChips Inc.

This document contains information that is proprietary to AutoChips Inc.

Unauthorized reproduction or disclosure of this information in whole or in part is strictly prohibited.

Specifications are subject to change without notice.

Document Revision History

Revision	Date	Author	Description
1.0	2018-03-31	AutoChips	Initial Version
1.01	2020-05-15	AutoChips	<ol style="list-style-type: none"> 1. Modify the I2C rack bit attribute from R to RW 2. Modify RTC some register bit attribute from R to RW
1.1	2021-06-16	AutoChips	<ol style="list-style-type: none"> 1. Add description of which UART module supporting LIN in chapter 1.1 and 9.1 2. Add description of MPU in chapter 2.1 3. Update option byte end offset address changed from 0x14 to 0x2F in chapter 2.2.1 4. Add description of Chip Model Information in chapter 2.3.6 5. Add description of Chip UUID Information in chapter 2.3.7 6. Update the description of KOER register in chapter 7.4 and 7.5 7. Change description of the DR bit in UART_LSR1 in chapter 9.12 8. Update description for UARTn_SLADDR in chapter 9.12 9. Modify the register of bit width in chapter 9.12 10. Modify the description of RS485 delay in chapter 9.12 11. Update the description for IDLE flag in chapter 9.12 12. Add description of ADC conversion format in chapter 10 13. Modify mode5 sequence figure error in chapter 10.3 14. Modify synchronization description in chapter 12 15. Modify "timer channel" to "timer" in chapter 14 16. Update the description of GPIO_ODR register in table 17-5 17. Update the description of GPIO_E4_E2 register in chapter 17.5 18. Add description of typical value for pull-up, pull-down resistor in chapter 17.5 19. Modify the default value of SPI_CFG1 register in chapter 19.14 20. Update the description of SPI_STATUS register in chapter 19.14 21. Modify the description of Byte mode in table 20-2 22. Update the description of DMA operation mode in chapter 20.4 23. Modify the description of DMA_MEM_END_ADDR register in chapter 20.6 24. Add the description of disable DMA in circular mode in chapter 20.6 25. Add description "The minimum programming bit width is 32 bits, and the programming address needs to be aligned with 4 bytes" in chapter 23.1.2 26. Update register eFLASH_SR1 reset value to 0x702 in chapter 23.5 27. Update the description of the register eFLASH_CTR2 CKDIV

			bit in chapter 23.5
1.2	2022-01-06	AutoChips	<ol style="list-style-type: none">1. Add description of ISP download pin in chapter 2.3 112. Modify description of BUSOFF bit in chapter 7.43. Add description of UART baud rate range in chapter 9.24. Add chapter 12.3.11 for PWM registers updated from write buffers5. Add chapter 12.3.22 for PWM features priority6. Add chapter 17.3 for GPIO block diagram7. Add description of SPI baud rate range in chapter 19.28. Add description of SPI TXEIO, RXFIE in chapter 19.149. Update description of DMA priority in chapter 20.5.210. Update description of DMA end address in chapter 20.5.3.3 and chapter 20.6

Legal Notice

This reference manual contains information that is confidential to AUTOCHIPS Inc. Unauthorized use or disclosure of the information contained herein is prohibited. You may be held responsible for any loss or damages suffered by Autochips Inc. for your unauthorized disclosure hereof, in whole or in part.

Information herein is subject to change without noticed. AUTOCHIPS Inc. does not assume any responsibility for any use of, or reliance on, the information contained herein.

THIS REFERENCE MANUAL AND ALL INFORMATION CONTAINED HEREIN IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE. AUTOCHIPS INC. SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE. NEITHER DOES AUTOCHIPS INC. PROVIDE ANY WARRANTY WHATSOEVER WITH RESPECT TO THE SOFTWARE OF ANY THIRD PARTY WHICH MAY BE USED BY, INCORPORATED IN, OR SUPPLIED WITH THIS REFERENCE MANUAL, AND USER AGREES TO LOOK ONLY TO SUCH THIRD PARTY FOR ANY WARRANTY CLAIM RELATING THERETO. AUTOCHIPS INC. SHALL ALSO NOT BE RESPONSIBLE FOR ANY AUTOCHIPS DELIBERABLES MADE TO USER'S SPECIFICATION OR TO CONFORM TO A PARTICULAR STANDARD OR OPEN FORUM.

Table of Contents

Document Revision History	2
Legal Notice	4
Table of Contents	5
List of Figures	16
List of Tables	19
1 Introduction	21
1.1 Overview	21
1.2 Module description	21
2 Memory and Bus Architecture	22
2.1 System architecture	22
2.2 Memory organization	24
2.2.1 Memory map	24
2.3 Detail function description	25
2.3.1 Internal SRAM	25
2.3.2 Bit banding	25
2.3.3 Fast IO memory map	26
2.3.4 Internal flash memory	26
2.3.5 Read internal flash memory	26
2.3.6 Chip Model	27
2.3.7 Chip UUID	27
2.3.8 I-Cache and D-Cache	27
2.3.9 AHB to APB bridge	27
2.3.10 Nested Vectored Interrupt Controller (NVIC)	27
2.3.11 Boot configuration	29
2.4 Address Table	30
3 Reset	32
3.1 Introduction	32

3.2	Block diagram	32
3.3	Reset detail function description	33
3.3.1	Power On Reset (POR).....	33
3.3.2	System reset source.....	33
3.4	Register mapping	34
4	Clock.....	36
4.1	Introduction	36
4.2	Clock control diagram.....	36
4.3	System Clock.....	37
4.4	Register mapping	38
5	Power Modes	46
5.1	Introduction	46
5.2	Power modes.....	46
5.3	Entering and exiting power modes	46
5.4	Module operation in low-power modes.....	46
6	System Power Management.....	48
6.1	Introduction	48
6.2	Feature list.....	48
6.3	Block diagram	48
6.4	Application notes	49
6.4.1	SPM power control program guide.....	49
6.4.2	XOSC/SYSPLL power control.....	49
6.5	Register mapping	50
7	CAN	58
7.1	Introduction	58
7.1.1	The CAN-CTRL core.....	58
7.1.2	The CAN protocol.....	58
7.2	Feature list.....	59
7.3	Message buffers	59

7.3.1	Message buffers concept	59
7.3.2	Receive buffer	60
7.3.3	Transmit buffer	60
7.4	Register definition	61
7.5	General operation	73
7.5.1	The bus off state	73
7.5.2	Acceptance filters	73
7.5.3	Message reception	74
7.5.4	Handling message receptions	74
7.5.5	Message transmission	75
7.5.6	Message transmission abort	76
7.5.7	A full STB	77
7.5.8	Extended status and error report	77
7.5.9	Extended features	78
7.5.10	Software reset	80
7.5.1	CAN bit time	83
8	LIN	86
8.1	Introduction	86
8.2	Feature list	86
8.3	Block diagram	86
8.4	Application note	87
8.4.1	Operating modes	87
8.4.2	Test modes	88
8.4.3	Baud rate generation	89
8.4.4	Error detection	89
8.4.5	Interrupt table	90
8.4.6	Master mode	90
8.4.7	Slave mode	91
8.4.8	Register definition	92

9	UART.....	103
9.1	Introduction	103
9.2	Features.....	103
9.3	Block diagram	104
9.4	Input & Output timing	104
9.5	Basic UART functions.....	105
9.5.1	Noise detection.....	105
9.5.2	Baud rate description	106
9.6	Hardware flow control function	107
9.7	RS485 function	107
9.8	LIN function.....	108
9.9	Two chip power mode	109
9.10	Baud rate configuration description	110
9.11	Configure note	111
9.12	Register Definition	112
10	ADC.....	122
10.1	ADC introduction	122
10.2	ADC features	122
10.3	ADC functional description	122
10.3.1	ADC power on sequence	123
10.3.2	ADC power modes	124
10.3.3	ADC operation modes	125
10.4	Analog monitor	130
10.5	Status flag description	132
10.6	Calibration.....	134
10.7	Sampling conversion time	134
10.8	Temperature sensor	134
10.9	DMA access.....	134
10.10	Program guide	135

10.10.1	Normal power mode	135
10.10.2	Low power mode	135
10.11	ADC register definition	135
11	ACMP	142
11.1	Introduction	142
11.2	Features	142
11.2.1	Block diagram	142
11.3	Functional description	143
11.3.1	ACMP0	143
11.3.2	ACMP1	144
11.3.3	Low power wakeup	144
11.4	Memory map and register definition	144
11.5	Interrupts	153
12	PWM	154
12.1	Introduction	154
12.1.1	PWM features	154
12.1.2	Block diagram	155
12.2	Memory map and register definition	157
12.3	Functional description	181
12.3.1	Clock source	181
12.3.2	Prescaler	182
12.3.3	Counter	182
12.3.4	Operation mode	183
12.3.5	Input capture mode	185
12.3.6	Output Compare mode	185
12.3.7	Edge-Aligned PWM(EPWM) mode	186
12.3.8	Center-Aligned PWM(CPWM) mode	187
12.3.9	Combine mode	187
12.3.10	Complementary mode	188

12.3.11	Registers updated from write buffers	189
12.3.12	PWM synchronization.....	190
12.3.13	Inverting.....	190
12.3.14	Channel trigger output.....	191
12.3.15	Initialization trigger	191
12.3.16	Capture Test mode.....	191
12.3.17	Dual Edge Capture mode.....	191
12.3.18	Software output control	192
12.3.19	Deadtime insertion	193
12.3.20	Fault control.....	193
12.3.21	Polarity control.....	194
12.3.22	Features priority	194
12.4	PWM interrupts	195
12.4.1	Count Overflow interrupt	195
12.4.2	Channel interrupt.....	195
12.4.3	Fault interrupt	195
13	Pulse Width Detect Timer(PWDT)	196
13.1	Introduction	196
13.2	Features.....	196
13.3	Functional description.....	196
13.3.1	Pulse Width Measurement function	197
13.3.2	Timer function.....	200
13.3.3	Reset operation	201
13.4	Program guide	201
13.4.1	Pulse Width Measurement function program guide	201
13.4.2	Timer function program guide	201
13.5	Register definition	202
14	TIMER.....	205
14.1	Introduction	205

14.2	Features.....	205
14.3	Functional description.....	205
14.3.1	General operation.....	206
14.3.2	Chained timers	206
14.4	Memory map and register definition	206
14.5	Interrupts.....	209
15	CTU.....	210
15.1	Introduction	210
15.2	Features.....	210
15.3	Function description	210
15.3.1	ACMP output capture	211
15.3.2	ADC hardware trigger.....	211
15.4	Register definition.....	211
16	CRC.....	215
16.1	Introduction	215
16.2	Features.....	215
16.3	Block diagram	215
16.4	Memory map and register descriptions	215
16.5	Functional description.....	218
16.5.1	CRC initialization/reinitialization	218
16.5.2	CRC calculations.....	218
16.5.3	Transpose feature	218
16.5.4	Types of transpose.....	218
16.5.5	CRC result complement	220
16.5.6	CRC data register (CRC_DATA).....	220
16.5.7	CRC Polynomial register (CRC_GPOLY)	220
16.5.8	CRC Control register (CRC_CTRL)	220
16.6	Program guide	220
16.6.1	16-bit CRC.....	220

16.6.2	32-bit CRC	221
17	GPIO	222
17.1	Introduction	222
17.2	Features	222
17.3	Block diagram	223
17.4	Mode of operation	223
17.4.1	External interrupt	223
17.4.2	Multi-Function	225
17.5	Application note	228
17.5.1	External interrupt	228
17.5.2	Multi-Function	228
17.5.3	GPIO function	228
17.5.4	Programming guide	228
17.6	Memory map and register descriptions	229
18	I2C	238
18.1	Introduction	238
18.2	Features	238
18.3	Block diagram	238
18.4	Mode of operation	239
18.5	Memory map and register descriptions	239
18.6	Functional description	248
18.6.1	Data flow & Algorithm	248
18.6.2	Input & Output timing	248
18.6.3	Master SCL output timing set	249
18.6.4	Data transmission	249
18.6.5	DMA operation	250
18.6.6	Slave low power wakeup	252
18.6.7	Interrupt	252
18.7	Program guide	253

18.7.1	Description	253
18.7.2	Master transmitter	253
18.7.3	Master receiver.....	254
18.7.4	Master combined format	254
18.7.5	Slave.....	255
19	SPI	256
19.1	Introduction	256
19.2	Feature list.....	256
19.3	Block diagram	257
19.4	Data flow & Algorithm	257
19.5	Input & Output timing	258
19.5.1	CPHA = 0 transfer format.....	258
19.5.2	CPHA = 1 transfer format.....	259
19.6	Master SCK output timing set.....	260
19.7	Master mode fault detect	260
19.8	Slave low power wakeup	261
19.9	Interrupt	262
19.10	Master CS continuous mode	262
19.11	Master CS discontinuous output.....	263
19.12	Slave mode.....	263
19.13	DMA mode.....	263
19.14	Register definition	264
20	DMA.....	269
20.1	Introduction	269
20.2	Features.....	269
20.3	Block diagram	270
20.4	Mode of operations	270
20.5	Function description	270
20.5.1	DMA request mapping.....	270

20.5.2	DMA arbiter	270
20.5.3	Configuration guide	271
20.6	Memory map and register descriptions	273
21	WDT	282
21.1	Introduction	282
21.2	Features.....	282
21.3	Block diagram	283
21.4	Mode of operation.....	283
21.5	Memory map and register descriptions	283
21.6	Functional description.....	287
21.6.1	Watchdog refresh mechanism	287
21.6.2	Configuring the Watchdog.....	287
21.6.3	Using interrupts to delay watchdog reset	287
21.6.4	Backup reset	288
22	RTC	289
22.1	Introduction	289
22.2	Features.....	289
22.3	Block diagram	290
22.4	Mode of operation.....	290
22.5	Memory map and register descriptions	290
22.6	Functional description.....	295
22.6.1	Low power mode operation	295
22.6.2	RTC backup registers.....	296
22.6.3	Configuration sequence	296
23	Embedded Flash.....	297
23.1	Embedded flash function overview.....	297
23.1.1	Introduction.....	297
23.1.2	Feature list.....	297
23.1.3	Block diagram.....	297

23.1.4	Data flow & Algorithm	298
23.2	Embedded flash memory organization	298
23.3	Embedded flash protect	299
23.3.1	Read and write protection	299
23.3.2	Others	300
23.4	Program guide	300
23.4.1	Brief introduction	300
23.4.2	Command operation description	300
23.5	Register definition	308
24	Serial Flash	316
24.1	Serial FLASH function overview	316
24.1.1	Introduction	316
24.1.2	Feature list:	316
24.1.3	Block diagram	316
24.2	Application note	316
24.2.1	Clock setting	316
24.2.2	RISC access NOR flash	317
24.2.3	Read NOR flash id	317
24.2.4	Write status register (WRSR)	319
24.2.5	Erase NOR flash (sector/block erase)	323
24.2.6	Erase NOR flash (chip erase)	325
24.2.7	Page Program NOR flash	328
24.2.8	4 x I/O Page Program(4PP)	332
24.2.9	AAI program NOR flash	335
24.2.10	Read NOR flash	336
24.3	System boot from serial flash	348
24.4	Register definitions	349

List of Figures

Figure 2-1 System architecture	22
Figure 3-1 Reset block diagram	32
Figure 4-1 Clock control diagram	37
Figure 4-2 System clock diagram	37
Figure 6-1 SPM block diagram	48
Figure 7-1 Connection to CAN bus and main features of the CAN-CTRL core	58
Figure 7-2 Message buffers concept	60
Figure 7-3 Access to the acceptance filters	69
Figure 7-4 Schematic of the FIFO-like RB (example with 6 slots)	74
Figure 7-5 Schematic of PTB and STB in FIFO mode (empty PTB and 6 STB slots)	75
Figure 7-6 CAN Bit Timing Specifications	83
Figure 8-1 LINCORE block diagram	87
Figure 8-2 LINCORE operating modes	87
Figure 8-3 Loop back mode	88
Figure 8-4 Self-test mode	89
Figure 8-5 The structure of a frame	90
Figure 8-6 LIN bus structure	90
Figure 8-7 LIN header reception	91
Figure 9-1 UART block diagram	104
Figure 9-2 UART transmitter flow	105
Figure 9-3 UART receiver flow	105
Figure 9-4 UART noise detection	105
Figure 9-5 Hardware flow control connection	107
Figure 9-6 Hardware flow control principle	107
Figure 9-7 Single byte data transmission	108
Figure 9-8 Multiple bytes data transmission	108
Figure 9-9 Practical circuit connection	108
Figure 9-10 LIN frame flow	109
Figure 9-11 Chip normal mode and stop mode condition	109
Figure 9-12 Typical flow for waking up the chip by UART	110
Figure 9-13 Diagram for baud rate generator	110
Figure 10-1 UART block diagram	123
Figure 10-2 ADC Power on sequence	124
Figure 10-3 CPU normal mode and sleep mode condition	124
Figure 10-4 Regular group sequence	125
Figure 10-5 Valid regular group sequence	125
Figure 10-6 Injection group sequence	126
Figure 10-7 Valid injection group sequence	126
Figure 10-8 Mode 1 operation flow	126
Figure 10-9 Mode 2 operation flow	127
Figure 10-10 Mode 3 typical operation flow	127
Figure 10-11 Mode 3 operation flow with injection trigger at ADC idle state	127
Figure 10-12 Mode 4 operation flow with auto injection trigger	128
Figure 10-13 Mode 5 typical operation flow	128
Figure 10-14 Mode 5 operation flow with injection trigger at ADC idle state	129
Figure 10-15 Mode 6 operation flow	129
Figure 10-16 Mode 7 operation flow	130
Figure 10-17 Mode 8 operation flow	130
Figure 10-18 Analog monitor detecting region	131
Figure 10-19 Three flags under condition 1	132
Figure 10-20 Three flags under condition 2	133
Figure 10-21 Three flags under condition 3	133
Figure 11-1 ACMP block diagram	142

Figure 11-2 Polling mode	144
Figure 12-1 PWM block diagram	156
Figure 12-2 PWM clock source	181
Figure 12-3 Prescaler	182
Figure 12-4 Up counting	183
Figure 12-5 Up-Down counting	183
Figure 12-6 Input Capture mode	185
Figure 12-7 Output Compare mode	186
Figure 12-8 Edge-aligned PWM mode	186
Figure 12-9 Center-aligned PWM mode	187
Figure 12-10 Combine mode	188
Figure 12-11 Complementary mode	188
Figure 12-12 Dual edge capture mode	192
Figure 12-13 Features priority	194
Figure 13-1 PWDTC block diagram	197
Figure 13-2 Four basic measurement modes	198
Figure 13-3 Hall measurement modes	198
Figure 13-4 Two common installation ways	199
Figure 13-5 Example for low level noise and filter	199
Figure 13-6 Example for high level noise and filter	200
Figure 13-7 PWDTC counter and counting error	200
Figure 13-8 Modify the TIMCNTVAL between the TIMEN=0 and TIMEN=1	201
Figure 13-9 Modify the TIMCNTVAL during TIMEN=1	201
Figure 14-1 TIMER block diagram	205
Figure 15-1 CTU block diagram	210
Figure 16-1 CRC block diagram	215
Figure 16-2 CTRL[TOTW] / CTRL[TOTR] is 01	219
Figure 16-3 CTRL[TOTW] / CTRL[TOTR] is 10	219
Figure 16-4 CTRL[TOTW] / CTRL[TOTR] is 11	219
Figure 17-1 GPIO block diagram	223
Figure 17-2 GPIO external interrupt	224
Figure 17-3 GPIO multi-function	225
Figure 18-1 I2C block diagram	238
Figure 18-2 Data flow of transmitter	248
Figure 18-3 Data flow of Receiver	248
Figure 18-4 START and STOP conditions	249
Figure 18-5 Data transmission format	249
Figure 18-6 Baud rate generation	249
Figure 18-7 BND sequence of master write slave mode	250
Figure 18-8 BND sequence of master read slave mode	250
Figure 18-9 Master transmitter case 1	250
Figure 18-10 Master transmitter case 2	251
Figure 18-11 Master receiver	251
Figure 18-12 Slave transmitter	251
Figure 18-13 Slave receiver	252
Figure 18-14 Wakeup sequence	252
Figure 18-15 Master write operation sequence	253
Figure 18-16 Master receiver operation sequence	254
Figure 18-17 Combined format option sequence	254
Figure 18-18 Typical I2C slave interrupt routine	255
Figure 19-1 SPI system connection	256
Figure 19-2 SPI block diagram	257
Figure 19-3 Data flow of master	257
Figure 19-4 Data flow of slave	258
Figure 19-5 CPHA=0 Transmission Format	259
Figure 19-6 CPHA=1 Transmission Format	260
Figure 19-7 Baud rate generation	260

Figure 19-8 SCK output timing with mode fault detect enable.....	261
Figure 19-9 Limitation of mode fault detect.....	261
Figure 19-10 Wakeup sequence	262
Figure 19-11 CS continuous mode.....	262
Figure 20-1 DMA block diagram	270
Figure 20-2 DMA configuration guide	271
Figure 20-3 DMA channel circular.....	272
Figure 21-1 WDG block diagram	283
Figure 22-1 RTC block diagram.....	290
Figure 23-1 Block diagram for flash and flash controller	297
Figure 23-2 Data flow & algorithm for flash and flash controller	298
Figure 23-3 Page erase command operation flow.....	302
Figure 23-4 Mass erase command operation flow.....	303
Figure 23-5 Page program command operation flow	304
Figure 23-6 Page erase verify command operation flow	305
Figure 23-7 Mass erase verify command operation flow	306
Figure 23-8 Option page erase command operation flow.....	307
Figure 23-9 Option page program command operation flow	308
Figure 24-1 Diagram of module sflash controller.....	316
Figure 24-2 RISC access NOR flash	317
Figure 24-3 Read NOR flash id sequence	319
Figure 24-4 Write status register sequence	321
Figure 24-5 Set Quad Enable bit flow	323
Figure 24-6 Erase NOR flash(sector/block erase) flow	325
Figure 24-7 Erase NOR flash(chip erase) flow solution 1	327
Figure 24-8 Erase NOR flash(chip erase) flow solution 2.....	328
Figure 24-9 Buffer enable flow	330
Figure 24-10 Buffer disable flow	331
Figure 24-11 Buffer enable flow	333
Figure 24-12 Buffer disable flow	334
Figure 24-13 AAI program NOR flash	336
Figure 24-14 Normal read NOR flash flow	338
Figure 24-15 Fast read NOR flash flow	340
Figure 24-16 Dual read NOR flash flow	342
Figure 24-17 2 x I/O read NOR flash flow.....	344
Figure 24-18 Quad read NOR flash flow.....	346
Figure 24-19 4 x I/O read NOR flash flow.....	348

List of Tables

Table 1-1 AC781x module description	21
Table 2-1 High-level device memory map	24
Table 2-2 Interrupt table	27
Table 2-3 Boot configuration	29
Table 2-4 Address assignment of each peripheral	30
Table 3-1 Reset register map	34
Table 4-1 Clock register map	38
Table 5-1 Module functionality in low-power modes	46
Table 6-1 SPM Register Map	50
Table 7-1 CAN-CRTL Register Mapping	61
Table 7-2 Bits in Register ACF_3, if SELMASK=1	70
Table 7-3 Receive Buffer Registers RBUF – Standard Format (r-0)	71
Table 7-4 Receive Buffer Registers RBUF – Extended Format (r-0)	71
Table 7-5 Transmit Buffer Registers TBUF – Standard Format (rw-u)	71
Table 7-6 Transmit Buffer Registers TBUF – Extended Format (rw-u)	71
Table 7-7 Control bits in RBUF and TBUF	72
Table 7-8 Definition of the DLC (according to the CAN 2.0 specification)	72
Table 7-9 Software reset	81
Table 7-10 CAN Timing Segments	83
Table 7-11 CAN-CTRL Timing Settings	84
Table 7-12 Sample settings for 48MHz can_clk	84
Table 7-13 Sample settings for 8MHz can_clk	85
Table 8-1 LIN interrupt table	90
Table 8-2 LIN register map	92
Table 9-1 Function classification and configuration	103
Table 9-2 UART input&output timing	104
Table 9-3 Typical baud rate and error@bclock=50MHz	106
Table 9-4 Typical baud rate and error @bclock=36MHz	106
Table 9-5 UART register map	112
Table 10-1 Power modes	124
Table 10-2 ADC operation modes and its corresponding configuration	125
Table 10-3 Analog monitor configuration	131
Table 10-4 ADC register map	135
Table 11-1 ACMP register map	144
Table 11-2 ACMP interrupt table	153
Table 12-1 PWM modules configuration	154
Table 12-2 PWM register map and reset values	157
Table 12-3 Operation mode configuration	183
Table 12-4 PWM_CNTIN register update buffer	189
Table 12-5 PWM_CH(n)V register update buffer	189
Table 12-6 PWM_MCVR register update buffer	189
Table 12-7 Software output control when COMP bit is zero	192
Table 12-8 Software output control when COMP bit is one	193
Table 12-9 Fault source and number table	193
Table 13-1 Measurable pulse width range	197
Table 13-2 Filterable pulse width range	199
Table 13-3 PWDT register map and reset values	202
Table 14-1 TIMER register map	206
Table 15-1 ADC hardware trigger1 source	211
Table 15-2 CTU register map	211
Table 16-1 CRC register map	215
Table 17-1 GPIO multi-function	225
Table 17-2 GPIO register map	229

Table 18-1 I2C register map	239
Table 18-2 Interrupt summary	252
Table 19-1 Interrupt summary	262
Table 19-2 SPI register map	264
Table 20-1 DMA request mapping	270
Table 20-2 Programmable data width & data alignment	273
Table 20-3 DMA register map	273
Table 21-1 WDG register map	283
Table 22-1 RTC register map	290
Table 23-1 Flash memory organization	298
Table 23-2 The content of the key addresses in option page	299
Table 23-3 Read protection description	299
Table 23-4 Write protection description	300
Table 23-5 Watch dog enable description	300
Table 23-6 Embedded flash register map	308
Table 24-1 Clock configuration register	317
Table 24-2 Flashif related register map	349

1 Introduction

1.1 Overview

This chapter provides an overview of AC781x. It also presents high level descriptions of the modules. AC781x is a high-performance, low-power MCU with ARM Cortex™-M3 core.

- Up to 100 MHz CPU frequency
- Temperature range (ambient): -40°C to +125°C
- Voltage range: 2.7 V to 5.5 V

1.2 Module description

Table 1-1 AC781x module description

Module	Descriptions
ARM Cortex CM3 core	<ul style="list-style-type: none"> • 32-bit MCU core from ARM's Cortex-M class • 96 MHz CPU frequency
Memories	<ul style="list-style-type: none"> • Up to 256 KB flash memory • Up to 64 KB SRAM
Clocks	<ul style="list-style-type: none"> • External crystal oscillator or resonator <ul style="list-style-type: none"> • Range: 4 MHz to 30 MHz • External square wave input clock • Internal clock references <ul style="list-style-type: none"> • 8 MHz oscillator • 32 kHz oscillator
Analog	<ul style="list-style-type: none"> • One 12-bit ADC with up to 16 channels • Two analog comparators (ACMP) with internal 6-bit digital-to-analog converter (DAC)
Timers	<ul style="list-style-type: none"> • One 6-channel PWM with full function • Three 2-channel PWMs with basic function • 8 * periodic interrupt timer (Timer) • Real time clock (RTC) • One pulse width detection timer (PWDT) • System tick timer (SysTick)
Communications	<ul style="list-style-type: none"> • Two SPI modules • Two I2C modules • Six UART modules, only UART6 support LIN break • Two CAN controller • One LIN controller
Interfaces	<ul style="list-style-type: none"> • General Purpose Input/Output (GPIO) • Interrupt (IRQ)
Debug interfaces	<ul style="list-style-type: none"> • JTAG • SWD

2 Memory and Bus Architecture

2.1 System architecture

The main system consists of :

- Four masters
 - Cortex™-M3 core ICode bus (I-bus).
 - Cortex™-M3 core DCode bus (D-bus).
 - Cortex™-M3 core System bus (S-bus).
 - DMA (general-purpose DMA).
- Five slaves
 - Internal SRAM.
 - Internal Flash memory.
 - External serial Flash memory.
 - Fast IO, CRC and GPIO.
 - AHB to APB bridges (AHB2APB), which connect all the APB peripherals.

These are interconnected using a multilayer AHB bus architecture as shown in [Figure 2-1](#).

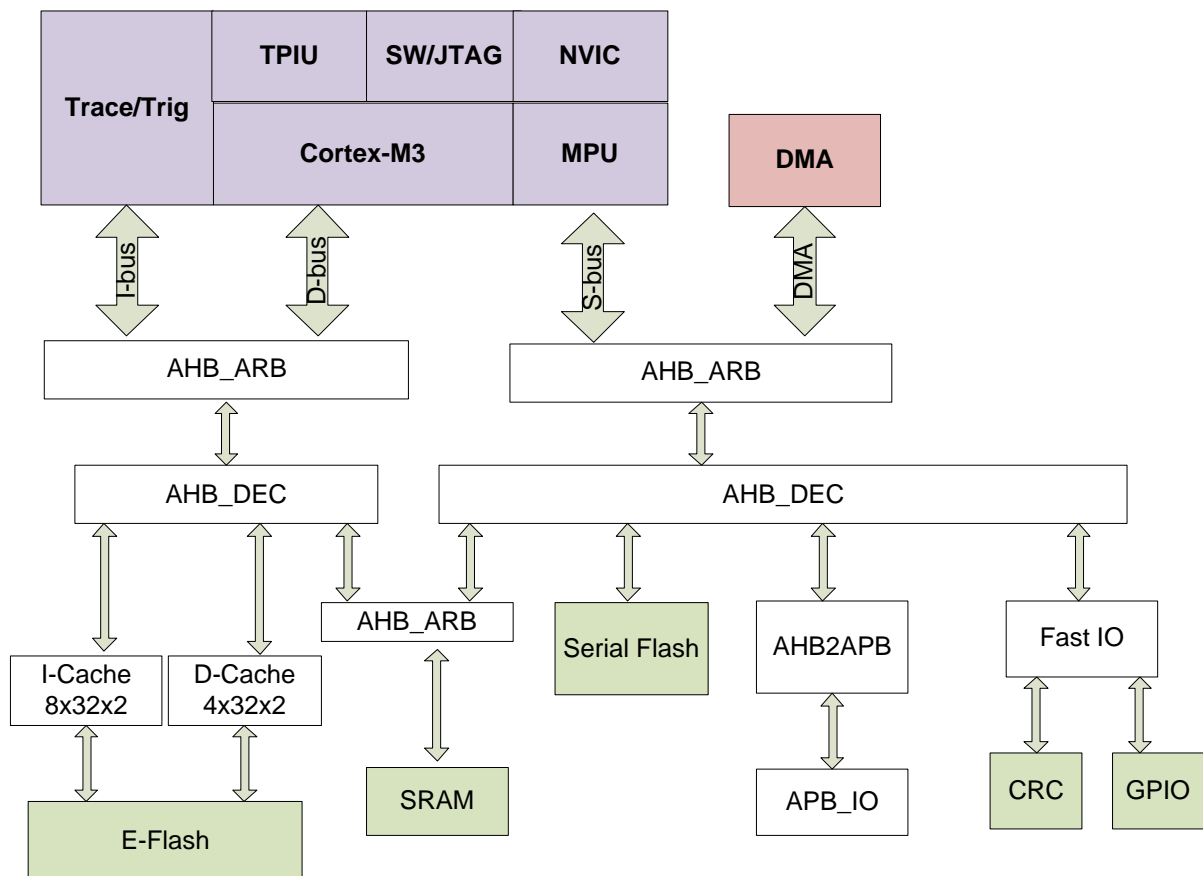


Figure 2-1 System architecture

ICode bus

This bus connects the Instruction bus of the Cortex™-M3 core to the Flash memory instruction interface. Prefetching is performed on this bus.

DCode bus

This bus connects the DCode bus (literal load and debug access) of the Cortex™-M3 core to the Flash Memory Data interface.

System bus

This bus connects the system bus of the Cortex™-M3 core (peripherals bus) to a BusMatrix which manages the arbitration between the core and the DMA.

MPU

AC781x supports MPU function, which is disabled by default. Please enable relevant configuration when using, please refer to the [Cortex™-M3 Technical Reference Manual](#).

DMA bus

This bus connects the AHB master interface of the DMA to the BusMatrix, which manages the access of CPU DCode and DMA to SRAM and peripherals.

BusMatrix

The BusMatrix manages the access arbitration between the core system bus and the DMA master bus. The arbitration uses a Round Robin algorithm. In connectivity line devices, the BusMatrix is composed of four masters (CPU ICode, CPU DCode, System bus and DMA) and five slaves (internal flash, serial flash, SRAM, fast IO and AHB2APB bridges).

Internal flash can only be accessed by ICode and DCode masters.

ICode and DCode can access SRAM and internal flash slaves, DCode master has higher priority than ICode.

AHB peripherals are connected on system bus through a BusMatrix to allow DMA access.

AHB2APB bridges (APB)

The AHB2APB bridge provides full synchronous connections between the AHB and the APB buses. APB1 is limited to ½ frequency of AHB frequency.

I-Cache

The I-Cache manages the access between Cortex™-M3 core ICode and internal flash memory. It includes direct access 8x32x2 SRAM to provide zero wait access time when cache matched. The cache structure is 8-ways, 2-set with LRU (Least Recently Used) replacement algorithm.

D-Cache

The D-Cache manages the access between Cortex™-M3 core DCode and internal flash memory. It includes direct access 4x32x2 SRAM to provide zero wait access time when cache matched. D-Cache has higher priority than I-Cache.

The cache structure is 2-ways, 2-set with LRU (Least Recently Used) replacement algorithm.

2.2 Memory organization

Program memory, data memory, registers and I/O ports are organized within the same linear 4-Gbyte address space.

The bytes are coded in memory in Little Endian format. The lowest numbered byte in a word is considered the word's least significant byte and the highest numbered byte the most significant.

For the detailed mapping of peripheral registers, please refer to the related chapters. The addressable memory space is divided into 8 main blocks, each of 512 MB.

2.2.1 Memory map

The following table is high-level device memory map, which includes four different memory map tables based on different boot up configuration, such as embedded flash memory boot up, ISP boot up, sram boot up and serial flash boot up.

All the memory areas that are not allocated to on-chip memories and peripherals are considered "Reserved".

Table 2-1 High-level device memory map

0xE010 0000	Reserved	0xE010 0000	Reserved	0xE010 0000	Reserved	0xE010 0000	Reserved
0xE00F FFFF	Cortex-M3's	0xE00F FFFF	Cortex-M3's	0xE00F FFFF	Cortex-M3's	0xE00F FFFF	Cortex-M3's
	internal		internal		internal		internal
0xE000 0000	peripherals	0xE000 0000	peripherals	0xE000 0000	peripherals	0xE000 0000	peripherals
0xDFFFFFFF	Reserved	0xDFFFFFFF	Reserved	0xDFFFFFFF	Reserved	0xDFFFFFFF	Reserved
0x6100 0000		0x6100 0000		0x6100 0000		0x6100 0000	
0x60FF FFFF	external	0x60FF FFFF	external	0x60FF FFFF	external	0x60FF FFFF	external
	memory		memory		memory		memory
0x6000 0000		0x6000 0000		0x6000 0000		0x6000 0000	
0x5FFF FFFF	Reserved	0x5FFF FFFF	Reserved	0x5FFF FFFF	Reserved	0x5FFF FFFF	Reserved
0x4400 0000		0x4400 0000		0x4400 0000		0x4400 0000	
0x43FF FFFF	Peripheral	0x43FF FFFF	Peripheral	0x43FF FFFF	Peripheral	0x43FF FFFF	Peripheral
	alias region		alias region		alias region		alias region
0x4200 0000		0x4200 0000		0x4200 0000		0x4200 0000	
0x41FF FFFF	Reserved	0x41FF FFFF	Reserved	0x41FF FFFF	Reserved	0x41FF FFFF	Reserved
0x4010 0000		0x4010 0000		0x4010 0000		0x4010 0000	
0x400F FFFF	Peripheral	0x400F FFFF	Peripheral	0x400F FFFF	Peripheral	0x400F FFFF	Peripheral
	APB		APB		APB		APB
0x4000 0000	Address	0x4000 0000	Address	0x4000 0000	Address	0x4000 0000	Address
0x3FFF FFFF	Reserved	0x3FFF FFFF	Reserved	0x3FFF FFFF	Reserved	0x3FFF FFFF	Reserved
0x2400 0000		0x2400 0000		0x2400 0000		0x2400 0000	
0x23FF FFFF	AHB SRAM	0x23FF FFFF	AHB SRAM	0x23FF FFFF	AHB SRAM	0x23FF FFFF	AHB SRAM
	alias region		alias region		alias region		alias region
0x2200 0000		0x2200 0000		0x2200 0000		0x2200 0000	
0x21FF FFFF	Reserved	0x21FF FFFF	Reserved	0x21FF FFFF	Reserved	0x21FF FFFF	Reserved

0x2008 2000		0x2008 2000		0x2008 2000		0x2008 2000	
0x2008 1FFF	AHB Fast IO	0x2008 1FFF	AHB Fast IO	0x2008 1FFF	AHB Fast IO	0x2008 1FFF	AHB Fast IO
0x2008 0000		0x2008 0000		0x2008 0000		0x2008 0000	
0x2007 FFFF	Reserved	0x2007 FFFF	Reserved	0x2007 FFFF	Reserved	0x2007 FFFF	Reserved
0x2001 0000		0x2001 0000		0x2001 0000		0x2001 0000	
0x2000 FFFF	AHB SRAM	0x2000 FFFF	AHB SRAM	0x2000 FFFF	AHB SRAM	0x2000 FFFF	AHB SRAM
0x2000 0000		0x2000 0000		0x2000 0000		0x2000 0000	
0x1FFF FFFF	Reserved	0x1FFF FFFF	Reserved	0x1FFF FFFF	Reserved	0x1FFF FFFF	Reserved
0x0804 3000		0x0804 3000		0x0804 3000		0x0804 3000	
0x0804 27FF	Reserved	0x0804 27FF	Reserved	0x0804 27FF	Reserved	0x0804 27FF	Reserved
0x0804 2000		0x0804 2000		0x0804 2000		0x0804 2000	
0x0804 1FFF	ISP Code	0x0804 1FFF	ISP Code	0x0804 1FFF	ISP Code	0x0804 1FFF	ISP Code
0x0804 0800		0x0804 0800		0x0804 0800		0x0804 0800	
0x0804 002F	option byte	0x0804 002F	option byte	0x0804 002F	option byte	0x0804 002F	option byte
0x0804 0000		0x0804 0000		0x0804 0000		0x0804 0000	
0x0803_FFFF	Flash memory	0x0803_FFFF	Flash memory	0x0803_FFFF	Flash memory	0x0803_FFFF	Flash memory
0x0800 0000		0x0800 0000		0x0800 0000		0x0800 0000	
0x07FF FFFF	Reserved	0x07FF FFFF	Reserved	0x07FF FFFF	Reserved	0x07FF FFFF	Reserved
0x0004_0000		0x0000 1800		0x0001 0000		0x0100 0000	
0x0003 FFFF	Flash memory	0x0000 17FF	system memory	0x0000 FFFF	AHB SRAM	0x00FF FFFF	external memory
0x0000 0000		0x0000 0000		0x0000 0000		0x0000 0000	
Flash memory boot up		ISP boot up		SRAM boot up		Serial flash boot up	

2.3 Detail function description

2.3.1 Internal SRAM

There is a 64 Kbytes of static SRAM. It can be accessed as bytes, half- words (16 bits) or full words (32 bits). The SRAM start address is 0x2000 0000.

2.3.2 Bit banding

The Cortex™-M3 memory map includes two bit-band regions. These regions map each word in an alias region of memory to a bit in a bit-band region of memory. Writing to a word in the alias region has the same effect as a read-modify-write operation on the targeted bit in the bit-band region.

In AC781x, both peripheral registers and SRAM are mapped in a bit-band region. This allows single bit-band write and read operations to be performed.

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

$$bit_word_addr = bit_band_base + (byte_offset \times 32) + (bit_number \times 4)$$

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.

Example:

The following example shows how to map bit 2 of the byte located at SRAM address 0x20000300 in the alias region:

$$0x22006008 = 0x22000000 + (0x300 \times 32) + (2 \times 4)$$

Writing to address 0x22006008 has the same effect as a read-modify-write operation on bit 2 of the byte at SRAM address 0x20000300.

Reading address 0x22006008 returns the value (0x01 or 0x00) of bit 2 of the byte at SRAM address 0x20000300 (0x01: bit set; 0x00: bit reset).

For more information on Bit-Banding, please refer to the [Cortex™-M3 Technical Reference Manual](#).

2.3.3 Fast IO memory map

The Fast IO bridge provides the channel to access CRC and GPIO through AHB bus with more efficiency, each has 4KB address range.

GPIO address range: 0x2008 0000 ~ 0x2008 0fff.

CRC address range: 0x2008 1000 ~ 0x2008 1fff.

2.3.4 Internal flash memory

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

- Density up to 256 Kbytes.
- Memory organization: The Flash memory is organized as a main block and an information block.
 - Main memory block of size: up to 64 Kb × 32 bits divided into 256 pages of 2 Kbytes each.
 - Information block of size: 512 × 32 bits.

The Flash memory controller features:

- Flash Program / Erase operation.
- Read / Write protection.
- Erase and blank check.
- Cache controller to improve read efficiency, the maximum efficiency is zero wait.

2.3.5 Read internal flash memory

Flash memory instructions and data access are performed through the AHB bus. The I-cache block is used for instruction fetches through the ICode bus. The D-cache block is used for data fetches through the DCode bus. Arbitration is performed to give the higher priority to DCode bus.

2.3.6 Chip Model

The model Type information can be accessed by users through the eflash read operation interface. The chip model information is stored in the continuous space (1*32 Bit) of 0x40002028 to 0x4000202B. Among them, the upper 8 bits are fixed 0xFF, and the lower 24 bits are chip model information.

2.3.7 Chip UUID

The UUID has a total of 128 Bit, which is generated by a random number and can be used as the unique identification mark of the chip. It can be accessed through the eflash read operation interface. The UUID information is stored in a continuous space (4*32Bit) from 0x4000202C to 0x40002038.

2.3.8 I-Cache and D-Cache

- Cache size is 8x32x2 and D-cache size is 4x32x2.
- The maximum of efficiency of I-cache controller is zero cycle wait of Cortex™ M3 running frequency.
- The maximum of efficiency of D-cache controller is zero cycle wait of Cortex™ M3 running frequency.
- DCode has the higher priority than ICode, D-cache will fetch data if D-cache missed and I-cache will stop and wait D-cache finished.

2.3.9 AHB to APB bridge

AHB to APB bridge translates AHB protocol to APB protocol and get low frequency interface. Most peripherals are APB interface. Below is the detail address assignment of each peripheral.

2.3.10 Nested Vectored Interrupt Controller (NVIC)

Features:

- 55 maskable interrupt channels (not including the 16 interrupt lines of Cortex™-M3).
- 128 programmable priority levels (7 bits of interrupt priority are used).
- Low-latency exception and interrupt handling.
- Power management control.
- Implementation of System Control Registers.

The NVIC and the processor core interface are closely coupled, which enables low latency interrupt processing and efficient processing of late arriving interrupts.

All interrupts including the core exceptions are managed by the NVIC. For more information on exceptions and NVIC programming see Chapter 5 Exceptions & Chapter 8 Nested Vectored Interrupt Controller of the [ARM Cortex™-M3 Technical Reference Manual](#).

Below is interrupt table:

Table 2-2 Interrupt table

Number	Type of priority	Acronym	Description
		Reserved	Reserved

	Number	Type of priority	Acronym	Description
	-3	fixed	Reset	System reset
	-2	fixed	NMI	Non-markable interrupt
	-1	fixed	Hard Fault	All class of fault
	0	settable	Mem Manage	Memory management
	1	settable	Bus Fault	AHB bus access fault
	2	settable	Usage Fault	Undefined instruction or illegal state
			Reserved	Reserved
			Reserved	Reserved
			Reserved	Reserved
			Reserved	Reserved
	3	settable	SVCall	System service call via SWI instruction
	4	settable	Debug Monitor	Debug monitor
			Reserved	Reserved
	5	settable	Pend SV	Pendable request for system service
	6	settable	Sys Tick	System tick timer
0	7	settable	PWDT	PWDT interrupt
1	8	settable	UART_1	UART1 interrupt
2	9	settable	UART_2	UART2 interrupt
3	10	settable	UART_3	UART3 interrupt
4	11	settable	UART_4	UART4 interrupt
5	12	settable	UART_5	UART5 interrupt
6	13	settable	UART_6	UART6 interrupt
7	14	settable	EXTI0	EXTI Line0 interrupt
8	15	settable	EXTI1	EXTI Line1 interrupt
9	16	settable	EXTI2	EXTI Line2 interrupt
10	17	settable	EXTI3	EXTI Line3 interrupt
11	18	settable	EXTI4	EXTI Line4 interrupt
12	19	settable	EXTI5	EXTI Line5 interrupt
13	20	settable	SPI_1	SPI0 interrupt
14	21	settable	SPI_2	SPI1 interrupt
15	22	settable	IIC_1	IIC0 interrupt
16	23	settable	IIC_2	IIC1 interrupt
17	24	settable	DMA_1	DMA channel 0 interrupt
18	25	settable	DMA_2	DMA channel 1 interrupt
19	26	settable	DMA_3	DMA channel 2 interrupt
20	27	settable	DMA_4	DMA channel 3 interrupt
21	28	settable	DMA_5	DMA channel 4 interrupt
22	29	settable	DMA_6	DMA channel 5 interrupt
23	30	settable	DMA_7	DMA channel 6 interrupt
24	31	settable	DMA_8	DMA channel 7 interrupt
25	32	settable	DMA_9	DMA channel 8 interrupt

	Number	Type of priority	Acronym	Description
26	33	settable	DMA_10	DMA channel 9 interrupt
27	34	settable	DMA_11	DMA channel 10 interrupt
28	35	settable	DMA_12	DMA channel 11 interrupt
29	36	settable	TIMER_0	Timer 0 interrupt
30	37	settable	TIMER_1	Timer 1 interrupt
31	38	settable	BKP	BKP tamper interrupt
32	39	settable	RTC	RTC interrupt
33	40	settable	WDT	Watch dog timer interrupt
34	41	settable	PWM_0	PWM 0 interrupt
35	42	settable	PWM_1	PWM 1 interrupt
36	43	settable	PWM_2	PWM 2 interrupt
37	44	settable	PVD	Power detect interrupt
38	45	settable	LIN	LIN interrupt
39	46	settable	EXTI6	EXTI Line6 interrupt
40	47	settable	SPM	System power manager interrupt
41	48		Reserved	
42	49	settable	CAN_1	CAN 1 interrupt
43	50	settable	CAN_2	CAN 2 interrupt
44	51	settable	ADC	ADC interrupt
45	52	settable	ACMP_0	ACMP 0 interrupt
46	53	settable	ACMP_1	ACMP 1 interrupt
47	54	settable	TIMER_2	TIMER 2 interrupt
48	55	settable	TIMER_3	TIMER 3 interrupt
49	56	settable	TIMER_4	TIMER 4 interrupt
50	57	settable	TIMER_5	TIMER 5 interrupt
51	58	settable	TIMER_6	TIMER 6 interrupt
52	59	settable	TIMER_7	TIMER 7 interrupt
53	60	settable	PWM_3	PWM 3 interrupt
54	61	settable	Embedded Flash	Embedded flash interrupt

2.3.11 Boot configuration

There are four boot up modes can be selected through below configuration table.

Table 2-3 Boot configuration

PIN Name	BOOT	UART1_CTS	UART1_RTS
eflash boot up	0	x	x
ISP boot	1	0	0
sram boot	1	1	0
serial flash boot	1	0	1

Note:

1. x means do not care.

AutoChips Confidential

© 2013 - 2021 AutoChips Inc.

Page 29 of 360

This document contains information that is proprietary to AutoChips Inc.

Unauthorized reproduction or disclosure of this information in whole or in part is strictly prohibited.

2. The control of the boot configuration is bound to the pins and must be PC7(UART1_RTS) and PD0(UART1_CTS).
3. If the application only needs EFLASH BOOT, UART1_RTS and UART1_CTS can be used for other function after controlling BOOT=0, which can ignore the level state.
4. ISP download pins are UART2's PD1 (TX) and PD2 (Rx).

The values on the boot up configuration pins are latched on the 8th rising edge of 8MHz clock after a Reset. It is up to the user to set these pins to select the required boot mode, user should keep these pins stable before latched.

Due to its fixed memory map, the code area starts from address 0x0000 0000 (accessed through the ICode/DCode buses). The Cortex-M3 CPU always fetches the reset vector on the ICode bus, which implies to have the boot space available only in the code area (typically, Flash memory). This microcontrollers implement a special mechanism to be able to boot also from SRAM and not only from main Flash memory and ISP Code memory.

Depending on the selected boot mode, main Flash memory, ISP, SRAM and serial flash is accessible as follows:

- Boot from main Flash memory: the main Flash memory is aliased in the boot memory space (0x0000 0000), but still accessible from its original memory space (0x800 0000). In other words, the Flash memory contents can be accessed starting from address 0x0000 0000 or 0x800 0000.
- Boot from ISP: the ISP is aliased in the boot memory space (0x0000 0000), but still accessible from its original memory space (0x08040800) .
- Boot from the embedded SRAM: SRAM is aliased in the boot memory space (0x0000 0000), but still accessible from its original memory space 0x2000 0000.
- Boot from the serial flash: serial flash address is aliased in the boot memory space (0x0000 0000), but still accessible from its original memory space 0x6000 0000.

2.4 Address Table

Table 2-4 Address assignment of each peripheral

APB Memory Map	Base_Address	Size
CKGEN	0x40000000	4K
GPIO	0x40001000 / 0x20080000	4K
Embedded flash	0x40002000	4K
ADC	0x40003000	4K
ACMP	0x40005000	4K
LIN_0	0x40007000	2K
CAN_1	0x40007800	1K
CAN_2	0x40007C00	1K
SPM	0x40008000	1K
RTC	0x40008400	1K
WDT	0x4000B000	4K
SPI_0	0x4000C000	4K
SPI_1	0x4000D000	4K
IIC_0	0x4000E000	4K

APB Memory Map	Base_Address	Size
IIC_1	0x4000F000	4K
Cortex-M3 controller	0x40010000	2K
Serial flash controller	0x40010800	2K
TIMER	0x40011000	4K
DMA	0x40012000	4K
PWM_0	0x40013000	4K
PWM_1	0x40014000	4K
PWM_2	0x40015000	4K
CTU	0x40016000	4K
PWDT	0x40017000	4K
UART_1	0x40018000	4K
UART_2	0x40019000	4K
UART_3	0x4001A000	4K
UART_4	0x4001B000	4K
UART_5	0x4001C000	4K
UART_LIN_6	0x4001D000	4K
PWM_3	0x4001E000	4K
CRC	0x20081000	4K

3 Reset

3.1 Introduction

There are 8 reset sources to generate IC reset.

POR reset:

- POR_rst: IC power up reset.

System reset:

- ext_rstin : external reset from IC pad, active low.
- lvd_rst: low power detection reset, active low.
- sysreset_req: Cortex-M3 software reset.
- lock_up: Cortex-M3 lock up reset.
- wdt_rst: watch dog timer reset, active in normal mode.
- wdt_rst_32k: watch dog timer reset, active in stop mode.
- stop_ack_error: stop mode acknowledge error reset. In stop mode, if wake up source is busy during 16'h0080 32K clock cycles, then generate acknowledge error to reset whole chip.

Each of the system reset sources has an associated bit in the status register 0x40000010.

3.2 Block diagram

The simple block diagram is listed in Figure 3-1. Note: Low-level reset signal trigger the system reset.

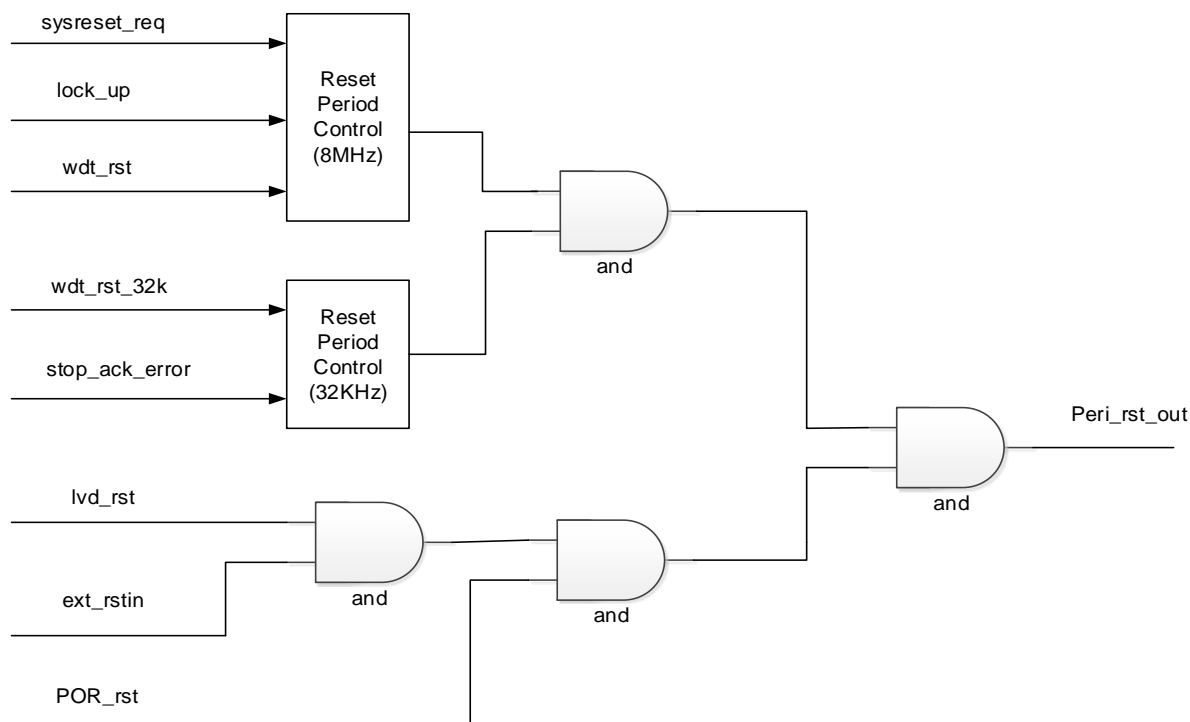


Figure 3-1 Reset block diagram

3.3 Reset detail function description

3.3.1 Power On Reset (POR)

When power is initially applied to the MCU or when the supply voltage drops below the power-on reset voltage level (VPOR), the POR circuit causes a POR reset condition. As the supply voltage rises, the LVD circuit holds the MCU in reset until the supply has risen above the LVD low threshold (VLVDL), please refer to the detail POR/LVD control document.

3.3.2 System reset source

Resetting the MCU provides a way to start processing from a known set of initial conditions. System reset begins with the on-chip regulator in full regulation and system clocking generation from an internal reference. When the processor exits reset, it performs the following:

- Read the start SP (SP_main) from vector-table offset 0.
- Read the start program counter (PC) from vector-table offset 4.
- The Link Register (LR) is set to 0xFFFF_FFFF.

3.3.2.1 External pin reset (ext_rstin)

There is a dedicated pin in the MCU, it is used to reset the whole MCU function and restart. As it is low active, suggest to add pull up function in external PCB to reject noise.

3.3.2.2 Low voltage detection (lvd_rst)

This device includes a system to protect against low-voltage conditions in order to protect memory contents and control MCU system states during supply voltage variations. This system consists of a power-on reset (POR) circuit, and a LVD circuit with a user selectable trip voltage, either high (VLVDH) or low (VLVDL).

Please refer to the detail POR/LVD control document.

3.3.2.3 Watch dog timer (wdt_rst/wdt_rst_32k)

The watchdog timer (WDOG) monitors the operation of the system by expecting periodic communication from the software. This communication is generally known as servicing (or refreshing) the watchdog. If this periodic refreshing does not occur, the watchdog issues a system reset. Please refer to watch dog timer section for detail.

3.3.2.4 XOSC monitor

XOSC monitor system can be activated by CKGEN_SRC_SEL[16]. In this case, the clock detector is enabled after the HSE oscillator startup delay, and disabled when this oscillator is stopped. If a failure is detected on the HSE oscillator clock, this oscillator is automatically disabled, [XOSC loss status](#) flag is active and NMI interrupt is generated to inform the software about the failure allowing the MCU to perform rescue operations.

3.3.2.5 LOCKUP reset (lock_up)

The lock_up gives immediate indication of seriously errant kernel software. This is the result of the core being locked because of an unrecoverable exception following the activation of the processor's built in system state protection hardware.

3.4 Register mapping

Table 3-1 Reset register map

Address	Name	Width	Register Function
4000000C	RESET_CTL1	32	chip reset control
40000010	RESET_STATUS	32	chip reset status

4000000C **RESET_CTL1** chip reset control 00000800

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name							cpu_lockup_rst_en	cpu_sysrst_en					reset_pulse_32K			
Type							RW	RW					RW			
Reset							0	1					0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	reset_pulse_8M															
Type	RW															
Reset	0	0	0	0	1	0	0	0								

Bit(s)	Name	Description
25	cpu_lockup_rst_en	CPU lock up reset enable 1'b1 : can generate reset for IC 1'b0 : can not generate reset for IC
24	cpu_sysrst_en	CPU system reset enable 1'b1 : can generate reset for IC 1'b0 : can not generate reset for IC
19:16	reset_pulse_32K	Programmable reset pulse with 32KHz clock, include watch dog timer 32KHz reset and stop mode ack error reset.
15:8	reset_pulse_8M	Programmable reset pulse with 8MHz clock, include cpu system reset, cpu lock up reset and watch dog timer reset.

40000010 **RESET_STATUS** chip reset status 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																clear_reset_status
Type																RW
Reset																0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name							xosc_loss_rst	stop_ack_err_rst_status	CPU_lockup_rst_status	CPU_sysrst_status	wdt_32k_res_status	wdt_reset_status	nowdt_reset_status	ext_reset_status	LVD_reset_status	POR_reset_status
Type							RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset							0	0	0	0	0	0	0	0	0	0

Bit(s)	Name	Description
16	clear_reset_status	clear reset status 1'b1 : clear all reset status 1'b0 : allow rereset status to update
9	xosc_loss_status	xosc loss status, high active
8	stop_ack_error_rst_status	stop mode acknowledge error reset status, high active
7	CPU_lockup_rst_status	CPU lock up reset status, high active
6	CPU_sysreset_status	CPU system reset status, high active
5	wdt_32k_reset_status	watch dog 32k type reset status, high active
4	wdt_reset_status	watch dog normal reset status, high active
3	nowdt_reset_status	Chip power on reset status, high active
2	ext_reset_status	external pin reset status, high active
1	LVD_reset_status	LVD reset status, high active
0	POR_reset_status	POR reset status, high active

4 Clock

4.1 Introduction

The clock control function provides clock source choices for the MCU. The module contains a phase-locked loop (PLL) as a clock source that is controllable by either an internal or an external reference clock. The module can provide this PLL clock or either of the internal or external reference clocks as a source for the MCU system clock.

There are configuration control signals to generate all modules' clock source and frequency.

4.2 Clock control diagram

This device contains the following on-chip clock sources:

- High speed internal RC(HSI): the internal RC OSC to provide 8MHz clock source.
- High speed external RC(HSE): the external OSC to provide 4MHZ ~30MHz crystal and oscillator.
- Low speed internal RC(LSI): the internal low speed RC OSC to provide 32kHz clock source.
- Phase-lock loop (SYSPLL): the PLL provides high speed clock up to 96MHz.

Each peripheral has dedicated clock enable signal to control clock on/off, please refer to register control chapter to know the detail address.

Note:

- System clock up to 100MHz (typical usage value is 96MHz)
- HCLK (AHB) up to 100MHz
- PCLK (APB) up to 50MHz. when HCLK is 100MHz, APBCLK_DIV minimum value is 2

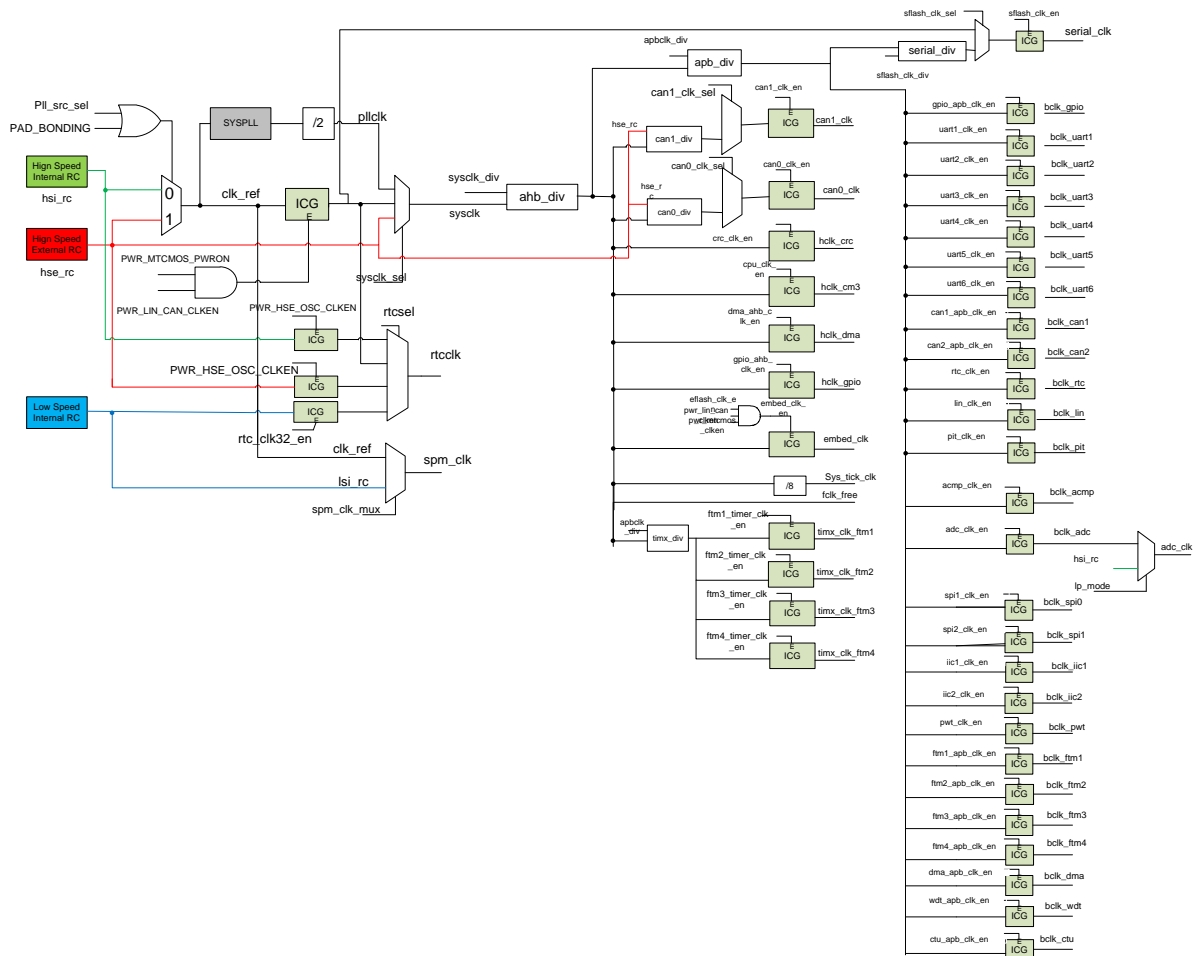


Figure 4-1 Clock control diagram

4.3 System Clock

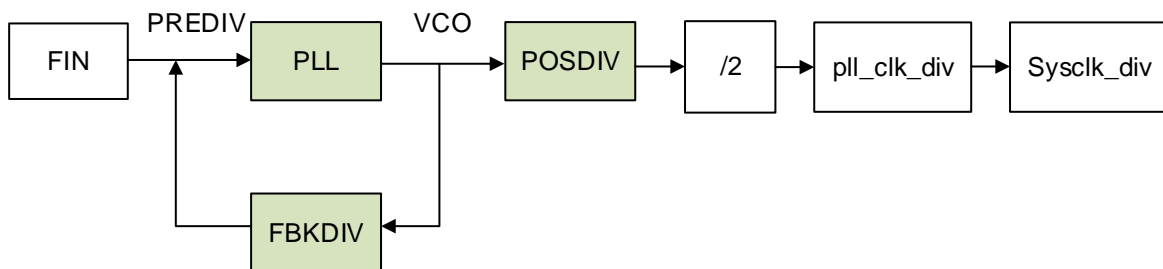


Figure 4-2 System clock diagram

FIN: Frequency Input, support 4MHz to 30MHz

$VCO = FIN * FBKDIV / PREDIV$

System Clock = $VCO / POSDIV / 2 / pll_clk_div / sysclk_div$

Note:

AutoChips Confidential

© 2013 - 2021 AutoChips Inc.

Page 37 of 360

This document contains information that is proprietary to AutoChips Inc.
Unauthorized reproduction or disclosure of this information in whole or in part is strictly prohibited.

1. The frequency of VCO divided by POSDIV cannot be greater than 400MHz, and the frequency range of VCO is 0.5GHz to 1.5GHz.
2. The FBKDIV recommendation is 72, 80, 96, 128, 144, 160, 192
3. The external crystal can be 4MHz to 30MHz, and the recommendation input frequency of PLL is less than 8MHz.

4.4 Register mapping

Table 4-1 Clock register map

Address	Name	Width	Register Function
40000000	CKGEN_SRC_SEL	32	clock source selection
40000004	PERI_CLK_EN_0	32	peripheral clock enable control 0
40000008	PERI_CLK_EN_1	32	peripheral clock enable control 1
40000018	PERI_SFT_RST1	32	peripheral software reset control 1
4000001C	PERI_SFT_RST2	32	peripheral software reset control 2
40000020	PLL_CLK_DIV	32	PLL clock divider
40008890	REG_MCU_SYSPLL1_CFG0	32	SYSPLL1 configuration register 0

40000000 <u>CKGEN_SRC_SEL</u> clock source selection 00000400																
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	can2_clk_div		can1_clk_div		can2_clk_sel	can1_clk_sel		sflash_clk_sel				PII_ref_sel				xosc_mon_enable
Type	RW		RW		RW	RW		RW				RW				RW
Reset	0	0	0	0	0	0		0				0				0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name						apbclk_div			sysclk_div			sflash_clk_div		sysclk_sel		
Type						RW			RW			RW		RW		
Reset						1	0	0	0	0	0	0	0	0	0	0

Bit(s)	Name	Description
31:30	can2_clk_div	CAN2 clock divider by AHB clock (can2_clk <= 50MHz) 2'h0 : divider by 1 2'h1 : divider by 2 2'h2 : divider by 4 2'h3 : divider by 8
29:28	can1_clk_div	CAN1 clock divider by AHB clock (can1_clk <= 50MHz) 2'h0 : divider by 1 2'h1 : divider by 2 2'h2 : divider by 4 2'h3 : divider by 8
27	can2_clk_sel	CAN2 clock source select 1'h0 : external oscillator clock 1'h1 : apb divided clock
26	can1_clk_sel	CAN1 clock source select 1'h0 : external oscillator clock 1'h1 : apb divided clock
24	sflash_clk_sel	serial flash clock source select 1'h0 : pll reference clock 1'h1 : apb divided clock
20	PII_ref_sel	pll reference clock select

Bit(s)	Name	Description
		1 : reference clock is external oscillator 0 : reference clock is internal oscillator
16	xosc_mon_enable	XOSC monitor enable
10:8	apbclk_div	APB clock divider by system clock 3'b0xx : divider by 1 3'b100 : divider by 2 3'b101 : divider by 4 3'b110 : divider by 8 3'b111 : divider by 16
7:4	sysclk_div	system clock divider 4'b0xxx : divider by 1 4'b1000 : divider by 2 4'b1001 : divider by 4 4'b1010 : divider by 8 4'b1011 : divider by 16 4'b1100 : divider by 64 4'b1101 : divider by 128 4'b1110 : divider by 256 4'b1111 : divider by 512
3:2	sflash_clk_div	serial flash clock divider by APB clock 2'h0 : divider by 1 2'h1 : divider by 2 2'h2 : divider by 4 2'h3 : divider by 8
1:0	sysclk_sel	system clock source select 2'h0 : internal oscillator 2'h1 : PLL output 2'h2 : external oscillator 2'h3 : 1'b0

40000004 PERI_CLK_EN_0 peripheral clock enable control 0 03400001

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	CLK_CAN2_CORE	CLK_CAN1_CORE	CLK_CAN2	CLK_CAN1	CLK_LIN	CLK_CR_C	CLK_WD_T_A_PB	CLK_GPI_O_A_HB	CLK_GPI_O_A_PB	CLK_DM_A_A_HB	CLK_DM_A_A_PB	CLK_RT_C	CLK_TIM	CLK_PW_M3_TIMER	CLK_PW_M2_TIMER	CLK_PW_M1_TIMER
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CLK_PWM0_TIMER	CLK_PWM3_APB	CLK_PWM2_A_PB	CLK_PWM1_A_PB	CLK_PWM0_APB	CLK_PW_DT	CLK_I2C_2	CLK_I2C_1	CLK_SPI_2	CLK_SPI_1	CLK_UA_RT6	CLK_UA_RT5	CLK_UA_RT4	CLK_UA_RT3	CLK_UA_RT2	CLK_UA_RT1
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Bit(s)	Name	Description
31	CLK_CAN2_CORE	CAN2 core clock enable 1'b1 : clock enable 1'b0 : clock disable
30	CLK_CAN1_CORE	CAN1 core clock enable 1'b1 : clock enable 1'b0 : clock disable

Bit(s)	Name	Description
29	CLK_CAN2	CAN2 clock enable 1'b1 : clock enable 1'b0 : clock disable
28	CLK_CAN1	CAN1 clock enable 1'b1 : clock enable 1'b0 : clock disable
27	CLK_LIN	LIN clock enable 1'b1 : clock enable 1'b0 : clock disable
26	CLK_CRC	CRC clock enable 1'b1 : clock enable 1'b0 : clock disable
25	CLK_WDT_APB	WDG APB clock enable 1'b1 : clock enable 1'b0 : clock disable
24	CLK_GPIO_AHB	GPIO AHB clock enable 1'b1 : clock enable 1'b0 : clock disable
23	CLK_GPIO_APB	GPIO APB clock enable 1'b1 : clock enable 1'b0 : clock disable
22	CLK_DMA_AHB	DMA AHB clock enable 1'b1 : clock enable 1'b0 : clock disable
21	CLK_DMA_APB	DMA APB clock enable 1'b1 : clock enable 1'b0 : clock disable
20	CLK_RTC	RTC clock enable 1'b1 : clock enable 1'b0 : clock disable
19	CLK_TIM	TIMER clock enable 1'b1 : clock enable 1'b0 : clock disable
18	CLK_PWM3_TIMER	PWM3 timer clock enable 1'b1 : clock enable 1'b0 : clock disable
17	CLK_PWM2_TIMER	PWM2 timer clock enable 1'b1 : clock enable 1'b0 : clock disable
16	CLK_PWM1_TIMER	PWM1 timer clock enable 1'b1 : clock enable 1'b0 : clock disable
15	CLK_PWM0_TIMER	PWM0 timer clock enable 1'b1 : clock enable 1'b0 : clock disable
14	CLK_PWM3_APB	PWM3 APB clock enable 1'b1 : clock enable 1'b0 : clock disable
13	CLK_PWM2_APB	PWM2 APB clock enable 1'b1 : clock enable 1'b0 : clock disable
12	CLK_PWM1_APB	PWM1 APB clock enable 1'b1 : clock enable

Bit(s)	Name	Description
		1'b0 : clock disable
11	CLK_PWM0_APB	PWM0 APB clock enable 1'b1 : clock enable 1'b0 : clock disable
10	CLK_PWDT	PWDT clock enable 1'b1 : clock enable 1'b0 : clock disable
9	CLK_I2C2	IIC2 clock enable 1'b1 : clock enable 1'b0 : clock disable
8	CLK_I2C1	IIC1 clock enable 1'b1 : clock enable 1'b0 : clock disable
7	CLK_SPI2	SPI2 clock enable 1'b1 : clock enable 1'b0 : clock disable
6	CLK_SPI1	SPI1 clock enable 1'b1 : clock enable 1'b0 : clock disable
5	CLK_UART6	UART6 clock enable 1'b1 : clock enable 1'b0 : clock disable
4	CLK_UART5	UART5 clock enable 1'b1 : clock enable 1'b0 : clock disable
3	CLK_UART4	UART4 clock enable 1'b1 : clock enable 1'b0 : clock disable
2	CLK_UART3	UART3 clock enable 1'b1 : clock enable 1'b0 : clock disable
1	CLK_UART2	UART2 clock enable 1'b1 : clock enable 1'b0 : clock disable
0	CLK_UART1	UART1 clock enable 1'b1 : clock enable 1'b0 : clock disable

40000008 PERI_CLK_EN_1 peripheral clock enable control 1 00000001

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name													CLK _AC _MP	CLK _AD _C	CLK _CT _U _A _PB	CLK _SF _LAS _H
Type													RW	RW	RW	RW
Reset													0	0	0	1

Bit(s)	Name	Description
3	CLK_ACMP	ACMP clock enable

Bit(s)	Name	Description
		1'b1 : clock enable 1'b0 : clock disable
2	CLK_ADC	ADC core clock enable 1'b1 : clock enable 1'b0 : clock disable
1	CLK_CTU_APB	CTU APB clock enable 1'b1 : clock enable 1'b0 : clock disable
0	CLK_SFLASH	serial flash controller clock enable 1'b1 : clock enable 1'b0 : clock disable

40000018 PERI_SFT_RST1 peripheral software reset control 1 03900001

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	res erv ed	reser ved	SRS T_C AN2	SRS T_C AN1	SR ST_ LIN	SR ST_ CR C	SR ST_ WD G	SR ST_ GPI O_ AH B	SR ST_ GPI O_ AP B	SR ST_ DM A_ AH B	SR ST_ DM A_ AP B	SR ST_ RT C	SR ST_ TIM	SR ST_ PW M3 _TI ME R	SR ST_ PW M2 _TI ME R	SR ST_ PW M1 _TI ME R
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	1	1	1	0	0	1	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	SR ST_ P WM O_ T IME R	SRS T_P WM3 _AP B	SRS T_P WM2 _AP B	SRS T_P WM1 _AP B	SR ST_ PW M0 _A PB	SR ST_ PW DT	SR ST_ I2C 2	SR ST_ I2C 1	SR ST_ SPI 2	SR ST_ SPI 1	SR ST_ UA RT 6	SR ST_ UA RT 5	SR ST_ UA RT 4	SR ST_ UA RT 3	SR ST_ UA RT 2	SR ST_ UA RT 1
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Bit(s)	Name	Description
29	SRST_CAN2	CAN2 software reset 0 : reset active 1 : reset inactive
28	SRST_CAN1	CAN1 software reset 0 : reset active 1 : reset inactive
27	SRST_LIN	LIN software reset 0 : reset active 1 : reset inactive
26	SRST_CRC	CRC software reset 0 : reset active 1 : reset inactive
25	SRST_WDG	Watch dog timer software reset 0 : reset active 1 : reset inactive
24	SRST_GPIO_AHB	GPIO AHB software reset 0 : reset active 1 : reset inactive
23	SRST_GPIO_APB	GPIO APB software reset 0 : reset active

Bit(s)	Name	Description
		1 : reset inactive
22	SRST_DMA_AHB	DMA AHB software reset 0 : reset active 1 : reset inactive
21	SRST_DMA_APB	DMA APB software reset 0 : reset active 1 : reset inactive
20	SRST_RTC	RTC software reset 0 : reset active 1 : reset inactive
19	SRST_TIM	TIMER software reset 0 : reset active 1 : reset inactive
18	SRST_PWM3_TIMER	PWM3 timer software reset 0 : reset active 1 : reset inactive
17	SRST_PWM2_TIMER	PWM2 timer software reset 0 : reset active 1 : reset inactive
16	SRST_PWM1_TIMER	PWM1 timer software reset 0 : reset active 1 : reset inactive
15	SRST_PWM0_TIMER	PWM0 timer software reset 0 : reset active 1 : reset inactive
14	SRST_PWM3_APB	PWM3 APB software reset 0 : reset active 1 : reset inactive
13	SRST_PWM2_APB	PWM2 APB software reset 0 : reset active 1 : reset inactive
12	SRST_PWM1_APB	PWM1 APB software reset 0 : reset active 1 : reset inactive
11	SRST_PWM0_APB	PWM0 APB software reset 0 : reset active 1 : reset inactive
10	SRST_PWDT	PWDT software reset 0 : reset active 1 : reset inactive
9	SRST_I2C2	IIC2 software reset 0 : reset active 1 : reset inactive
8	SRST_I2C1	IIC1 software reset 0 : reset active 1 : reset inactive
7	SRST_SPI2	SPI2 software reset 0 : reset active 1 : reset inactive
6	SRST_SPI1	SPI1 software reset 0 : reset active 1 : reset inactive
5	SRST_UART6	UART6 software reset

Bit(s)	Name	Description
		0 : reset active 1 : reset inactive
4	SRST_UART5	UART5 software reset 0 : reset active 1 : reset inactive
3	SRST_UART4	UART4 software reset 0 : reset active 1 : reset inactive
2	SRST_UART3	UART3 software reset 0 : reset active 1 : reset inactive
1	SRST_UART2	UART2 software reset 0 : reset active 1 : reset inactive
0	SRST_UART1	UART1 software reset 0 : reset active 1 : reset inactive

4000001C PERI_SFT_RST2 peripheral software reset control 2 00000011

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name												SRST_ANA_REG	SRST_ACMP	SRST_ADC	SRST_CTU	SRST_SFLASH
Type												RW	RW	RW	RW	RW
Reset												1	0	0	0	1

Bit(s)	Name	Description
4	SRST_ANA_REG	ANA reg soft reset
3	SRST_ACMP	ACMP software reset 0 : reset active 1 : reset inactive
2	SRST_ADC	ADC software reset 0 : reset active 1 : reset inactive
1	SRST_CTU	CTU software reset 0 : reset active 1 : reset inactive
0	SRST_SFLASH	Sflash software reset 0 : reset active 1 : reset inactive

40000020 PLL CLK DIV pll clk divider 00000000

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

Bit(s)	Name	Description
3:0	pll_clk_div	divider = pll_clk_div + 1

40008890 REG MCU SYSPLL1 CFG0 SYSPLL1 configuration register 0 003240C0

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	RG_MCU_SYSPLL1_PREDIV		RG_MCU_SYSPLL1_POSDIV					RG_MCU_SYSPLL1_FBKDIV								
Type	RW		RW					RW								
Reset	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Reserved					Reserved										
Type																
Reset	0	1	0	0	0	0	0	0	1	1	0					

Bit(s)	Name	Description
31:30	RG_MCU_SYSPLL1_PREDIV	Pre-divider ratio 2'b00: Fref = Fin/1 2'b01: Fref = Fin/2 2'b1X: Fref = Fin/4
27	RG_MCU_SYSPLL1_FBKSEL	Feedback clock select (set1'b1, when VCO>750MHz) 1'b0: Fvco/1 1'b1: Fvco/2
29:28	RG_MCU_SYSPLL1_POSDIV	Post-divider ratio for single-end clock output 2'b00: VCO/1 2'b01: VCO/2 2'b1X: VCO/4
24:17	RG_MCU_SYSPLL1_FBKDIV	Feedback divide ratio 8'd6: /6 8'd255: /255

5 Power Modes

5.1 Introduction

This chapter describes the chip power modes and functionality of each module in these modes.

5.2 Power modes

The device supports Run, Sleep, Stop, and Standby modes which are easy to use for customers both from different power consumption level and functional requirement. I/O states are held in Run, Sleep, Stop modes.

- Run mode – CPU clocks can be run at full speed.
- Sleep mode – CPU into sleep mode, system clocks and bus clock are running.
- Stop mode – CPU into deep sleep mode, some modules can wake up CPU.
- Standby mode – CPU and modules are shut down, RTC, WKUP pin can wake up CPU.

5.3 Entering and exiting power modes

1. Enable the wakeup source required with the `REG_EN_PERIPH_WKUP`.
2. Set the power mode with the `REG_SLEEP_MODE[1:0]` before WFI, the configuration as follows:
 - 1) 2'b00: stop mode lite
 - 2) 2'b01: stop mode
 - 3) 2'b1x: standby mode
3. Call WFI to enter the power mode.

The processor exits the low-power mode via an interrupt.

5.4 Module operation in low-power modes

The following table illustrates the functionality of each module while the chip is in each of the low-power modes. The standard behavior is shown with some exceptions.

Table 5-1 Module functionality in low-power modes

Module	Sleep	Stop	Stop lite	Standby
CM3	Standby	On	On	Off
SRAM	ON	Sleep	Sleep	Off
eflash	ON	Off	Off	Off
I2C	ON	On	On ³	Off
SPI	ON	On	On ²	Off
GPIO	ON	On	On ⁴	Off
WDT	ON	On	On	Off
PWDT	ON	Off	On	Off
UART	ON	Off	On	Off
DMA	ON	Off	On	Off

Module	Sleep	Stop	Stop lite	Standby
TIM	ON	Off	On	Off
PWM	ON	Off	On	Off
CRC	ON	Off	On	Off
CTU	ON	Off	On	Off
CAN	ON	On	On ¹	Off
LIN	ON	On	On ¹	Off
RTC	ON	On	On	On
SPM	ON	On	On	On
PLL	ON	Off	Off	Off
XOSC	ON	Off	Off	Off
LFOSC	ON	Off	Off	Off
LPOSC	ON	On	On	On
LVD	Selectable inactive or active	Selectable inactive or active	Selectable inactive or active	Selectable inactive or active

Note:

1. Supports wake-up on edge in Stop mode.
2. Supports slave mode receive and wake-up in Stop mode.
3. Supports address match wake-up in Stop mode.
4. Supports pin interrupt wake-up in Stop mode.

6 System Power Management

6.1 Introduction

SPM (System Power Management) provides the max flexibility for software developers to develop the system housekeeping task. It aims for low level sleep/wake-up task such as MTCMOS power domain, SRAM and analog module power control.

6.2 Feature list

- Stop/standby mode whole chip power management.

6.3 Block diagram

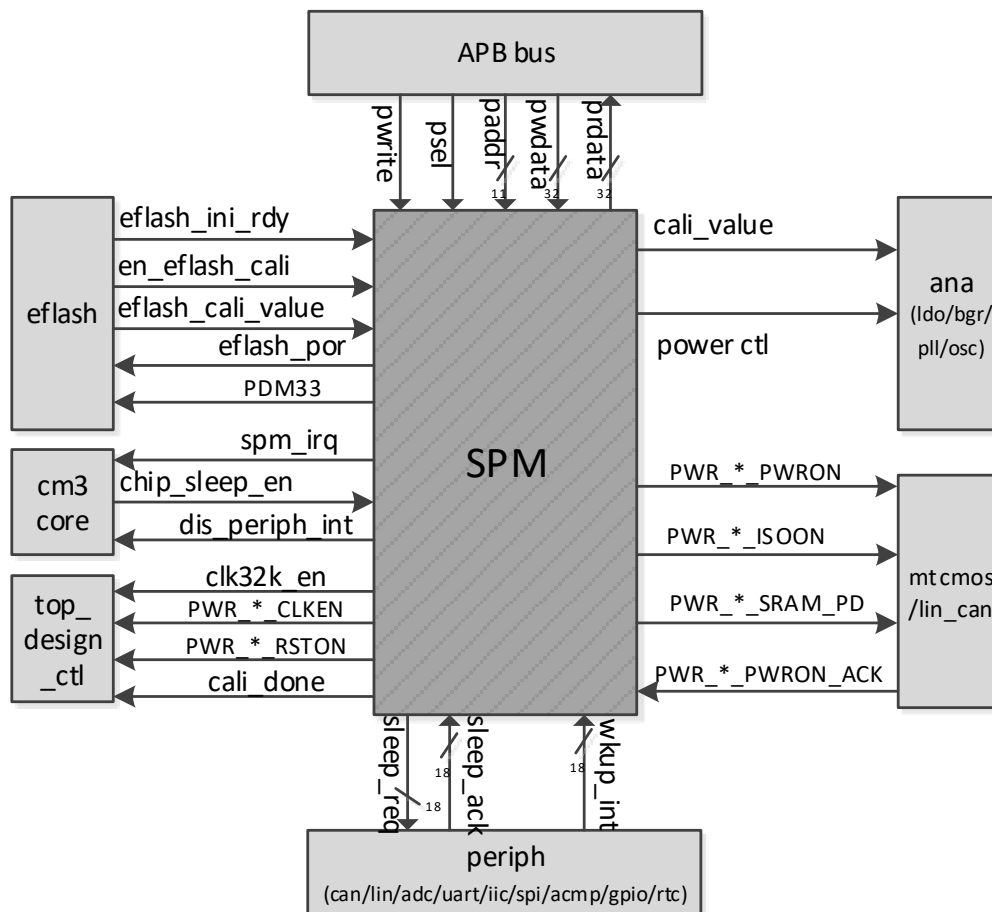


Figure 6-1 SPM block diagram

6.4 Application notes

6.4.1 SPM power control program guide

Stop mode and standby mode are supported in AC781x. For stop mode, PD_MTCMOS power off, but in standby mode, all digital modules are external power off except module SPM.

The WFI instruction invokes stop and standby modes for the chip, and processor exits the low-power mode via an interrupt.

Program sequence:

1. Program power timer counter initial value if need.
2. Program SPM config registers to determined power mode, wakeup source supported.
3. WFI.

Power related config bits:

- **force sleep mode:** when cm3 triggers a WFI instruction, SPM will send a sleep request to all peripherals to check the working state. If all peripherals idle and a sleep ack is replayed in a certain clk cycle (programmable), SPM will continue the sleep sequence. If not, SPM will stop the sleep sequence and send over counter flag to cm3. But when enable force sleep mode, chip will enter sleep state at the end of sleep ack waiting count, even not all peripherals reply sleep ack.
- **fast boot mode:** when enable, chip will wake up immediately when receive a wakeup interrupt during sleeping sequence.
- **en_adc_wakeup:** when enable, ADC wakeup supported, Ifosc (8M)/BGR will be always on. But SPM will work in clk 32k frequency.
- **en_periph_ack[17:0]:** when disable, SPM will not check corresponding peripheral sleep ack reply before enter sleep sequence.
- **en_periph_wakeup_int[19:0]:** when disable, corresponding peripheral wakeup is not supported, and this int will be ignored.
- **en_periph_irq_trigger:** when enables, SPM triggers spm_irq to wakeup cm3. For peripheral (e.g. rtc/adc/gpio/..) trigger wakeup irq themselves, just set corresponding bit 0.
- **en_rtc_32k_aon:** 1'b1 enable rtc 32k clk always on. When disable, rtc 32k gated when chip enter sleep state.

6.4.2 XOSC/SYSPLL power control

XOSC/SYSPLL is off at default. When XOSC/SYSPLL is needed, XOSC/SYSPLL can be power on by configuring REG_PWR_MGR_CFG1.

SPM register REG_PWR_MGR_CFG1:

- **REG_XOSC_ON:** External high-speed clock enables.
- **REG_XOSC_HSEBYP:** External high-speed clock bypass.
- **REG_SYSPLL_ON:** SYSPLL enable.

When corresponding bit is set to 1'b1, SPM will power XOSC/PLL on following the power on sequence and it may take a few times. So you should be waiting for XOSC/SYSPLL power on finish and clock ready before use it. XOSC/PLL power on states can be determined by reading SPM register REG_PWR_MGR_CFG1.

- **XOSC_RDY:** External high-speed clock ready flag.
- **SYSPLL_RDY:** PLL clock ready flag.

For example, before clk source switches to PLL clk, you should power SYSPLL on first, and wait for SYSPLL clock stable.

When chip wakes up from stop mode, SPM will keep XOSC/SYSPLL on or off same as state before sleep. But when wakeup from standby mode, XOSC/SYSPLL will be off.

6.5 Register mapping

Table 6-1 SPM Register Map

SPM: 0x40008000

ADDRESS	TITLE	DESCRIPTION
SPM + 0x000	REG_PWR_MGR_CFG0	Power Manager Configuration 0 Register
SPM + 0x004	REG_PWR_MGR_CFG1	Power Manager Configuration 1 Register
SPM + 0x00C	REG_PERIPH_SLEEP_ACK_STATUS	Periph Sleep Ack Status
SPM + 0x010	REG_EN_PERIPH_SLEEP_ACK_WAIT	Periph Sleep Ack Waiting Enable Register
SPM + 0x014	REG_EN_PERIPH_WKUP	Periph Wakeup Enable Register
SPM + 0x018	REG_EN_TRIGGER_SPM_IRQ	Periph Trigger SPM IRQ Enable Register
SPM + 0x01C	REG_SPM_WAKEUP_IRQ_STATUS	SPM Wakeup IRQ Flags Status Register

SPM + 0x000 REG_PWR_MGR_CFG0 Power Manager Configuration 0 Register

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name							REG_PWR_MGR_CFG0[9:0]									
Type							RW									
Reset							0	1	0	0	0	1	1	0	0	0

Bit Field	Name	Description
31:10	reserved	
9:8	REG_SLEEP_MODE	sleep mode 2'b00:stop mode lite 2'b01:stop mode 2'b1x:standby mode
7	REG_EN_IO_SUS_STOP_MODE	enable IO suspend in stop mode/stop lite mode 1'b1:I/O suspend when enter stop mode/stop lite mode This ctrl bits does not affect in standby mode, hardware auto suspend io.
6	REG_EN_CAN1_FILTER	enable can1 wakeup int filter 1'b1:enable when enable SPM will use int after analog filter as can2 wakeup int.
5	REG_EN_CAN0_FILTER	enable can0 wakeup int filter 1'b1:enable when enable SPM will use int after analog filter as can1 wakeup int.

4	REG_EN_PWRLVD	chip lower voltage detect control bit 1'b1:enable AD_MCU_PORLPVD_PWRLVD_RST=0 @ DA_MCU_PORLPVD_PWRLVD_PD=0, AVDD50<VLVD
3	Reserved	
2	REG_EN_PWRWARN	chip lower voltage warning control bit 1'b1:enable AD_MCU_PORLPVD_PWR_WARNING=1 @DA_MCU_PORLPVD_PWRWARN_PD=0, AVDD50<VPVD
1	REG_EN_FAST_BOOT	enable fast boot mode 1'b1:enable fast boot mode: to save wakeup time, chip will stop sleep sequence and wake up immediately when receive a wakeup interrupt.
0	REG_PWR_EN	SPM power control enable 1'b1: enable. enable SPM power control.

SPM + 0x004

REG_PWR_MGR_CFG1

Power Manager Configuration 1 Register

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type	R	R	R/W	R/W	R/W											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									REG_PWR_MGR_CFG1[7:0]							
Type									R/W							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit Field	Name	Description
31	XOSC_RDY	XOSC clock ready flag 1:ready
30	SYSPLL_RDY	PLL clock ready flag 1:ready
29	REG_XOSC_HSEON	External high-speed clock enable 1'b1: enable XOSC
28	REG_XOSC_HSEBYP	External high-speed clock bypass 1'b1: bypassing the oscillator with an external clock
27	REG_SYSPLL_ON	SYSPLL enable 1'b1:SPM power SYSPLL on
26:7	reserved	
6:4	reserved	

PVDLVD setting

4'b0000: VLVDL = 2.65V VPVDL_0 = 2.7V

4'b0x01: VLVDL = 2.65V VPVDL_0 = 2.8V

4'b0x10: VLVDL = 2.65V VPVDL_0 = 2.9V

4'b0x11: VLVDL = 2.65V VPVDL_0 = 3.0V

4'b1x00: VLVDL = 4.3V VPVDL_0 = 4.4V

4'b1x01: VLVDL = 4.3V VPVDL_0 = 4.5V

4'b1x10: VLVDL = 4.3V VPVDL_0 = 4.6V

4'b1x11: VLVDL = 4.3V VPVDL_0 = 4.7V

SPM + 0x0c

REG_PERIPH_SLEEP_ACK_STATUS

Periph Sleep Ack Status Register

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name														eflash	gpio	adc
Type														R	R	R
Reset														0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	dma	uart6	uart5	uart4	uart3	uart2	uart1	can1	can0	lin	spi2	spi1	iic2	iic1	acmp1	acmp0
Type	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit Field	Name	Description
31:19	reserved	
18	eflash	eflash idle status 1:eflash idle
17	gpio	gpio sleep ack status. 1'b1:ack
16	adc	adc sleep ack status. 1'b1:ack
15	dma	dma sleep ack status. 1'b1:ack
14	uart6	uart6 sleep ack status. 1'b1:ack
13	uart5	uart5 sleep ack status. 1'b1:ack
12	uart4	uart4 sleep ack status. 1'b1:ack
11	uart3	uart3 sleep ack status. 1'b1:ack
10	uart2	uart2 sleep ack status. 1'b1:ack
9	uart1	uart1 sleep ack status. 1'b1:ack
8	can1	can1 sleep ack status. 1'b1:ack
7	can0	can0 sleep ack status.

		1'b1:ack
6	lin	lin sleep ack status. 1'b1:ack
5	spi2	spi2 sleep ack status. 1'b1:ack
4	spi1	spi1 sleep ack status. 1'b1:ack
3	iic2	iic2 sleep ack status. 1'b1:ack
2	iic1	iic1 sleep ack status. 1'b1:ack
1	acmp1	acmp1 sleep ack status. 1'b1:ack
0	acmp0	acmp0 sleep ack status. 1'b1:ack

SPM + 0x010

REG_EN_PERIPH_SLEEP_ACK

Periph Sleep Ack Waiting Enable Register

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name															gpio	adc
Type															R/W	R/W
Reset															1	1
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	dma	uart6	uart5	uart4	uart3	uart2	uart1	can1	can0	lin	spi2	spi1	iic2	iic1	acmp1	acmp0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit Field	Name	Description
31:18	reserved	
17	gpio	enable gpio sleep ack waiting. 1'b1: enable. SPM wait for gpio sleep ack before enter sleep sequence. when disable SPM will not check gpio sleep ack reply before enter sleep sequence.
16	adc	enable adc sleep ack waiting. 1'b1:enable.
15	dma	enable dma sleep ack waiting. 1'b1:enable.
14	uart6	enable uart6 sleep ack waiting. 1'b1:enable.
13	uart5	enable uart5 sleep ack waiting. 1'b1:enable.
12	uart4	enable uart4 sleep ack waiting. 1'b1:enable.
11	uart3	enable uart3 sleep ack waiting. 1'b1:enable.
10	uart2	enable uart2 sleep ack waiting. 1'b1:enable.

9	uart1	enable uart1 sleep ack waiting. 1'b1:enable.
8	can1	enable can1 sleep ack waiting. 1'b1:enable.
7	can0	enable can0 sleep ack waiting. 1'b1:enable.
6	lin	enable lin sleep ack waiting. 1'b1:enable.
5	spi2	enable spi2 sleep ack waiting. 1'b1:enable.
4	spi1	enable spi1 sleep ack waiting. 1'b1:enable.
3	iic2	enable iic2 sleep ack waiting. 1'b1:enable.
2	iic1	enable iic1 sleep ack waiting. 1'b1:enable.
1	acmp1	enable acmp1 sleep ack waiting. 1'b1:enable.
0	acmp0	enable acmp0 sleep ack waiting. 1'b1:enable.

SPM + 0x014

REG_EN_PERIPH_WAKEUP

Periph Wakeup Enable Register

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name													lvd_warning	nmi	gpio	adc
Type													R/W	R/W	R/W	R/W
Reset													0	0	1	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	rtc	uart6	uart5	uart4	uart3	uart2	uart1	can1	can0	lin	spi2	spi1	iic2	iic1	acmp1	acmp0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit Field	Name	Description
31:20	reserved	
19	lvd warning	enable lvd warning wakeup 1'b1: enable. When disable,lvd warning wakeup is not supported, and this wakeup int will be ignored.
18	nmi	enable nmi wakeup 1'b1:enable
17	gpio	enable gpio wakeup. 1'b1:enable.
16	adc	enable adc wakeup. 1'b1:enable.
15	rtc	enable rtc wakeup. 1'b1:enable.

14	uart6	enable uart6 wakeup. 1'b1:enable.
13	uart5	enable uart5 wakeup. 1'b1:enable.
12	uart4	enable uart4 wakeup. 1'b1:enable.
11	uart3	enable uart3 wakeup. 1'b1:enable.
10	uart2	enable uart2 wakeup. 1'b1:enable.
9	uart1	enable uart1 wakeup. 1'b1:enable.
8	can1	enable can1 wakeup. 1'b1:enable.
7	can0	enable can0 wakeup. 1'b1:enable.
6	lin	enable lin wakeup. 1'b1:enable.
5	spi2	enable spi2 wakeup. 1'b1:enable.
4	spi1	enable spi1 wakeup. 1'b1:enable.
3	iic2	enable iic2 wakeup. 1'b1:enable.
2	iic1	enable iic1 wakeup. 1'b1:enable.
1	acmp1	enable acmp1 wakeup. 1'b1:enable.
0	acmp0	enable acmp0 wakeup. 1'b1:enable.

SPM + 0x018

REG_EN_TRIGGER_IRQ

Periph Trigger SPM IRQ Enable Register

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name												over count	porlpvd warn	nmi	gpio	adc
Type												R/W	R/W	R/W	R/W	R/W
Reset												1	0	1	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	rtc	uart6	uart5	uart4	uart3	uart2	uart1	can1	can0	lin	spi2	spi1	iic2	iic1	acmp1	acmp0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0

Bit Field	Name	Description
31:22	reserved	

Bit Field	Name	Description
20	SPM over count	enable SPM over count irq 1'b1: enable. (1).counter overflow: before enter sleep state, SPM waiting for all periph reply sleep ack in certain cycle. If not, timer counter overflow. (2).when enables SPM over count & counter overflow, trigger spm_irq to wakeup cm3. (3).for periph (e.g. rtc/adc/gpio/..) trigger wakeup irq themselves, just set corresponding bit 0.
19	porlpvd warning	enable porlpvd warning irq 1'b1: enable.
18	nmi	enable nmi irq 1'b1:enable
17	gpio	enable gpio irq. 1'b1:enable.
16	adc	enable adc irq. 1'b1:enable.
15	rtc	enable rtc irq. 1'b1:enable.
14	uart6	enable uart6 irq. 1'b1:enable.
13	uart5	enable uart5 irq. 1'b1:enable.
12	uart4	enable uart4 irq. 1'b1:enable.
11	uart3	enable uart3 irq. 1'b1:enable.
10	uart2	enable uart2 irq. 1'b1:enable.
9	uart1	enable uart1 irq. 1'b1:enable.
8	can1	enable can1 irq. 1'b1:enable.
7	can0	enable can0 irq. 1'b1:enable.
6	lin	enable lin irq. 1'b1:enable.
5	spi2	enable spi2 irq. 1'b1:enable.
4	spi1	enable spi1 irq. 1'b1:enable.
3	iic2	enable iic2 irq. 1'b1:enable.
2	iic1	enable iic1 irq. 1'b1:enable.
1	acmp1	enable acmp1 irq. 1'b1:enable.

Bit Field	Name	Description
0	acmp0	enable acmp0 irq. 1'b1:enable.

SPM + 0x01c REG_SPM_WAKEUP_IRQ_STATUS SPM Wakeup IRQ Flags Register

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name												over count	porlpvd warn	nmi	gpio	adc
Type												R	R	R	R	R
Reset												0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	rtc	uart6	uart5	uart4	uart3	uart2	uart1	can1	can0	lin	spi2	spi1	iic2	iic1	acmp1	acmp0
Type	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit Field	Name	Description
31:21	reserved	
20	spm over count	SPM over count irq flag 1'b1: SPM counter trigger this spm_irq (1). when spm_irq trigger and cm3 be wakeup, cm3 will read this register to determine which periph trigger this wakeup. (2). cm3 should write corresponding bit 1 back to clear spm_irq.
19	porlpvd warning	porlpvd warning irq flag
18	nmi	nmi irq flag
17	gpio	gpio irq flag.
16	adc	adc irq flag.
15	rtc	rtc irq flag.
14	uart6	uart6 irq flag.
13	uart5	uart5 irq flag.
12	uart4	uart4 irq flag.
11	uart3	uart3 irq flag.
10	uart2	uart2 irq flag.
9	uart1	uart1 irq flag.
8	can1	can1 irq flag.
7	can0	can0 irq flag.
6	lin	lin irq flag.
5	spi2	spi2 irq flag.
4	spi1	spi1 irq flag.
3	iic2	iic2 irq flag.
2	iic1	iic1 irq flag.
1	acmp1	acmp1 irq flag.
0	acmp0	acmp0 irq flag.

7 CAN

7.1 Introduction

7.1.1 The CAN-CTRL core

The CAN-CTRL core is a serial communications controller that performs serial communication according to the CAN protocol. This CAN bus interface uses the basic CAN principle and meets all constraints of the CAN-specification 2.0B active.

The CAN protocol uses a multi-master bus configuration for the transfer of frames (communication objects) between nodes of the network and manages the error handling without any burden on the CPU. The CAN-CTRL bus controller enables the user to set up economic and reliable links between various components. The CAN-CTRL core appears to a microcontroller as a memory-mapped I/O device. A CPU accesses the CAN-CTRL core to control transmission or reception of frames through a two wire CAN bus system. The connection to a CAN bus is illustrated in Figure 7-1.

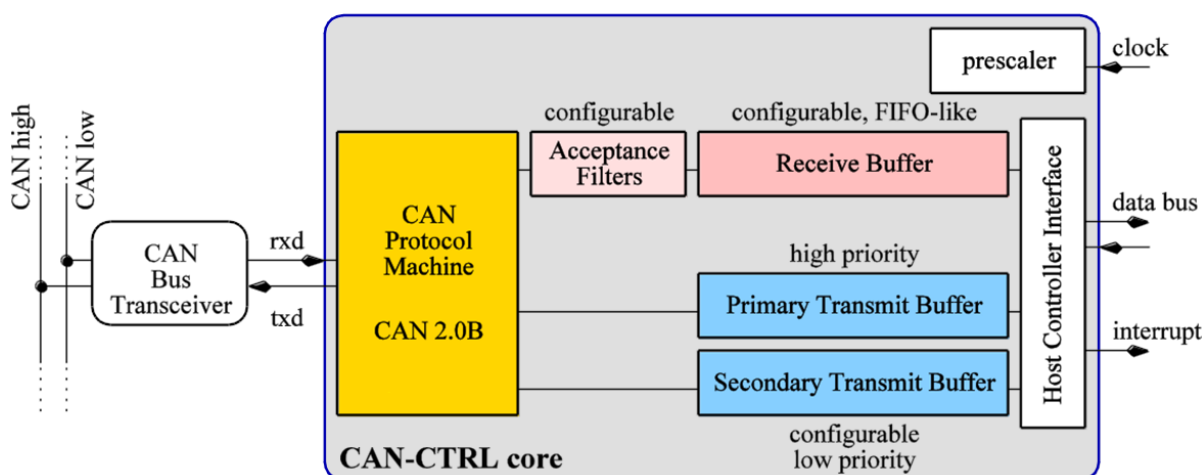


Figure 7-1 Connection to CAN bus and main features of the CAN-CTRL core

7.1.2 The CAN protocol

CAN communication is organized in frames. Two types of frames exist: standard and extended frames. For CAN 2.0, the maximum data payload is up to 8 bytes can be transmitted using one frame.

All CAN nodes are equal in terms of bus access. There is no super-node because the CAN is a multi-master bus.

Data addressing is done using message identifiers. In a CAN network, only one node shall transmit messages with a certain identifier. All nodes receive all messages and the node host controller has to decide if it was addressed by the appropriate message identifier. To reduce the load of a host controller, a CAN core may use acceptance filters. These filters compare all received message identifiers to user-selectable bit patterns. Only if a message passes an acceptance filter, it will be stored in the receive buffer and signaled to the host controller.

The identifiers of CAN frames are also used for bus arbitration. The CAN protocol machine stops transmission of a message with a low-priority identifier when a message with a higher priority identifier

is transmitted by another CAN node. The CAN protocol machine automatically attempts to re-transmit the stopped message at the next possible transmit position.

CAN 2.0B defines data bit rates up to 1 Mbit/s.

7.2 Feature list

- Support CAN specification: CAN 2.0A/B (up to 8 bytes payload, verified by Bosch reference model).
- Free programmable data rate: CAN 2.0B defines data rates up to 1Mbit/s.
- Programmable baud rate prescaler (1 to 1/256).
- Nine receive buffers with FIFO-like behavior.
- Two transmit buffers:
 - One Primary Transmit Buffer (PTB);
 - Five Secondary Transmit Buffer (STB), operation in FIFO or priority decision mode.
- 16 Independent and programmable internal 29 bits acceptance filters.
- Extended features:
 - Single Shot Transmission Mode (for PTB and / or for STB).
 - Listen Only Mode.
 - Loop Back Mode (internal and external).
 - Transceiver Standby Mode.
- Extended status and error report:
 - Capturing of last occurred kind of error and of arbitration lost position.
 - Programmable Error Warning Limit.
- Configurable interrupt sources.
- Time-stamping
 - ISO 11898-4 Time-Triggered CAN with partial hardware support.
- Programmable wake-up functionality with integrated low-pass filter.

7.3 Message buffers

7.3.1 Message buffers concept

The concept of the message buffers is illustrated. This schematic focuses on the buffers and hides other details of the CAN-CTRL core. All buffer slots are big enough to store frames with the maximum length.

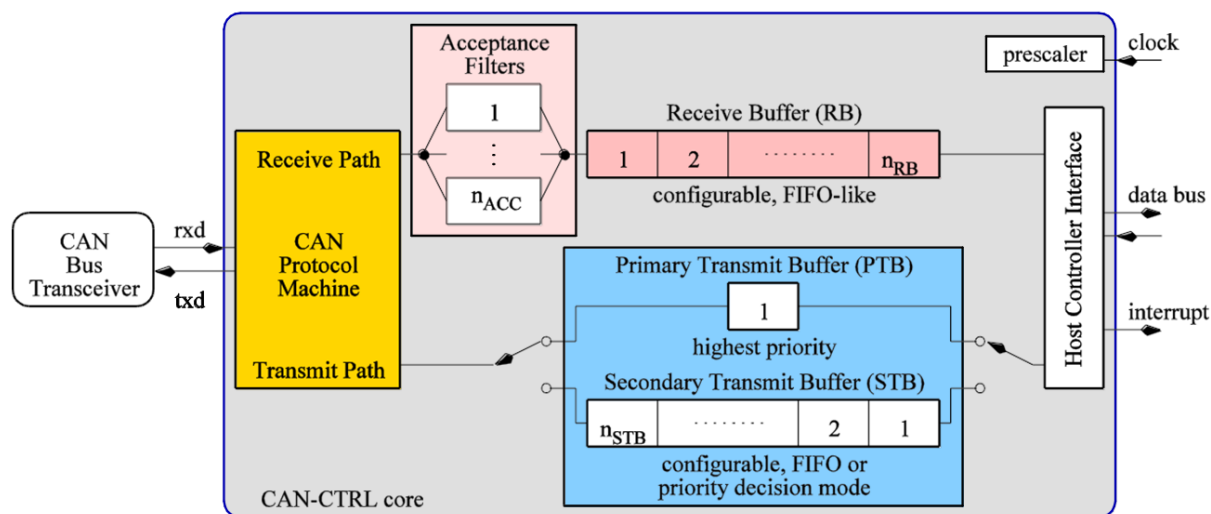


Figure 7-2 Message buffers concept

7.3.2 Receive buffer

To reduce the load of received frames for the host controller, the core uses acceptance filters. The CAN-CTRL core checks the message identifier during acceptance filtering. If the received frame matches the filter criteria of one of the acceptance filters, then it will be stored in the Receive Buffer (RB), which has FIFO-like behavior.

Depending on the number of available message slots, the host controller does not need to read incoming messages immediately. The CAN-CTRL core is able to generate interrupts upon every received message, when the RB is full or filled to a user-selectable “almost full” limit. Because of the FIFO-like behavior, the host controller always reads the oldest message from the RB.

7.3.3 Transmit buffer

For frame transmission purposes, two Transmit Buffers (TB) are offered. The Primary TB (PTB) has a higher priority, but is able to buffer only one frame. The Secondary TB (STB) has a lower priority. It can act in FIFO or in priority mode. The priority decision between PTB and STB is fixed and fully independent from the CAN bus arbitration. Bus arbitration is a priority decision based on the frame identifiers.

The STB can be commanded to transmit one or all stored frames. In FIFO mode with every transmission, the oldest frame inside this buffer is transmitted first. In priority mode, the frame with the highest priority inside this buffer (based on the frame identifier) is transmitted first.

A frame located in the PTB has always a higher priority for the CAN protocol machine than the frames in the STB regardless of the frame identifiers. A PTB transmission stops and delays an STB transmission. The STB transmission is automatically restarted after the PTB frame has been successfully transmitted.

A PTB transmission starts at the next transmit position that is possible by the CAN protocol (after the next interframe space). Because of this, a STB transmission that has won the arbitration and is actually transmitted, will be completed before.

Interrupting STB transmissions using a PTB transmission may happen in the following cases:

1. The STB is commanded to output all stored frames and the host controller decides to command a PTB transmission before all STB transmissions are completed.

2. The STB is commanded to output a single frame and the host controller decides to command a PTB transmission before the STB transmission is completed.

If the host controller waits until each commanded transmission is completed, then it can easily decide which buffer shall transmit the next frame. As a drawback a message with a low-priority identifier may block more important messages, then the host could abort the message.

7.4 Register definition

CAN-CTRL is a 32 bit component and offers downward-compatible interfaces for 8 and 16 bit hosts. The mapping of the registers for 8 host interfaces.

Table 7-1 CAN-CRTL Register Mapping

CAN_1: **0x40007800**

CAN_2: **0x40007C00**

	Bit position								Register name
	7	6	5	4	3	2	1	0	
0x00 to 0x4F	Receive Buffer Registers and Reception Time Stamp								RBUF (and RTS)
0x50 to 0x97	Transmit Buffer Registers								TBUF
0xA0	RESET	LBME	LBMI	TPSS	TSSS	RACTIVE	TACTIVE	BUSOFF	CFG_STAT
0xA1	TBSEL	LOM	STBY	TPE	TPA	TSONE	TSALL	TSA	TCMD
0xA2	-	TSNEXT	TSMODE	-			TSSTAT(1:0)		TCTRL
0xA3	-	ROM	ROV	RREL	-		RSTAT(1:0)		RCTRL
0xA4	RIE	ROIE	RFIE	RAFIE	TPIE	TSIE	EIE	TSFF	RTIE
0xA5	RIF	ROIF	RFIF	RAFIF	TPIF	TSIF	EIF	AIF	RTIF
0xA6	EWARN	EPASS	EPIE	EPIF	ALIE	ALIF	BEIE	BEIF	ERRINT
0xA7	AFWL(3:0)				EWL(3:0)				LIMIT
0xA8	S_Seg_1(7:0)								S_Seg_1
0xA9	-	S_Seg_2(6:0)							S_Seg_2
0xAA	-	S_SJW(6:0)							S_SJW
0xAB	S_PRESC(7:0)								S_PRESC
0xB0	KOER(2:0)			ALC(4:0)					EALCAP
0xB2	RECNT								RECNT
0xB3	TECNT								TECNT
0xB4	-		SELMASK	-	ACFADR				ACFCTRL
0xB6	AE_7	AE_6	AE_5	AE_4	AE_3	AE_2	AE_1	AE_0	ACF_EN_0
0xB7	AE_15	AE_14	AE_13	AE_12	AE_11	AE_10	AE_9	AE_8	ACF_EN_1
0xB8	ACODE_x or AMASK_x (7:0)								ACF_0
0xB9	ACODE_x or AMASK_x (15:8)								ACF_1
0xBA	ACODE_x or AMASK_x (23:16)								ACF_2
0xBB	-	AIDEE	AIDE	ACODE_x or AMASK_x (28:24)					ACF_3
0xBC	VERSION(7:0)								VER_0

0xBD	VERSION(15:8)	VER_1
------	---------------	-------

0x0A0 CFG_STAT Configuration and Status Register

Bits	Name	Access	Function
7	RESET	rw-1	Reset request bit 1: The host controller performs a local reset of CAN-CTRL. 0: no local reset of CAN-CTRL. Some register (e.g. for node configuration) can only be modified if RESET=1. Bit RESET forces several components to a reset state. RESET is automatically set if the node enters "bus off" state. Note: A CAN node will participate in CAN communication after RESET is switched to 0 after 11 CAN bit times. This delay is required by the CAN standard (bus idle time). If RESET is set to 1 and immediately set to 0, then it takes some time until RESET can be read as 0 and becomes inactive. The reason is clock domain crossing from host to CAN clock domain. RESET is held active as long as needed depending on the relation between host and
6	LBME	rw-0	Loop Back Mode, External 0: Disabled 1: Enabled LBME should not be enabled while a transmission is active.
5	LBMI	rw-0	Loop Back Mode, Internal 0: Disabled 1: Enabled LBMI should not be enabled while a transmission is active.
4	TPSS	rw-0	Transmission Primary Single Shot mode for PTB 0: Disabled 1: Enabled
3	TSSS	rw-0	Transmission Secondary Single Shot mode for STB 0: Disabled 1: Enabled
2	RACTIVE	r-0	Reception Active (Receive Status bit) 1: The controller is currently receiving a frame. 0: No receive activity.
1	TACTIVE	r-0	Transmission Active (Transmit Status bit) 1: The controller is currently transmitting a frame. 0: No transmit activity.
0	BUSOFF	rw-0	Bus Off (Bus Status bit) 1: The controller status is "bus off". 0: The controller status is "bus on". Writing a 1 will clear TECNT and RECNT and exit BUSOFF when in BUSOFF status. This should be done only for debugging.

0x0A1 TCMD Command Register

Bits	Name	Access	Function
7	TBSEL	rw-0	Transmit Buffer Select Selects the transmit buffer to be loaded with a message. Use the TBUF registers for access. TBSEL needs to be stable all the time the TBUF registers are written and when TSNEXT is set. 0: PTB (high-priority buffer) 1: STB

Bits	Name	Access	Function
6	LOM	rw-0	Listen Only Mode 0: Disabled 1: Enabled LOM should not be enabled while a transmission is active. No transmission can be started if LOM is enabled.
5	STBY	rw-0	Transceiver Standby Mode 0: Disabled 1: Enabled This register bit is connected to the output signal standby which can be used to control a standby mode of a transceiver. STBY cannot be set to 1 if TPE=1, TSONE=1 or TSALL=1. If the host sets STBY to 0 then the host needs to wait for the time required by the transceiver to start up before the host requests a new transmission.
4	TPE	rw-0	Transmit Primary Enable 1: Transmission enable for the message in the high-priority PTB 0: No transmission for the PTB If TPE is set, the message from the PTB will be transmitted at the next possible transmit position. A started transmission from the STB will be completed before, but pending new messages are delayed until the PTB message has been transmitted. TPE stays set until the message has been transmitted successfully or it is aborted using TPA. The host controller can set TPE to 1 but cannot reset it to 0. This would only be possible using TPA and aborting the message. During the short time while the CAN-CTRL core resets the bit, it cannot be set by the host. The bit will be reset to the hardware reset value if RESET=1, BUSOFF=1, STBY=1, LOM=1.
3	TPA	rw-0	Transmit Primary Abort 1: Aborts a transmission from PTB which has been requested by TPE=1 but not started yet. (The data bytes of the message remains in the PTB.) 0: no abort. The bit has to be set by the host controller and will be reset by CAN-CTRL. Setting TPA automatically de-asserts TPE. The host controller can set TPA to 1 but cannot reset it to 0. During the short time while the CAN-CTRL core resets the bit, it cannot be set by the host. The bit will be reset to the hardware reset value if RESET=1, BUSOFF=1. TPA should not be set simultaneously with TPE.
2	TSONE	rw-0	Transmit Secondary One frame 1: Transmission enable of one in the STB. In FIFO mode this is the oldest message and in priority mode this is the one with the highest priority. TSONE in priority mode is difficult to handle, because it is not always clear which message will be transmitted if new messages are written to the STB meanwhile. The controller starts the transmission as soon as the bus becomes vacant and no request of the PTB (bit TPE) is pending. 0: No transmission for the STB. TSONE stays set until the message has been transmitted successfully or it is aborted using TSA. The host controller can set TSONE to 1 but cannot reset it to 0. This would only be possible using TSA and aborting the message. During the short time while the CAN-CTRL core resets the bit, it cannot be set by the host. The bit will be reset to the hardware reset value if RESET=1, BUSOFF=1, STBY=1, LOM=1.

Bits	Name	Access Function
1	TSALL	rw-0
Transmit Secondary All frames 1: Transmission enable of all messages in the STB. The controller starts the transmission as soon as the bus becomes vacant and no request of the PTB (bit TPE) is pending. 0: No transmission for the STB. TSALL stays set until all messages have been transmitted successfully or they are aborted using TSA. The host controller can set TSALL to 1 but cannot reset it to 0. This would only be possible using TSA and aborting the messages. During the short time while the CAN-CTRL core resets the bit, it cannot be set by the host. The bit will be reset to the hardware reset value if RESET=1, BUSOFF=1, STBY=1, LOM=1. If during a transmission the STB is loaded with a new frame, then the new frame will be transmitted too. In other words: a transmission initiated by TSALL is finished when the STB becomes empty.		
0	TSA	rw-0
Transmit Secondary Abort 1: Aborts a transmission from STB which has been requested but not started yet. For a TSONE transmission, only one frame is aborted while for a TSALL Transmission, all frames are aborted. One or all message slots will be released which updates TSSTAT. All aborted messages are lost because they are not accessible any more. If in priority mode a TSONE transmission is aborted, then it is not clear which frame will be aborted if new frames are written to the STB meanwhile. 0: no abort. The bit has to be set by the host controller and will be reset by CAN-CTRL. Setting TSA, automatically de-asserts TSONE or TSALL respectively. The host controller can set TSA to 1 but cannot reset it to 0. The bit will be reset to the hardware reset value if RESET=1 or BUSOFF=1. TSA should not be set simultaneously with TSONE or TSALL.		

Setting both TSONE and TSALL is meaningless. While TSALL is already set, it is impossible to set TSONE and vice versa. If both TSONE and TSALL are set simultaneously then TSALL wins and TSONE is cleared by the CAN-CTRL core.

0x0A2 TCTRL

Transmit Control Register

Bits	Name	Access Function
6	TSNEXT	rw-0
Transmit buffer Secondary Next 0 : no action 1 : STB slot filled, select next slot. After all frame bytes are written to the TBUF registers, the host controller has to set TSNEXT to signal that this slot has been filled. Then the CAN-CTRL core connects the TBUF registers to the next slot. Once a slot is marked as filled a transmission can be started using TSONE or TSALL. It is possible to set TSNEXT and TSONE or TSALL together in one write access. TSNEXT has to be set by the host controller and is automatically reset by the CAN-CTRL core immediately after it was set. Setting TSNEXT is meaningless if TBSEL=0. In this case TSNEXT is ignored and automatically cleared. It does not do any harm. If all slots of the STB are filled, TSNEXT stays set until a slot becomes free.		

Bits	Name	Access	Function
5	TSMODE	rw-0	Transmit buffer Secondary Operation Mode 0: FIFO mode 1: priority decision mode In FIFO mode frames are transmitted in the order in that they are written into the STB. In priority decision mode the frame with the highest priority in the STB is automatically transmitted first. The ID of a frame is used for the priority decision. A lower ID means a higher priority of a frame. A frame in the PTB has always the highest priority regardless of the ID. TSMODE shall be switched only if the STB is empty.
4:2	-	r-0	Reserved
1:0	TSSTAT	r-0	Transmission Secondary Status bits 00: STB is empty 01: STB is less than or equal to half full 10: STB is more than half full 11: STB is full

0x0A3 RCTRL

Receive Control Register

Bits	Name	Access	Function
7	-	r-0	reserved
6	ROM	rw-0	Receive buffer Overflow Mode In case of a full RBUF when a new message is received, then ROM selects the following: 1: The new message will not be stored. 0 : The oldest message will be overwritten.
5	ROV	r-0	Receive buffer Overflow 1: Overflow. At least one message is lost. 0: No Overflow. ROV is cleared by setting RREL=1.
4	RREL	rw-0	Receive buffer Release The host controller has read the actual RB slot and releases it. Afterwards the CAN-CTRL core points to the next RB slot. RSTAT gets updated. 1: Release: The host has read the RB. 0 : No release
3:2	-	r-0	Reserved
1:0	RSTAT	r-0	Receive buffer Status 00: empty 01: > empty and < almost full (AFWL) 10: almost full (programmable threshold by AFWL) but not full and no overflow 11: full (stays set in case of overflow – for overflow signaling see ROV)

0x0A4 RTIE

Receive and Transmit Interrupt Enable Register

Bits	Name	Access	Function
7	RIE	rw-1	Receive Interrupt Enable 0 : Disabled, 1 : Enabled
6	ROIE	rw-1	RB Overrun Interrupt Enable 0 : Disabled, 1 : Enabled
5	RFIE	rw-1	RB Full Interrupt Enable 0:Disabled, 1 : Enabled
4	RAFIE	rw-1	RB Almost Full Interrupt Enable 0 : Disabled, 1 : Enabled
3	TPIE	rw-1	Transmission Primary Interrupt Enable 0 :Disabled, 1 :Enabled
2	TSIE	rw-1	Transmission Secondary Interrupt Enable 0 : Disabled, 1 : Enabled
1	EIE	rw-1	Error Interrupt Enable 0 : Disabled, 1 :Enabled

Bits	Name	Access	Function
0	TSFF	r-0	Transmit Secondary Buffer Full Flag 1: The STB is filled with the maximal number of messages. 0 : The STB is not filled with the maximal number of messages

0x0A5 RTIF

Receive and Transmit Interrupt Flag Register

Bits	Name	Access	Function
7	RIF	rw-0	Receive Interrupt Flag 1: Data or a remote frame has been received and is available in the receive buffer. 0: No frame has been received.
6	ROIF	rw-0	RB Overrun Interrupt Flag 1: At least one received message has been overwritten in the RB. 0: No RB overwritten. In case of an overrun both ROIF and RFIF will be set.
5	RFIF	rw-0	RB Full Interrupt Flag 1: All RBs are full. If no RB will be released until the next valid message is received, the oldest message will be lost. 0: The RB FIFO is not full.
4	RAFIF	rw-0	RB Almost Full Interrupt Flag 1: number of filled RB slots \geq AFWL _i 0: number of filled RB slots $<$ AFWL _i
3	TPIF	rw-0	Transmission Primary Interrupt Flag 1: The requested transmission of the PTB has been successfully completed. 0: No transmission of the PTB has been completed.
2	TSIF	rw-0	Transmission Secondary Interrupt Flag 1: The requested transmission of the STB has been successfully completed. 0: No transmission of the STB has been completed successfully.
1	EIF	rw-0	Error Interrupt Flag 1: The border of the error warning limit has been crossed in either direction, or the BUSOFF bit has been changed in either direction. 0: There has been no change.
0	AIF	rw-0	Abort Interrupt Flag 1: After setting TPA or TSA the appropriated message(s) have been aborted. It is recommended to not set both TPA and TSA simultaneously because both source AIF. 0: No abort has been executed. The AIF does not have an associated enable register.

0x0A6 ERRINT

Error Interrupt Enable and Flag Register

Bits	Name	Access	Function
7	EWARN	r-0	Error Warning limit reached 1: One of the error counters RECNT or TECNT is equal or bigger than EWL 0 : The values in both counters are less than EWL.
6	EPASS	r-0	Error Passive mode active 0: not active (node is error active) 1 : active (node is error passive)
5	EPIE	rw-0	Error Passive Interrupt Enable
4	EPIF	rw-0	Error Passive Interrupt Flag. EPIF will be activated if the error status changes from error active to error passive or vice versa and if this interrupt is enabled.
3	ALIE	rw-0	Arbitration Lost Interrupt Enable
2	ALIF	rw-0	Arbitration Lost Interrupt Flag

Bits	Name	Access	Function
1	BEIE	rw-0	Bus Error Interrupt Enable
0	BEIF	rw-0	Bus Error Interrupt Flag

To reset an interrupt flag, the host controller needs to write a 1 to the flag. Writing a 0 has no effect. If a new interrupt event occurs while the write access is active, then this event will set the flag and override the reset. This ensures that no interrupt event is lost.
Interrupt flags will only be set if the associated interrupt enable bit is set.

0x0A7 LIMIT Warning Limits Register

Bits	Name	Access	Function
Receive buffer Almost Full Warning Limit			
AFWL defines the internal warning limit AFWL _i with n_{RB} being the number of available RB slots. AFWL _i is compared to the number of filled RB slots and triggers RAFIF if equal. The valid range of AFWL _i : 1... n_{RB} . AFWL _i = 0 is meaningless and automatically treated as 0x1. (Note that AFWL is meant in this rule and not AFWL _i .) AFWL _i > n_{RB} is meaningless and automatically treated as n_{RB} . AFWL _i = n_{RB} is a valid value, but note that RFIF also exists.			
7:4	AFWL(3:0)	rw-0x1	
Programmable Error Warning Limit = (EWL+1) *8. Possible Limit values: 8, 16, ... 128. The value of EWL controls EIF.			
3:0	EWL(3:0)	rw-0xB	

0x0A8 S_Seg_1 Bit Timing Register

Bits	Name	Access	Function
Bit Timing Segment 1			
7:0	S_Seg_1(7:0)	rw-0x3	The sample point will be set to $t_{Seg_1} = (Seg_1 + 2) \cdot TQ$ after start of bit time.

0x0A9 S_Seg_2 Bit Timing Register

Bits	Name	Access	Function
7	-	r-0	Reserved
Bit Timing Segment 2			
6:0	S_Seg_2(6:0)	rw-0x2	Time $t_{Seg_2} = (Seg_2 + 1) \cdot TQ$ after the sample point. Please note the synthesis parameter SHORT_SEG2.

0x0AA S_SJW Bit Timing Register

Bits	Name	Access	Function
7	-	r-0	Reserved
Synchronization Jump Width			
6:0	S_SJW(6:0)	rw-0x2	The Synchronization Jump Width $t_{SJW} = (SJW + 1) \cdot TQ$ is the maximum time for shortening or lengthening the Bit Time for resynchronization, where TQ is a time quanta.

0x0AB S_PRESC Prescaler Registers

Bits	Name	Access	Function
7:0	S_PRES C	rw-0x01	Prescaler The prescaler divides the system clock to get the time quanta clock tq_clk. Valid range PRESC=[0x00, 0xff] results in divider values 1 to 256.

0x0B0 EALCAP Error and Arbitration Lost Capture Register

Bits	Name	Reset Value	Function
7:5	KOER(2:0)	r-0x0	Kind Of Error (Error code) 000 : no error 001 : BIT ERROR 010 : FORM ERROR 011 : STUFF ERROR 100 : ACKNOWLEDGEMENT ERROR 101 : CRC ERROR 110: OTHER ERROR (dominant bits after own error flag, received active Error Flag too long, dominant bit during Passive-Error-Flag after ACK error). 111: not used KOER is updated with each new error. Therefore, it stays untouched when frames are successfully transmitted or received.
4:0	ALC(4:0)	r-0x0	Arbitration Lost Capture (bit position in the frame where the arbitration has been lost)

0x0B2-B3 RECNT Error Counter Registers (TECNT (0xB3))

Bits	Name	Access	Function
7:0	RECNT	r-0x00	Receive Error Count (number of errors during reception) RECNT is incremented and decremented as defined in the CAN specification. RECNT does not overflow. RECNT signals 0xff = 255 as maximum value. For more details about RECNT and the "bus off" state.
7:0	TECNT	r-0x00	Transmit Error Count (number of errors during transmission) TECNT is incremented and decremented as defined in the CAN specification. TECNT does not overflow. TECNT signals 0xff = 255 as maximum value. For more details about TECNT and the "bus off" state.

0x0B4 ACFCTRL Acceptance Filter Control Register

Bits	Name	Access	Function
7:6	-	r-0	reserved
5	SELMASK	rw-0	Select acceptance Mask 0 : Registers ACF_x point to acceptance code 1 : Registers ACF_x point to acceptance mask. ACFADR selects one specific acceptance filter.
4	-	r-0	Reserved
3:0	ACFADR	rw-0	Acceptance filter address ACFADR points to a specific acceptance filter. The selected filter is accessible using the registers ACF_x. Bit SELMASK selects between acceptance code and mask for the selected acceptance filter. A value of ACFADR>NR_OF_ACF-1 is meaningless and automatically treated as value NR_OF_ACF-1.

The acceptance filter registers ACF_x provide access to the acceptance filter codes ACODE_x and acceptance filter masks AMASK_x depending on the setting of SELMASK. Read/ Write access to ACF_x is only possible if RESET=1. If RESET=0 then reading from ACF_x results in the value 0.

The acceptance filters are build using a true 32 bits wide memory and therefore a write access needs to be performed as 32 bits write.

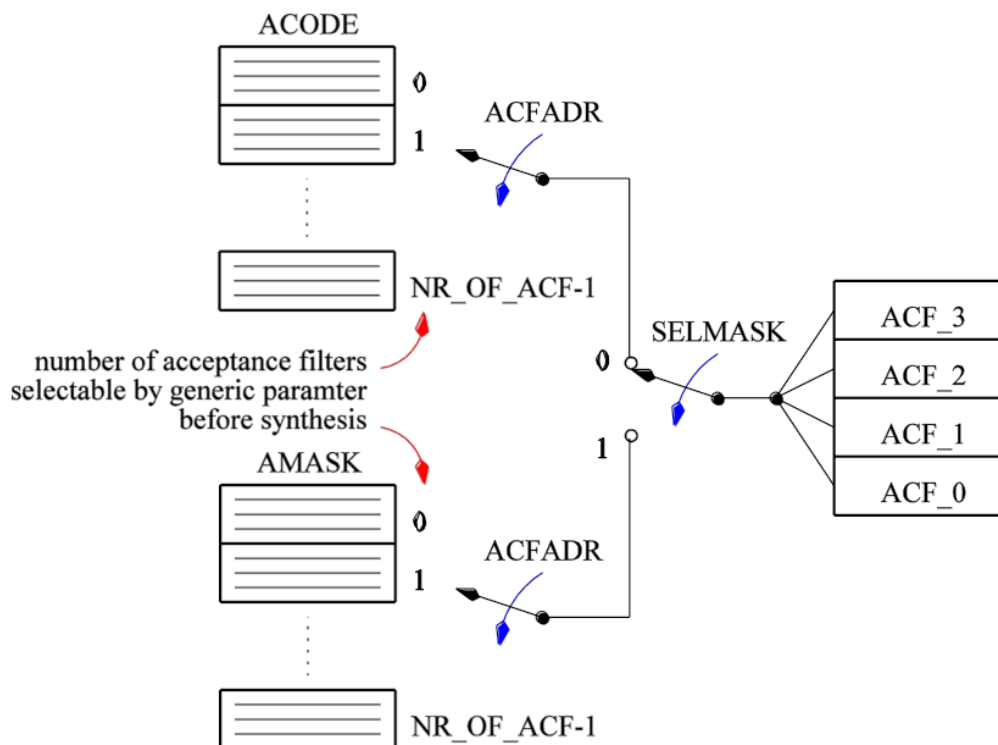


Figure 7-3 Access to the acceptance filters

0x0B8-BB ACF_x Acceptance CODE ACODE_x Register

Bits	Name	Access	Function
7:0	ACODE_0 ACODE_x	rw-0x00 rw-u	Acceptance CODE 1: ACC bit value to compare with ID bit of the received message 0: ACC bit value to compare with ID bit of the received message ACODE_x(10:0) will be used for extended frames. ACODE_x(28:0) will be used for extended frames. Only filter 0 is affected by the power-on reset. All other filters stay uninitialized.

0x0B8-BB ACF_x Acceptance CODE AMASK_x Register

Bits	Name	Access	Function
7:0	AMASK_0 AMASK_x	rw-0xFF rw-u	Acceptance Mask 1: acceptance check for these bits of receive identifier disabled 0: acceptance check for these bits of receive identifier enable AMASK_x(10:0) will be used for extended frames. AMASK_x(28:0) will be used for extended frames. Disabled bits result in accepting the message. Therefore, the default configuration after reset for filter 0 accepts all messages. Only filter 0 is affected by the power-on reset. All other filters stay uninitialized.

The AMASK_x includes additional bits in register ACF_3 which can be only accessed if SELMASK=1. These bits can be used to accept only either standard or extended frames with the selected ACODE/AMASK setting or to accept both frame types. Only acceptance filter 0 is affected by the power-on reset and it is configured to accept both frame types after power-up.

Table 7-2 Bits in Register ACF_3, if SELMASK=1

Bits	Name	Access	Function
6	AIDEE	rw-0 rw-u	Acceptance mask IDE bit check enable 1: acceptance filter accepts either standard or extended as defined by AIDE 0: acceptance filter accepts both standard or extended frames Only filter 0 is affected by the power-on reset. All other filters stay uninitialized.
5	AIDE	rw-0 rw-u	Acceptance mask IDE bit value If AIDEE=1 then: 1: acceptance filter accepts only extended frames 0: acceptance filter accepts only standard frames Only filter 0 is affected by the power-on reset. All other filters stay uninitialized.

0x0B6 ACF_EN_0 Acceptance Filter Enable Register

Bits	Name	Access	Function
7:0	AE_x	rw-0x01	Acceptance filter Enable 1: acceptance filter enable 0: acceptance filter disable Each acceptance filter (AMASK / ACODE) can be individually enabled or disabled. Only filter number 0 is enabled by default after hardware reset. Disabled filters reject a message. Only enabled filters can accept a message if the appropriate AMASK / ACODE configuration matches. To accept all messages one filter x has to be enabled by setting AE_x=1, AMASK_x=0xff and ACODE_x=0x00. This is the default configuration after hardware reset for filter x=0 while all other filters are disabled.

0x0B7 ACF_EN_1 Acceptance Filter Enable Register

Bits	Name	Access	Function
7:0	AE_x	rw-0x00	Acceptance filter Enable 1: acceptance filter enable 0: acceptance filter disable Each acceptance filter (AMASK / ACODE) can be individually enabled or disabled. Disabled filters reject a message. Only enabled filters can accept a message if the appropriate AMASK / ACODE configuration matches.

0x0BC-BD VER_0 – 1 Version Information Register

Bits	Name	Access	Function
15:0	VER_0 VER_1	r	Version of CAN-CTRL, given as decimal value. VER_1 holds the major version and VER_0 the minor version. Example: version 5x16N00S00 is represented by VER_1=5 and VER_0=16.

Table 7-3 Receive Buffer Registers RBUF – Standard Format (r-0)

Address	Bit position								Function
	7	6	5	4	3	2	1	0	
RBUF	ID(7:0)								Identifier
RBUF+1	-				ID(10:8)				Identifier
RBUF+2	-								Identifier
RBUF+3	-								Identifier
RBUF+4	IDE=0	RTR	-		DLC(3:0)			Control	
RBUF+5	-								-
RBUF+6									-
RBUF+7									-
RBUF+8	d1(7:0)								Data byte 1
RBUF+9	d2(7:0)								Data byte 2
.
RBUF+15	d8(7:0)								Data byte 8

Table 7-4 Receive Buffer Registers RBUF – Extended Format (r-0)

Address	Bit position								Function
	7	6	5	4	3	2	1	0	
RBUF	ID(7:0)								Identifier
RBUF+1	ID(15:8)								Identifier
RBUF+2	ID(23:16)								Identifier
RBUF+3	-			ID(28:24)					Identifier
RBUF+4	IDE=1	RTR	-		DLC(3:0)				Control
RBUF+5									-
RBUF+6									-
RBUF+7									-
RBUF+8	d1(7:0)								Data byte 1
RBUF+9	d2(7:0)								Data byte 2
.
RBUF+15	d8(7:0)								Data byte 8

The RBUF registers (0x00 to 0x47) point the message slot with the oldest received message in the RB.
All RBUF registers can be read in any order.
Please mind the gap inside the addressing range of RBUF at RBUF+7. This is for better address segment alignment for a wider host controller interface.

Table 7-5 Transmit Buffer Registers TBUF – Standard Format (rw-u)

Address	Bit position								Function
	7	6	5	4	3	2	1	0	
TBUF	ID(7:0)								Identifier
TBUF+1	-					ID(10:8)			Identifier
TBUF+2	-								Identifier
TBUF+3	-								Identifier
TBUF+4	IDE=0	RTR	-		DLC(3:0)				Control
TBUF+8	d1(7:0)								Data byte 1
TBUF+9	d2(7:0)								Data byte 2
.
TBUF+15	d8(7:0)								Data byte 8

Table 7-6 Transmit Buffer Registers TBUF – Extended Format (rw-u)

Address	Bit position								Function
	7	6	5	4	3	2	1	0	
TBUF	ID(7:0)								Identifier
TBUF+1	ID(15:8)								Identifier
TBUF+2	ID(23:16)								Identifier
TBUF+3	-			ID(28:24)					Identifier
TBUF+4	IDE=1	RTR	-		DLC(3:0)				Control
TBUF+8	d1(7:0)								Data byte 1
TBUF+9	d2(7:0)								Data byte 2
.
TBUF+15	d8(7:0)								Data byte 8

The TBUF registers (0x48 to 0x8f) point the next empty message slot in the STB if TBSEL=1 or to the PTB otherwise. All TBUF registers can be written in any order. For the STB it is necessary to set TSNEXT to mark a slot filled and to jump to the next message slot.

Please mind the gap inside the addressing range of TBUF from TBUF+5 to TBUF+7. This is for better address segment alignment. The memory cells in the gap can be read and written, but have no meaning for the CAN protocol.

TBUF is build using a true 32 bit wide memory and therefore a write access needs to be performed as 32 bit write.

Both RBUF and TBUF include some frame-individual control bits (Table 7-7). For RBUF these bits signal the status of the appropriate CAN control field bits of the received CAN frame while for TBUF these bits select the appropriate CAN control field bit for the frame that has to be transmitted.

In contrast to RTS, which is stored for every received frame, TTS is stored only for the last transmitted frame. TTS is not related to the actual selected TBUF slot.

Table 7-7 Control bits in RBUF and TBUF

Bit	Description
IDE	Identifier Extension
	0 – Standard Format: ID(10:0)
	1 – Extended Format: ID(28:0)
RTR	Remote Transmission Request
	0 – data frame
	1 – remote frame
	Only CAN 2.0 frames can be remote frames.

The Data Length Code (DLC) in RBUF and TBUF defines the length of the payload – the number of payload bytes in a frame.

Remote frames (only for CAN 2.0 frames where EDL=0) are always transmitted with 0 payload bytes, but the content of the DLC is transmitted in the frame header. Therefore, it is possible to code some information into the DLC bits for remote frames. But then care needs to be taken if different CAN nodes are allowed to transmit a remote frame with the same ID. In this case, all transmitters need to use the same DLC because otherwise this would result in an unresolvable collision.

Table 7-8 Definition of the DLC (according to the CAN 2.0 specification)

DLC (binary)	Frame Type	Payload in Bytes
0000 to 1000	CAN 2.0	0 to 8
1001 to 1111	CAN 2.0	8

The TBUF registers are readable and writable. Therefore, a host controller may use TBUF to successively prepare a message bit-by-bit if necessary.

7.5 General operation

7.5.1 The bus off state

The “bus off” state is signaled using the status bit BUSOFF in register CFG_STAT. A CAN node enters the “bus off” state automatically if its transmit error counter becomes >255. Then it will not take part in further communications until it returns into the error active state again. A CAN node returns to error active state if it is reset by a power-on reset or if it receives 128 sequences of 11 recessive bits (recovery sequences).

In the “bus off” state, RECNT is used to count the recovery sequences while TECNT stays unchanged. Please note that while entering bus off state TECNT rolls over and therefore may hold a small value. Therefore, it is recommended to use TECNT before the node enters bus off state and status bit BUSOFF afterwards.

If the node recovers from “bus off” state, then RECNT and TECNT are automatically reset to 0. When BUSOFF gets set, then RESET is automatically set. Therefore, both RECNT and TECNT are not affected by the software reset.

7.5.2 Acceptance filters

To reduce the load of received frames for the host controller, the core uses acceptance filters. The CAN- CTRL core checks the message identifier during acceptance filtering. Therefore, the length of each acceptance filter is 29 bits.

If a message passes one of the filters, then it will be accepted. If accepted, the message will be stored into the RB and finally RIF is set if RIE is enabled. If the message is not accepted, RIF is not set and the RB FIFO pointer is not increased. Messages that are not accepted will be discarded and overwritten by the next message. No stored valid message will be overwritten by any not accepted message.

Independently of the result of acceptance filtering, the CAN-CTRL core checks every message on the bus and sends an acknowledge or an error frame to the bus.

The acceptance mask defines which bits shall be compared while the acceptance code defines the appropriate values. Setting mask bits to 0 enables the comparison of the selected acceptance code bits with the corresponding message identifier bits. Mask bits that are set to 1 are disabled for the acceptance check and this results in accepting the message.

The identifier bits will be compared with the corresponding acceptance code bits ACODE as follows:

- Standard: ID (10:0) with ACODE (10:0).
- Extended: ID (28:0) with ACODE (28:0).

Example: If AMASK_x (0) = 0 and all other AMASK_x bits are 1, then the value of the last ID bit has to be equal to ACODE (0) for an accepted message. All other ID bits are ignored by the filter.

Note: Disabling a filter by setting AE_x=0 blocks messages. In contrast to this disabling a mask bit in AMASK_x disables the check for this bit which results in accepting messages.

The definitions of AMASK and ACODE alone do not distinguish between standard or extended frames. If bit AIDEE=1 then the value of AIDE defines with frame type is accepted. Otherwise if AIDE=0 both types are accepted.

After power-on reset, the CAN-CTRL core is configured to accept all messages. (Filter 0 is enabled by AE_0=1, all bits in AMASK_0 are set to 1 and AIDEE=0. All other filters are disabled. Filter 0 is the only filter that has defined reset values for AMASK/ ACODE while all other filters have undefined reset values.)

7.5.3 Message reception

The received data will be stored in the RB as shown in Figure 7-4. The RB is configurable by a pre-synthesis parameter and has FIFO-like behavior. Every received message that is valid and accepted sets RIF=1 if RIE is enabled. RSTAT is set depending of the fill state. When the number of filled buffers is equal to the programmable value AFWL, then RAFIF is set if RAFIE is enabled. In case, when all buffers are full, the RFIF is set if RFIE is enabled.

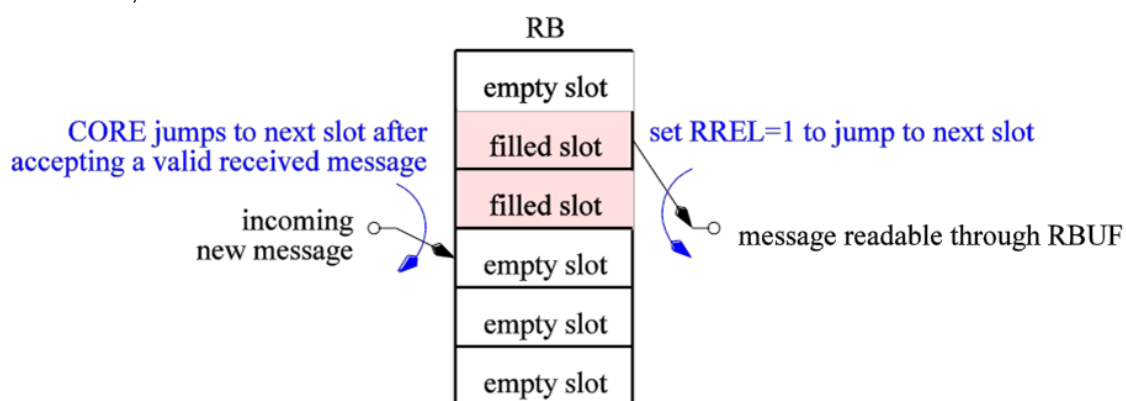


Figure 7-4 Schematic of the FIFO-like RB (example with 6 slots)

The RB always maps the message slot containing the oldest message to the RBUF registers.

The maximum payload length for CAN 2.0 messages is 8. The individual length of each message is defined by the DLC. Because of this, the RB provides slots for each message and the host controller is required to set RREL to jump to the next RB slot. All RBUF bytes of the actual slot can be read in any order.

If the RB is full, the next incoming message will be stored temporarily until it passes for valid (6th EOF bit). Then if ROM=0 the oldest message will be overwritten by the newest or if ROM=1 the newest message will be discarded. In both cases, ROIF is set if ROIE is enabled. If the host controller reads the oldest message and sets RREL before a new incoming message becomes valid, then no message will be lost.

7.5.4 Handling message receptions

Without acceptance filtering, the CAN-CTRL core would signal the reception of every frame and the host would be required to decide if it was addressed. This would result in quite a big load on the host controller.

It is possible to disable interrupts and use the acceptance filters to reduce the load for the host controller. For a basic operation RIF is set to 1 if RIE is enabled and the CAN-CTRL core has received a valid message. To reduce the number of reception interrupts, it is possible to use RAIE / RAIF (RB

Almost full Interrupt) or RFIE/RFIF (RB Full Interrupt) instead of RIE/RIF (Reception Interrupt). The “almost full limit” is programmable using AFWL.

The RB contains a number of RB slots, which is selectable before synthesis using a generic parameter.

Reading the RB shall be done as follows:

1. Read the oldest message from the RB FIFO using the RBUF registers.
2. Release the RB slot with RREL=1. This selects the next message (the next FIFO slot). RBUF will be updated automatically.
3. Repeat these actions until RSTAT signals an empty RB.

If the RB FIFO is full and a new received message is recognized as valid (6th EOF bit) then one message will be lost (see bit ROM). Before this event, no message is lost. This should give enough time for the host controller to read at least one frame from the RB after the RB FIFO has been filled and the selected interrupt has occurred. To enable this behavior, the RB includes one more (hidden) slot than specified by the synthesis parameter RBUF_SLOTS. This hidden slot is used to receive a message, validate it and check it if it matches the acceptance filters before an overflow occurs.

7.5.5 Message transmission

Before starting any transmission, at least one of the transmit buffers (PTB or STB) has to be loaded with a message. TPE signals if the PTB is locked and TSSTAT signals the fill state of the STB. The TBUF registers provide access to both the PTB as well as to the STB. Below is the recommended programming flow:

1. Set TBSEL to the desired value to select either the PTB or the STB.
2. Write the frame to the TBUF registers.
3. For the STB set TSNEXT=1 to finish loading of this STB slot.

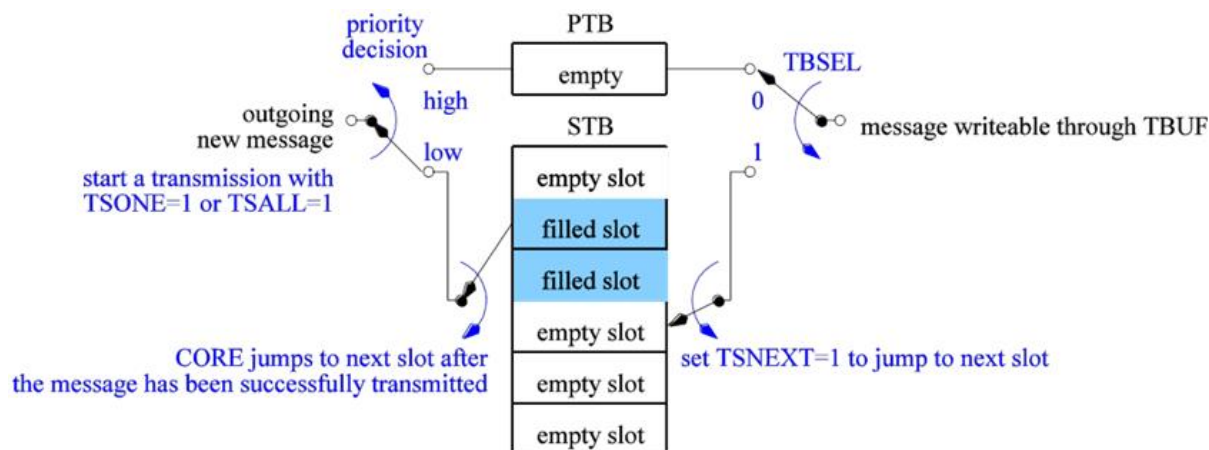


Figure 7-5 Schematic of PTB and STB in FIFO mode (empty PTB and 6 STB slots)

The maximum payload length for CAN 2.0 messages is 8 bytes. The individual length of each message is defined by the DLC. For remote frames (bit RTR), the DLC becomes meaningless, because remote frames always have a data length of 0 bytes. The host controller is required to set TSNEXT to jump to the next STB slot. All TBUF bytes can be written in any order.

Setting TSNEXT=1 is meaningless if TBSEL=0 selects the PTB. In this case TSNEXT is automatically cleared and does no harm.

Bit TPE should be set to start a transmission when using the PTB. To use the STB, TSONE has to be set to start a transmission of a single message or TSALL to transmit all messages.

The PTB has always a higher priority than the STB. If both transmit buffers have the order to transmit, the PTB message will be always sent first regardless of the frame identifiers. If a transmission from the STB is already active, it will be completed before the message from the PTB is sent at the next possible transmit position (the next frame slot). After the PTB transmission is completed or aborted, the CAN-CTRL core returns to process other pending messages from the STB.

When the transmission is completed, the following transmission interrupts are set:

- For the PTB, TPIF is set if TPIE is enabled.
- For the STB using TSONE, TSIF is set if one message has been completed and TSIE is enabled.
- For the STB using TSALL, TSIF is set if all messages have been completed and if TSIE is enabled. In other words, TSIE is set if the STB is empty. Therefore, if the host controller writes an additional message to the STB after a TSALL transmission has been started then the additional message will be also transmitted before TSIF will be set.

It is meaningless to set TSONE or TSALL while the STB is empty. In such a case TSONE respectively TSALL will be reset automatically. No interrupt flag will be set and no frame will be transmitted.

7.5.6 Message transmission abort

If the situation arises, where a message in a transmit buffer cannot be sent due to its low priority, this would block the buffer for a long time. In order to avoid this, the host controller can withdraw the transmission request by setting TPA or TSA respectively, if the transmission has not yet been started.

Both TPA and TSA source a single interrupt flag: AIF. The CAN protocol machine executes an abort only if it does not transmit anything to the CAN bus. Therefore, the following is valid:

- There is no abort during bus arbitration.
 - If the node loses arbitration, the abort will be executed afterwards.
 - If the node wins arbitration, the frame will be transmitted.
- There is no abort while a frame is transmitted.
 - If a frame is transmitted successfully, then a successful transmission is signaled to the host controller. In this case no abort is signaled. This is done by the appropriate interrupt and status bits.
 - After an unsuccessful transmission where the CAN node does not receive an acknowledge, the error counter is incremented and the abort will be executed.
 - If there is at least one frame left in the STB, while the host has commanded all frames to be transmitted (TSALL=1), then both the completed frame as well as the abort is signaled to the host.

Because of these facts aborting a transmission may take some time depending on the CAN communication speed and frame length. If an abort is executed, this results in the following actions:

- TPA releases the PTB which results in TPE=0.
The frame data is still stored in the PTB after releasing the PTB.
- TSA releases one single message slot or all message slots of the STB. This depends on whether TSONE or TSALL was used to start the transmission. TSSTAT will be updated accordingly. Releasing a frame in the STB results in discarding the frame data because the host cannot access it.

Setting both TPA and TSA simultaneously is not recommended. If a host controller decides to do it anyway, then AIF will be set and both transmissions from PTB and STB will be aborted if possible. As

already stated if one transmission will be completed before the abort can be executed, this will result in signaling a successful transmission. Therefore, the following interrupt flags may be set if enabled:

- AIF (once for both PTB and STB transmission abort)
- TPIF + AIF
- TSIF + AIF
- TPIF + TSIF (very seldom, will only happen if the host does not handle TPIF immediately)
- TPIF + TSIF + AIF (very seldom, will only happen if the host does not handle TPIF and TSIF immediately)

To clear the entire STB, both TSALL and TSA need to be set. In order to detect if a message cannot be sent for a long time because it loses arbitration, the host may use the ALIF/ALIE.

7.5.7 A full STB

After writing a message to the STB, TSNEXT=1 marks a buffer slot filled and jumps to the next free message slot. TSNEXT is automatically reset to 0 by the CAN-CTRL core after this operation.

If the last message slot has been filled and therefore all message slots are occupied, then TSNEXT stays set until a new message slot becomes free. While TSNEXT=1, then writing to TBUF is blocked by the CAN-CTRL core.

When a slot becomes free, then the CAN-CTRL core automatically resets TSNEXT to 0. A slot becomes free if a frame from the STB is transmitted successfully or if the host requests an abort (TSA=1). If a TSALL transmission is aborted, then TSNEXT is also reset, but additionally the complete STB is marked as empty.

7.5.8 Extended status and error report

During CAN bus communication transmission errors may occur. The following features support detection and analysis of them.

7.5.8.1 Programmable error warning limit

Errors during reception/ transmission are counted by RECNT and TECNT. A programmable error warning limit EWL, located in register LIMIT, can be used by the host controller for flexible reaction on those events. The limit values can be chosen in steps of 8 errors from 8 to 128:

Limit of count of errors = (EWL+1) *8.

The interrupt EIF will be set if enabled by EIE under the following conditions:

- The border of the error warning limit has been crossed in either direction by RECNT or TECNT
- The BUSOFF bit has been changed in either direction.

7.5.8.2 Arbitration Lost Capture (ALC)

The core is able to detect the exact bit position in the Arbitration Field where the arbitration has been lost. This event can be signaled by the ALIF interrupt if it is enabled. The value of ALC stays unchanged if the node is able to win the arbitration. Then ALC holds the old value of the last loss of arbitration.

The value of ALC is defined as follows: A frame starts with the SOF bit and then the first bit of the ID is transmitted. This first ID bit has ALC value 0, the second ID bit ALC value 1 and so on.

Arbitration is only allowed in the arbitration field. Therefore, the maximum value of ALC is 31 which is the RTR bit in extended frames.

Additional hint: If a standard remote frame arbitrates with an extended frame, then the extended frame loses arbitration at the IDE bit and ALC will be 12. The node transmitting the standard remote frame will not notice that an arbitration has been taken place, because this node has won.

It is impossible to get an arbitration loss outside of the arbitration field. Such an event is a bit error.

7.5.8.3 Kind of Error (KOER)

The core recognizes errors on the CAN bus and stores the last error event in the KOER bits. A CAN bus error can be signaled by the BEIF interrupt if it is enabled. Every new error event overwrites the previous stored value of KOER. Therefore, the host controller has to react quickly on the error event.

7.5.9 Extended features

7.5.9.1 Single shot transmission

Sometimes an automatic re-transmission is not desired. Therefore, the order to transmit a message only once can be set separately for the transmit buffers PTB by the bit TPSS and for the transmit buffer STB by the bit TSSS. In this case no re-transmission will be performed in the event of an error or arbitration lost if the selected transmission is active.

In the case of an immediate successful transmission, there is no difference to normal transmission. But in the case of an unsuccessful transmission, the following will happen:

- TPIF gets set if enabled, the appropriate transmit buffer slot gets cleared.
- In case of an error, KOER and the error counters get updated. BEIF gets set if BEIE is enabled and the other error interrupt flags will act accordingly.
- In case of a lost arbitration, ALIF gets set if ALIE is enable.

Therefore, for single shot transmission TPIF alone does not indicate if the frame has been successfully transmitted. Therefore, single shot transmission should only be used together with BEIF and ALIF.

If single shot transmission is used with TSALL and there is more than one frame in the STB then for each frame a single shot transmission is done. Regardless if any of the frames is not successfully transmitted (e.g. because of an ACK error), the CAN-CTRL advances to the next frame and stops if the STB is empty. Therefore, in this scenario only the error counters indicate what has happened. This can be quite complex to evaluate because if one of two frames got errors, the host cannot detect which one was the successful one.

If the bus is occupied by another frame, if a single shot transmission is started, then CAN-CTRL waits till the bus is idle and then tries to transmit the single shot frame.

7.5.9.2 Listen Only Mode (LOM)

LOM provides the ability of monitoring the CAN bus without any influence to the bus.

Another application is the automatic bit rate detection where the host controller tries different timing settings until no errors occur.

Errors will be monitored (KOER, BEIF) in LOM.

- In LOM, the core is not able to write dominant bits onto the bus (no active error flags or overload flags, no acknowledge). This is done using the following rules.
- In LOM, the protocol machine acts as if in error passive mode where only recessive error flags are generated. Only the protocol machine acts as if in error passive mode. All other components including the status registers are not touched.
- In LOM, the protocol machine does not generate a dominant ACK.

- The error counters stay unchanged regardless of any error condition.

Important facts regarding ACKs for LOM:

- If a frame is transmitted by a node, then an ACK visible at the bus will be only generated if at least one additional node is attached to the bus that is not in LOM. Then there will be no error and all nodes (also those in LOM) will receive the frame.
- If there is an active or passive error flag after an ACK error, then the node in LOM is able to detect this as ACK error.

Activation of LOM should not be done while a transmission is active. The host controller has to take care of this. There is not additional protection from CAN-CTRL.

If LOM is enabled, then no transmission can be started.

7.5.9.3 Bus connection test

To check if a node is connected to the bus, the following steps shall be done:

- Transmit a frame. If the node is connected to the bus, then its TX bits are visible on its RX input.
- If there are other nodes connected to the CAN bus, then a successful transmission (including an acknowledge from the other nodes) is expected. No error will be signaled.
- If the node is the only node that is connected to the CAN bus (but the connection between the bus, the transceiver and the CAN-CTRL core is fine), then the first regular error situation occurs in the ACK slot because of no acknowledge from other nodes. Then a BEIF error interrupt will be generated if enabled and KOER= "100" (ACK error).
- If the connection to the transceiver or the bus is broken, then immediately after the SOF bit the BEIF error interrupt will be set and KOER= "001" (BIT error).

7.5.9.4 Loop Back Mode (LBMI and LBME)

CAN-CTRL supports two Loop Back Modes: internal (LBMI) and external (LBME). Both modes result in reception of the own transmitted frame which can be useful for self-tests.

In LBMI, CAN-CTRL is disconnected from the CAN bus and the txd output is set to recessive. The output data stream is internally fed back to the input. In LBMI, the node generates a self-ACK to avoid an ACK error.

In LBME, CAN-CTRL stays connected to the transceiver and a transmitted frame will be visible on the bus. With the help of the transceiver, CAN-CTRL receives its own frame. In LBME, the node does not generate a self-ACK. Therefore, in LBME, there are two possible results upon a frame transmission:

1. Another node receives the frame too and generates an ACK. This will result in a successful transmission and reception.
2. No other node is connected to the bus and this results in an ACK error. To avoid retransmissions and incrementing the error counters, it is recommended to use TPSS or TSSS if it is unknown if other nodes are connected.

In Loop Back Mode, the core receives its own message, stores it in the RBUF and sets the appropriate receive and transmit interrupt flags if enabled.

LBMI can be useful for chip-internal and software tests while LBME can test the transceiver and the connections to it.

Activation of both Loop Back Modes should not be done while a transmission is active. The host controller has to take care of this. There is not additional protection from CAN-CTRL.

If the node is connected to a CAN bus, then switching back from LBMI to normal operation must not be done by simply setting LBMI to 0, because then it may be that this occurs just at the moment while another CAN node is transmitting. In this case, switching back to normal operation shall be done by setting bit RESET to 1. This automatically clears LBMI to 0. Finally RESET can be disabled and the core returns back to normal operation. In contrast to this, LBME can be disabled every time.

7.5.9.5 Transceiver standby mode

Using the register bit STBY, the output signal standby is driven. It can be used to activate a standby mode for a transceiver.

Once standby is activated, no transmission is possible and therefore TPE, TSONE and TSALL cannot be set. On the other hand, CAN-CTRL does not allow STBY to be set if a transmission is already active (TPE, TSONE or TSALL is set).

If STBY is set, the transceiver enters a low-power mode. In this mode, it is unable to receive a frame at full speed but monitors the CAN bus for a dominant state. If the dominant state is active for a time which is defined in the transceivers data sheet, the transceiver will pull the rxd signal low. If rxd is low, CAN-CTRL automatically clears STBY to 0 which disables standby mode for the transceiver. This is done silently without an interrupt to the host controller.

Switching from standby mode to active mode takes some time for the transceiver and therefore the initial wakeup frame cannot be received successfully. Therefore, the node recently being in standby will not respond with an ACK. If no CAN node at the bus responds to the wakeup frame with an ACK, then this results in an ACK error for the transmitter of the wakeup frame. Then the transmitter will automatically repeat the frame. During the repetition, the transceiver will be back in active mode and CAN-CTRL will receive the frame and will respond with an ACK.

In summary, one node transmits a frame for wakeup. If all others nodes are in standby mode, then the transmitter gets an ACK error and will automatically repeat the frame. During the repetition, the nodes are back in active mode and will respond with an ACK.

STBY is not affected by bit RESET.

7.5.9.6 Error counter reset

According to the CAN standard RECNT counts receive errors and TECNT counts transmit errors. After too many transmit errors, a CAN node has to go to bus off state. This will activate the bit RESET. Deactivating the bit, RESET does not modify the errors counters or bus off state. The CAN specification defines rules how to disable bus off state and to decrease the error counters. A good node will recover from all of this automatically if only a temporary error has caused the problems. The classic CAN 2.0B specification requires this automatic behavior without host controller interaction to avoid the babbling idiot problem.

Writing a 1 to the bit BUSOFF resets the error counters and therefore forces the node to leave the bus off state. This is done without setting EIF.

7.5.10 Software reset

If bit RESET in CFG_STAT is set to 1, then the software reset is active. Several components are forced to a reset state if RESET=1 but not all components are touched by RESET. Some components are only sensitive to a hardware reset. The reset values of all bits are always the same for software and hardware reset.

Table 7-9 Software reset

Register	RESET	Comment
RBUF	(Yes)	All RB slots are marked as empty. RBUF contains unknown data.
TBUF	(Yes)	All STB slots are marked as empty. Because of TBSEL TBUF points to the PTB.
TTS	No	-
LBME	Yes	-
LBMI	Yes	-
TPSS	Yes	-
TSSS	Yes	-
RACTIVE	Yes	Reception is immediately cancelled even if a reception is active. No ACK will be generated.
TACTIVE	Yes	All transmissions are immediately terminated with RESET. If a transmission is active this will result in an erogenous frame. Other nodes will generate error frames.
BUSOFF	No	-
TBSEL	Yes	TBUF fixed to point to PTB.
LOM	Yes	-
STBY	No	-
TPE	Yes	-
TPA	Yes	-
TSONE	Yes	-
TSALL	Yes	-
TSA	Yes	-
TSNEXT	Yes	-
TSMODE	No	-
TSSTAT	Yes	All STB slots are marked as empty.
ROM	No	-
ROV	Yes	All RB slots are marked as empty.
RREL	Yes	-
RSTAT	Yes	All RB slots are marked as empty.
RIE	No	-
ROIE	No	-
RFIE	No	-
RAFIE	No	-
TPIE	No	-
TSIE	No	-
EIE	No	-

Register	RESET	Comment
TSFF	Yes	All STB slots are marked as empty.
RIF	Yes	-
ROIF	Yes	-
RFIF	Yes	-
RAFIF	Yes	-
TPIF	Yes	-
TSIF	Yes	-
EIF	No	-
AIF	Yes	-
EWARN	No	-
EPASS	No	-
EPIE	No	-
EPIF	Yes	-
ALIE	No	-
ALIF	Yes	-
BEIE	No	-
BEIF	Yes	-
S_Seg_1	No	Register is writeable if RESET=1 and write-locked if 0.
S_Seg_2	No	Register is writeable if RESET=1 and write-locked if 0.
S_SJW	No	Register is writeable if RESET=1 and write-locked if 0.
S_PRESC	No	Register is writeable if RESET=1 and write-locked if 0.
AFWL	No	-
EWL	Yes	-
KOER	Yes	-
ALC	Yes	-
RECN	No	-
TECN	No	-
SELMASK	No	-
ACFADR	No	-
AE_x	No	-
ACODE_x	No	Register is writeable if RESET=1 and write-locked if 0.
AMASK_x	No	Register is writeable if RESET=1 and write-locked if 0.

7.5.1 CAN bit time

CAN 2.0B defines data bit rates up to 1Mbit/s. For real life systems the data rates are limited by the used transceiver and the achievable clock frequency for the CAN-CTRL core which depends on the used target cell library.

The CAN-CTRL core can be programmed to arbitrarily chosen data rates only limited by the range of the bit settings in the appropriate bit timing and prescaler registers.

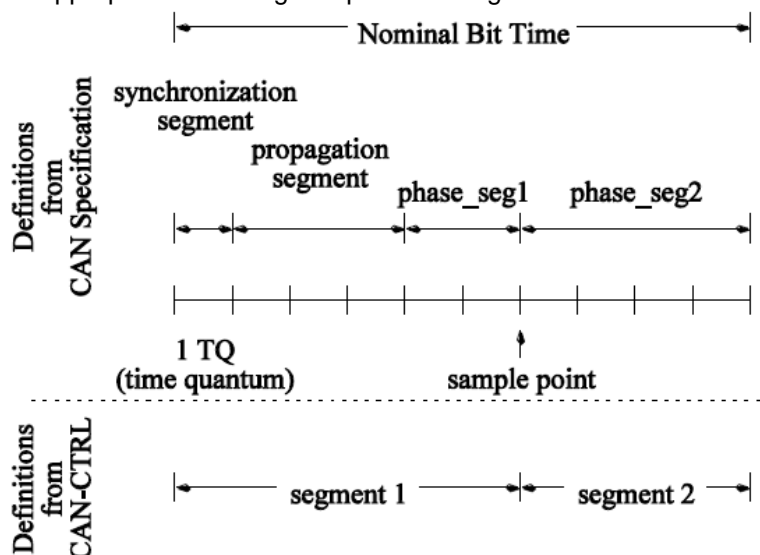


Figure 7-6 CAN Bit Timing Specifications

The CAN bit time BT consists of several segments as shown in Figure 7-6. Each segment consists of a number of time quanta units n_{TQ} . The duration of a time quanta TQ is:

$$TQ = n_{prescaler} / f_{CLOCK}$$

$$BT = n_{prescaler} * n_{TQ} / f_{CLOCK} = t_{Seg_1} + t_{Seg_2}$$

The CAN specification requires several relationships between the segment lengths (Table 7-10) which results in relationships between t_{Seg_1} , t_{Seg_2} and the maximum synchronization jump width t_{SJW} . Please note that Table 7-10 lists the minimum configuration ranges defined by the CAN specification.

Table 7-10 CAN Timing Segments

Segment	Description
SYNC_SEG	Synchronization Segment = 1 TQ
PROP_SEG	Propagation Segment [1...8] TQ CAN 2.0 bit rate CAN FD not enabled [1...48] TQ CAN 2.0 bit rate CAN FD enabled
PHASE_SEG1	Phase Buffer Segment 1 [1...8] TQ CAN 2.0 bit rate CAN FD not enabled [1...16] TQ CAN 2.0 bit rate CAN FD enabled
PHASE_SEG2	Phase Buffer Segment 2 [2...8] TQ CAN 2.0 bit rate CAN FD not enabled [2...16] TQ CAN 2.0 bit rate CAN FD enabled
SJW	Synchronization Jump Width [1...4] TQ CAN 2.0 bit rate CAN FD not enabled [1...16] TQ CAN 2.0 bit rate CAN FD enabled

As can be seen in Figure 7-10 the CAN-CTRL core collects SYNC_SEG, PROP_SEG and PHASE_SEG1 into one group and the length of the group is configurable with t_{Seg_1} . Table 7-11 lists the available configuration ranges. Please note that the CAN-CTRL core does not check if all rules are met and offers a wider configuration range than defined by the CAN specification.

Table 7-11 CAN-CTRL Timing Settings

Setting	Requirements
t_{Seg_1}	[2...65] TQ CAN 2.0 bit rate (slow)
t_{Seg_2}	[1...16] TQ $t_{Seg_1} \geq t_{Seg_2} + 2$ CAN 2.0 bit rate (slow)
t_{SJW}	[1...16] TQ $t_{Seg_2} \geq t_{SJW}$ CAN 2.0 bit rate (slow)

For CAN 2.0 bit rate nominal bit rate the register settings S_Seg_1, S_Seg_2, S_SJW and S_PRESC define the appropriate segment lengths.

$$t_{Seg_1} = (S_Seg_1 + 2) * TQ$$

$$t_{Seg_2} = (S_Seg_2 + 1) * TQ$$

$$t_{SJW} = (S_SJW + 1) * TQ$$

$$n_{prescaler} = S_PRESC + 1$$

The following steps need to be carried for the configuration of the CAN-CTRL core:

1. Set bit RESET=1.
2. Set registers S_Seg_1 and S_Seg_2:
In the example the data rate on the bus 1M baud and the system clock is 48 MHz.
The values of n_{TQ} and $n_{prescaler}$ have to be selected to fit BT .
In this example $n_{prescaler} = 2$ and $n_{TQ} = 24$ are chosen which results in a perfect match:
 $BT = 24TQ$.
the time segment definitions $t_{Seg_1} = 18TQ$; $t_{Seg_2} = 6TQ$ can be chosen as suitable values which finally results in the register settings S_Seg_1=16 and S_Seg_2=5.
3. Load the acceptance code and mask registers (optional).
4. Set register S_SJW:
With $t_{Seg_2} \geq t_{SJW}$ one is free to chose $t_{SJW} = 4$ which finally results in 3.
5. Load the clock prescaler register S_PRESC: $n_{prescaler} = PRESC + 1$ results in S_PRESC=1.
6. Set bit RESET=0.
The given order is not mandatory. It is merely necessary to set bit RESET=1 at the beginning, as it is otherwise not possible to load the bit timing, ACODE and AMASK registers. RESET=0 is required upon completion of the configuration. The controller then waits for 8 recessive bits (frame end) and resumes its normal operation.
7. Continue with configuration of interrupts other configuration bits and execute commands.

There are some sample tables for the bit timing settings that should apply to all nodes in a CAN network:

Table 7-12 Sample settings for 48MHz can_clk

Bit Rate[Mbit/s]	SP[%]	Prescaler	Bit Time[TQ]	Seg1 [TQ]	Seg2 [TQ]	SJW [TQ]
1	75	2	24	18	6	4
0.8	75	3	20	15	5	4
0.5	75	4	24	18	6	4
0.25	75	8	24	18	6	4
0.125	75	16	24	18	6	4

Bit Rate[Mbit/s]	SP[%]	Prescaler	Bit Time[TQ]	Seg1 [TQ]	Seg2 [TQ]	SJW [TQ]
0.1	75	20	24	18	6	4

Table 7-13 Sample settings for 8MHz can_clk

Bit Rate[Mbit/s]	SP[%]	Prescaler	Bit Time[TQ]	Seg1 [TQ]	Seg2 [TQ]	SJW [TQ]
1	75	1	8	6	2	2
0.8	80	1	10	8	2	2
0.5	75	1	16	12	4	4
0.25	75	2	16	12	4	4
0.125	75	4	16	12	4	4
0.1	75	4	20	15	5	4

8 LIN

8.1 Introduction

The controller interfaces the local interconnect network (LINCORE), compliant with LIN Protocol Specification Revisions 1.3, 2.0, 2.1, and 2.2, which works in master and slave modes, and judges errors automatically.

8.2 Feature list

- LIN Protocol Specification Revisions 1.3, 2.0, 2.1, and 2.2.
- Classic checksum or enhanced checksum selectable.
- Autonomous header and response handling.
- Up to 8 data bytes in response field.
- Detects dominant level to produce wake-up signal.
- Resynchronizes clock in slave mode.
- 16 identifier filters in slave mode.
- Fractional baud rate generator.
- 3 operating modes for power saving and configuration registers lock:
 - Initialization
 - Normal
 - Sleep
- 2 test mode
 - Loop back
 - Self-test
- Error detection
 - Bit error
 - Checksum error
 - Frame timeout error
 - Physical bus error
 - Framing error
 - Overrun error

8.3 Block diagram

The structure of the LINCORE is as follows.

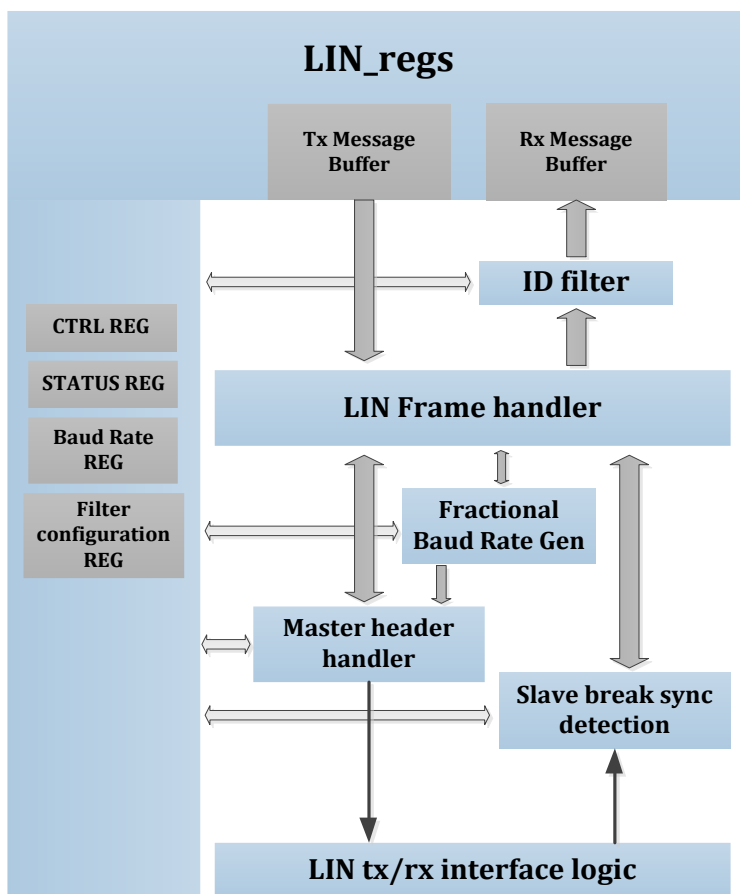


Figure 8-1 LINCORE block diagram

8.4 Application note

8.4.1 Operating modes

LINCORE has three main operating modes: Initialization, Normal and Sleep.

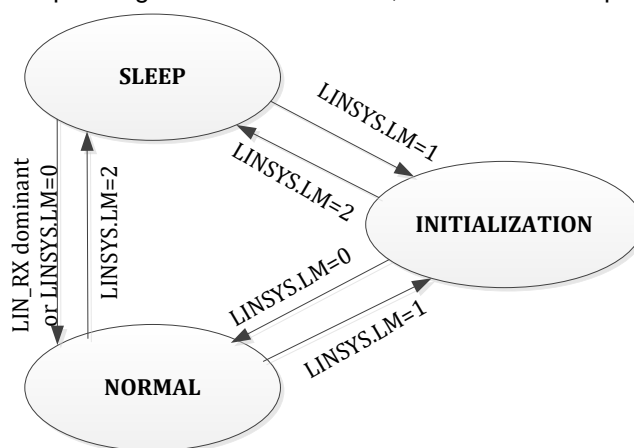


Figure 8-2 LINCORE operating modes

8.4.1.1 Initialization mode

The software sets the LM bits to 0x01 in the LINSYS to enter initialization mode, sets the LM bits to other value, either 0 or 2, to exit this mode.

In initialization mode, the software can set up work mode (master or slave mode), baud rate and filter function.

8.4.1.2 Normal mode

After initialization, the software enters to normal mode by setting the LM bits to 0 in LINSYS.

8.4.1.3 Sleep mode

The LINCORE enters the sleep mode to reduce power consumption by setting the LM bits to 2.

If auto-wake mode is enabled, software programs the LM bit or LINCORE detects the dominant level, and then LINCORE can exit sleep mode.

8.4.2 Test modes

The LINCORE has two test mode: loop back mode and self-test mode. They can be set by the LBM and STM bits in the LINC1 in Initialization mode.

8.4.2.1 Loop back mode

In this mode, the LINCORE performs an internal feedback from its Tx output to its Rx input. The RX input pin is disconnected from the LINRX pin. The LINCORE can receive messages that itself sent in master mode, and the bus can receive messages.

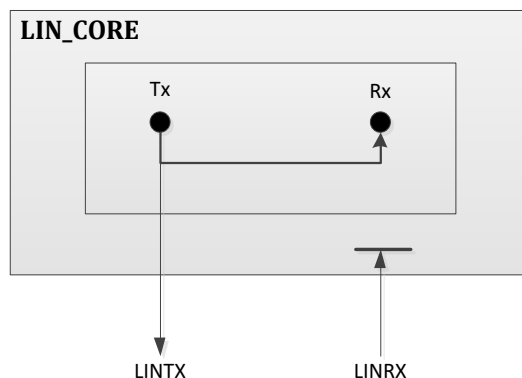


Figure 8-3 Loop back mode

8.4.2.2 Self-test mode

In this mode, the LINCORE performs an internal feedback from its Tx output to its Rx input. The RX input pin is disconnected from the LINRX pin, and the TX output pin is disconnected from the LINRTX pin. The LINCORE can receive messages that itself sent in master mode, and the bus cannot receive messages.

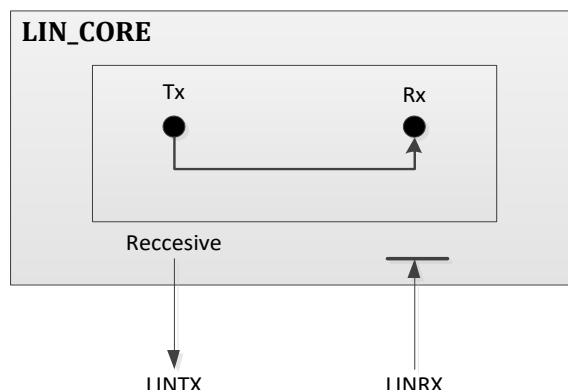


Figure 8-4 Self-test mode

8.4.3 Baud rate generation

The LINCORE can work with different baud rate, using the Mantissa and Fraction registers.

$$\text{Baud Rate} = \frac{F_{bus}}{(16 \times DIV)}$$

DIV is calculated by the previous formula, for example:

F_{bus} = 48MHz, if set baud rate = 19200

DIV = 156.25

Mantissa = 156

Fraction = $0.25 \times 16 = 4$

Actual baud rate = 19200, error = 0%

8.4.4 Error detection

LINCORE is able to detect and handle LIN communication errors. A code stored in the LIN error status register (LINESTS) signals the errors to the software.

- **Bit error:** during transmission, the value read back from the bus differs from the transmitted value.
- **Header error:** an error occurred during header reception in slave mode only (Break Delimiter error, Inconsistent Synch Field, Parity ID Error).
- **Framing error:** a dominant state has been sampled on the stop bit of the currently received character (sync field, identifier field or data field).
- **Checksum error:** the computed checksum does not match the received one.
- **Time out error:** header timeout is checked start from the end of sync byte stop bit to the end of the parity id stop bit, the minimum value is 11, the maximum value is 32. Check the response space timeout in the response time space with a maximum timeout value of 36. Response timeout is checked in the time space including response space, data field and the checksum field, the timeout value is set according to the data field length.

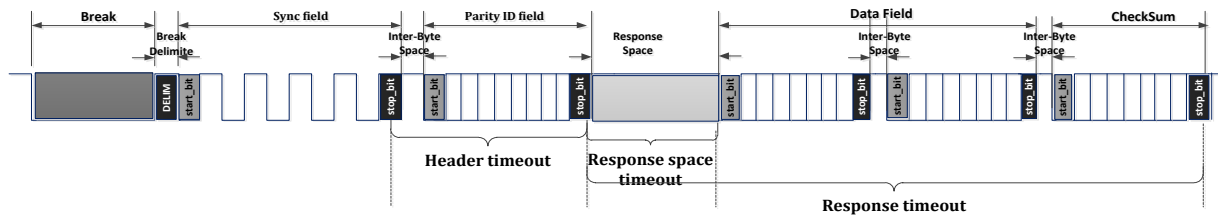


Figure 8-5 The structure of a frame

8.4.5 Interrupt table

Table 8-1 LIN interrupt table

Interrupt event	Flag bit	Enable control bit
Header Transmitted/Received interrupt	HTRF	HTRIE
Data Transmitted interrupt	DTF	DTIE
Data Received interrupt	DRF	DRIE
Wake-up interrupt	WUF	WUIE
Buffer Overrun interrupt	BOF	BOIE
Framing Error interrupt	FEF	FEIE
Header Error interrupt	IPEF, BDEF, SFEF	HEIE
Checksum Error interrupt	CEF	CEIE
Bit Error interrupt	BEF	BEIE
Time out interrupt	TOEF	TOIE
Long Zero interrupt	LZEF	LZIE

8.4.6 Master mode

When the LINCR1[SM] bit is cleared, the master mode is selected. In the master mode, the application handles the messages from schedule table.

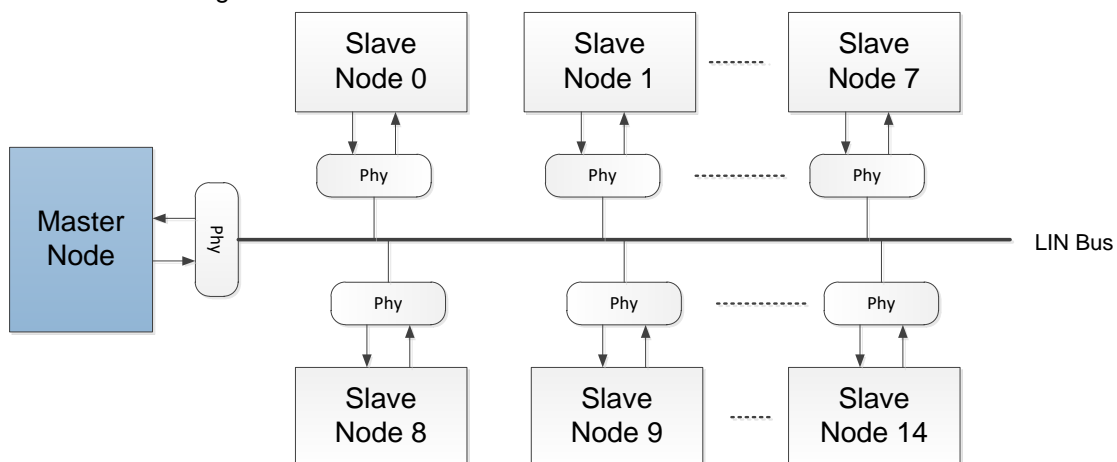


Figure 8-6 LIN bus structure

8.4.6.1 Header transmission

Before requesting the header transmission, the application must set up the identifier, the data field length and configure the message (response direction and checksum type). Once the header has been transmitted, the header transmitted flag LINSTS[HRF] is set and an interrupt is generated if the LINIER[HTRIE] is enabled.

8.4.6.2 Response transmission

If the response direction of a special identifier is publisher, the application must fill the data buffer before requesting the header transmission. The LINCORE transmits the data and the checksum according to the data field length and the checksum type. The LINSTS[DTF] bit is set if the response has been sent successfully, otherwise the DTF flag is not set if there are errors.

8.4.6.3 Response reception

If the response direction is receiver, the LINCORE receives response that sending from slave node, after the header is received with a corresponding identifier. The LINSTS[DRF] is set if the response has been received successfully.

8.4.7 Slave mode

When the LINCR1[SM] bit is set, the slave mode is selected. In the slave mode, the LINCORE receives the headers from the master mode, then transmits or receives response.

8.4.7.1 Header reception

The LINCORE detects break, break delimiter, sync field and identifier in slave mode. A valid break field is longer than 10 bits or 11 bits dominant time. After each LIN Break delimiter, the LINCORE detects five falling edges, then measures the baud rate when the re-synchronization function is enabled. This function makes the slave frequency tolerance to be better than $\pm 14\%$.

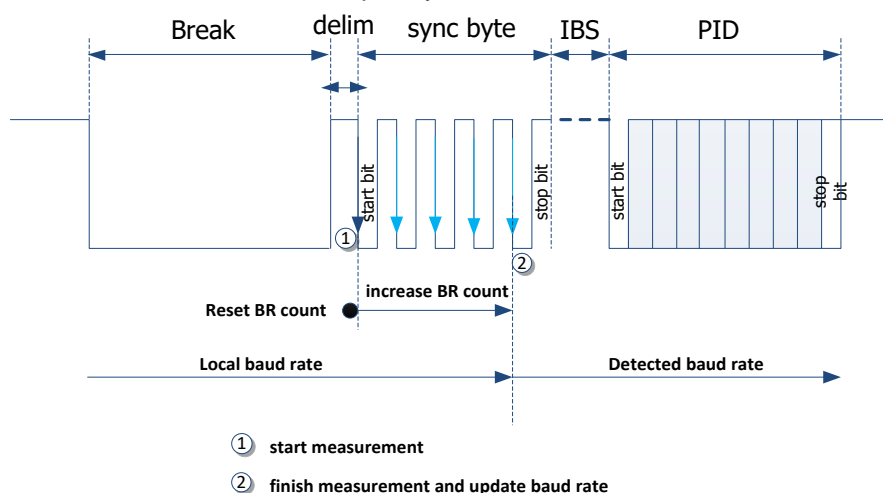


Figure 8-7 LIN header reception

After sync field, the LINCORE receives the identifier, check the parity data when the Manual Checksum is disable.

8.4.7.2 Response transmission

The application checks the identifier after receives the header, if the response direction is publisher, then fills the data buffer, sets the data field and triggers the data transmission by setting the LINCR2[RTR] bit. If the received identifier comes from an identifier filter, the software only needs to fill the data buffer before transmission. The LINSTS[DTF] bit is set if the response has been sent successfully.

8.4.7.3 Response reception

If the response direction is receiver, the LINCORE receives response that sending from other node, after the header is received with a corresponding identifier. The LINSTS[DRF] is set, if the response has been received successfully.

8.4.7.4 Filters

16 identifier filters can be configured in filter mode, with each filter identify one identifier. These filters can be configured in mask mod to handle more identifiers. In this mode, the odd number filter registers are used as identifier registers, and the even number filter registers are used as mask registers. The mask register sets this bit to 1 to indicate that it is irrelevant and can be ignored. If you configured filter in mask mode with don't care bits, that means more than received identifiers will match this filter. Please make sure that these identifiers must have the same response direction, data length and checksum type.

8.4.8 Register definition

Table 8-2 LIN register map

LIN: 0x40007000

ADDRESS	TITLE	DESCRIPTION
LIN + 0x000	LINSYS	LIN system control register
LIN + 0x004	LINCTRL1	LIN control register 1
LIN + 0x008	LIN_CTRL2	LIN control register 2
LIN + 0x00C	LINIEN	LIN interrupt enable register
LIN + 0x010	LINSTS	LIN status register
LIN + 0x014	LINESTS	LIN error status register
LIN + 0x018	LINTO1	LIN time out ctrl register
LIN + 0x01C	LINTO2	LIN timeout ctrl register
LIN + 0x020	LINIBR	LIN integer baud rate register
LIN + 0x024	LINFBR	LIN fractional baud rate register
LIN + 0x028	LINCS	LIN Checksum field register
LIN + 0x02C	LINFRM	LIN Frame Control register
LIN + 0x030	BDLR	LIN Buffer data low register
LIN + 0x034	BDHR	LIN Buffer data high register
LIN + 0x038	LIN_IFEN	LIN ID filter enable register
LIN + 0x03C	LIN_IFMR	LIN ID filter mode register
LIN + 0x040	LIN_IFMI	LIN ID filter match index register
LIN + 0x044	LIN_IFCR2n	LIN ID filter control register Offsets: 0x0044–0x007C (8 registers)

ADDRESS	TITLE	DESCRIPTION
LIN + 0x080	LIN_IFCR2n+1	LIN ID filter control register Offsets: 0x0048–0x0080 (8 registers)

0x000 LINSYS LIN system control register

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name																LM
Type																WR
Reset																0x2

Bit(s)	Mnemonic	Name	Description
[1:0]	LM	LIN_MODE	0x0 = Normal mode 0x1 = Initial mode 0x2 = Sleep mode 0x3 = undefined mode

0x004 LINCTRL1 LIN control register 1

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name									DIO B			RIL		RSL		HIL
Type									WI_ R			WI_R		WI_R		WI_R
Reset									0			0		0		0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	BDL	CS M	RSE	AW U				BTL				BDT	SC M	STM	LB M	SM
Type	WI_ R	WI_ R	WI_ R	WI_ R				WI_R				WI_ R	WI_ R	WI_ R	WI_ R	WI_ R
Reset	0	0	0	0				3				0	0	0	0	0

Bit(s)	Mnemonic	Name	Description
0	SM	Slave mode	0: Master mode 1: Slave mode
1	LBM	Loopback mode	0: Disable 1: Enable
2	STM	Self-test mode	0: Disable 1: Enable
3	SCM	Software control mode	0: Disable 1: Enable (not use filter)
4	BLT	Break length threshold	Break length threshold in slave mode 0: 11-bit 1: 10-bit
[11:8]	BTL	Break TX length	Break field transmit length in mater mode 0: 10-bit time 1: 11-bit time - 15: 25-bit time
12	ATWU	AUTO_WAKEUP	This bit controls the behavior of the LIN hardware during Sleep mode. 0: The Sleep mode is exited on software. 1: The Sleep mode is exited automatically by hardware on

Bit(s)	Mnemonic	Name	Description
			LINRX dominant state detection.
13	RSE	Re-synchronization enable	LIN slave automatic resynchronization enable 0: Automatic re-synchronization disable. 1 : Automatic re-synchronization enable.
14	MCS	Manual Checksum mode	Disable the hardware checksum calculation 0: Hardware automatic checksum calculation 1 : SW manually load the checksum
15	BDL	Break delimiter length	Break delimiter length in master mode 0: 1-bit 1 : 4-bit
[17:16]	HIL	Header inter length	Header inter-byte length between sync field and PID field in master mode 0: 0-bit time 1: 2-bit time 2: 4-bit time 3: 8-bit time
[19:18]	RSL	Response space length	Response inter space length, only for master mode 0: 0-bit time 1: 1-bit time 2: 4-bit time 3 : 8-bit time
[21:20]	RIL	Response inter length	Response inter-byte length between response data 0: 0-bit time 1: 1-bit time 2: 2-bit time 3 : 4-bit time
[23]	DIOB	Disable IDLE on bit error	LIN state machine goes to IDLE on bit error detection 0: Enable IDLE on bit error 1 : Disable IDLE on bit error

0x008 LIN_CTRL2

LIN control register 2

Bit	7	6	5	4	3	2	1	0
Name				WTR	RDR	RTR		HTR
Type				W_R0	W_R0	W_R0		W_R0
Reset				0	0	0		0

Bit(s)	Mnemonic	Name	Description
0	HTR	Header TX request	Header Transmission Request (write 1) Set by software to request the transmission of the LIN header. Cleared by hardware when the request has been completed or aborted. Note: This bit can be written when LIN core is in normal mode (LINSYS.LM=0)
2	RTR	Response TX request	Data Transmission Request (write 1) Set by software in Slave mode to request the transmission of the LIN Data field stored in the Buffer data register. This bit can be set only when HRF bit in LINSTS is set. Cleared by hardware when the request has been completed or aborted or on an error condition. In Master mode, this bit is set by hardware when

Bit(s)	Mnemonic	Name	Description
			LINFRM[DIR] = 1 and header transmission is completed. Note: This bit can be written when LIN core is in normal mode (LINSYS.LM=0)
3	RDR	Response Discard Request	Response Discard Request (write 1) Set by software to stop data reception if the frame does not concern the node. This bit is reset by hardware once LIN has moved to idle state. In Slave mode, this bit can be set only when HRF bit in LINSTS is set and identifier did not match any filter. Note: This bit can be written when LIN core is in normal mode (LINSYS.LM=0)
4	WUTR	Wakeup TX request	Wake-up TX Request (write 1) Setting this bit generates a wake-up pulse. It is reset by hardware when the wake-up character has been transmitted. The character sent is copied from DATA0 in TX buffer. Note that this bit cannot be set in Sleep mode. Software has to exit Sleep mode before requesting a wake-up. Bit error is not checked when transmitting the wake-up request. Note: This bit can be written when LIN core is in normal mode (LINSYS.LM=0)

0x00C LINIEN

LIN interrupt enable register

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	LZIE	TOIE	BEIE	CEIE	HEIE		BOIE	FEIE			WUIE			DRIE	DTIE	HTRIE
Type	RW	RW	RW	RW	RW		RW	RW			RW			RW	RW	RW
Reset	0	0	0	0	0		0	0			0			0	0	0

Bit(s)	Mnemonic	Name	Description
0	HRIE	Header Received Interrupt Enable	Header Received Interrupt Enable 0 : No interrupt when a valid LIN header has been received. 1 : Interrupt generated when a valid LIN header has been received, that is, HRF bit in LINSTS is set.
1	RTIE	Response Transmitted Interrupt Enable	Response Transmitted Interrupt Enable 0 : No interrupt when data transmission is completed. 1 : Interrupt generated when data transmitted flag (DTF) is set in LINSTS.
2	RRIE	Response Reception Complete Interrupt Enable	Response Reception Complete Interrupt Enable 0 : No interrupt when data reception is completed. 1 : Interrupt generated when data received flag (DRF) in LINSTS is set.
5	WUIE	Wake-up Interrupt Enable	Wake-up Interrupt Enable 0 : No interrupt when WUF bit in LINSTS is set. 1 : Interrupt generated when WUF bit in LINSTS is set.
8	FEIE	Framing Error Interrupt Enable	Framing Error Interrupt Enable 0 : No interrupt on Framing error. 1 : Interrupt generated on Framing error.
9	BOIE	Buffer Overrun Interrupt Enable	Buffer Overrun Interrupt Enable 0 : No interrupt on Buffer overrun. 1 : Interrupt generated on Buffer overrun.
11	HEIE	Header Error Interrupt Enable	Header Error Interrupt Enable 0 : No interrupt on Break Delimiter error, Synch Field error, Identifier field error. 1 : Interrupt generated on Break Delimiter error, Synch Field

Bit(s)	Mnemonic	Name	Description
			error, Identifier field error.
12	CEIE	Checksum Error Interrupt Enable	Checksum Error Interrupt Enable 0 : No interrupt on Checksum error. 1 : Interrupt generated when checksum error flag (CEF) in LINSTS is set.
13	BEIE	Bit Error Interrupt Enable	Bit Error Interrupt Enable 0 : No interrupt when BEF bit in LINSTS is set. 1 : Interrupt generated when BEF bit in LINSTS is set.
14	TOIE	Timeout Interrupt Enable	Timeout Interrupt Enable 0 : No interrupt when TOEF is set. 1 : Interrupt generated when TOEF is set.
15	LZIE	Long Zero Error Interrupt Enable	Long Zero Error Interrupt Enable 0 : No interrupt when LZEF is set. 1 : Interrupt generated when LZEF is set.

0x010 LINSTS

LIN status register

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name					LINS					RRIP	RTIP	RPS	WUF	DRF	DTF	HTRF
Type					R					R	R	R	W1C	W1C	W1C	W1C
Reset					0					0	0	0	0	0	0	0

Bit(s)	Mnemonic	Name	Description
0	HRF	Header Transmission Reception Flag	Header Transmission/ Reception Flag This bit is set by hardware and indicates a valid header reception is completed. This bit must be cleared by software. This bit is reset by hardware in Initialization mode and at end of completed.
1	DTF	Data Transmission Completed Flag	Data Transmission Completed Flag This bit is set by hardware and indicates the data transmission is completed. This bit must be cleared by software and the next frame's header received event. It is reset by hardware in Initialization mode.
2	DRF	Data Reception Completed Flag	Data Reception Completed Flag This bit is set by hardware and indicates the data reception is completed. This bit must be cleared by software. It is reset by hardware in Initialization mode. Note: This flag is not set in case of bit error or framing error.
3	WUF	Wake-up Flag	Wake-up Flag This bit is set by hardware and indicates to the software that LIN has detected a falling edge on the LINRX pin when: – Slave is in Sleep mode. – Master is in Sleep mode or idle state. This bit must be cleared by software. It is reset by hardware in Initialization mode. An interrupts generated if WUIE bit in LINIER is set.
4	PS	LIN receive pin state	LIN receive pin state This bit reflects the current status of LINRX pin for diagnostic purposes

Bit(s)	Mnemonic	Name	Description
5	RTIP	Response TX In Progress	Response TX in Progress LIN is in Transmitting data
6	RRIP	Response RX In Progress	Response RX in Progress LIN is Transmitting data
7	RSV	Reserved	Reserved
[11:8]	STS	LIN mode status	LIN mode status 0000: Idle This state is entered on several events: – SLEEP bit and INIT bit in LIN_SYS_CTRL have been cleared by software. – A falling edge has been received on RX pin and AWUM bit is set. – The previous frame reception or transmission has been completed or aborted. 001: Break In Slave mode, a falling edge followed by a dominant state has been detected. Receiving Break. <i>Note: In Slave mode, in case of error new LIN state can be either Idle or Break depending on last bit state. If last bit is dominant new LIN state is Break, otherwise Idle.</i> In Master mode, Break transmission ongoing. 0010: Break Delimiter In Slave mode, a valid Break has been detected for break length configuration (10-bit or 11-bit). Waiting for a rising edge. In Master mode, Break transmission has been completed. Break Delimiter transmission is ongoing. 0011: Synch Field In Slave mode, a valid Break Delimiter has been detected (recessive state for at least one bit time). Receiving Synch Field. In Master mode, Synch Field transmission is ongoing. 0100: Parity Identifier Field In Slave mode, a valid Synch Field has been received. Receiving Identifier Field. In Master mode, identifier transmission is ongoing. 0101: Header reception/transmission completed In Slave mode, a valid header has been received and identifier field is available. In Master mode, header transmission is completed. 0110: Data reception/transmission Field Response reception/transmission is ongoing. 0111: Checksum Data reception/transmission completed. Checksum reception/transmission ongoing. 1111: Wake up transmission field. Wake up transmission ongoing.

0x014 LINESTS

LIN error status register

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name							LZEF	TOEF	BEF	CEF	SFEF	BDEF	IPEF	FEF	BOF	NF
Type							W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset							0	0	0	0	0	0	0	0	0	0

Bit(s)	Mnemonic	Name	Description
0	NF	Noise Flag	This bit is set by hardware when noise is detected on a received character. Note: This bit needs to be cleared by software.
1	BOF	Buffer Overrun Flag	This bit is set by hardware when a new data byte is received and the buffer full flag is not cleared. Note: This bit needs to be cleared by software.
2	FEF	Framing Error Flag	This bit is set by hardware and indicates to the software that LINCORE has detected a framing error (invalid stop bit). This error can occur during reception of any data in the response field (Master or Slave mode) or during reception of Synch Field or Identifier Field in Slave mode.
3	IPEF	Identifier Parity Error Flag	This bit is set by hardware and indicates that an identifier Parity error occurred. Note: Header interrupt is triggered when SFEF or BDEF or IPEF bit is set and HEIE bit in LINIEN is set.
4	BDEF	Break Delimiter Error Flag	Break Delimiter Error Flag This bit is set by hardware and indicates that the received Break Delimiter is too short (less than one bit time).
5	SFEF	Synch Field Error Flag	Synch Field Error Flag This bit is set by hardware and indicates that a Synch Field error occurred (inconsistent Synch Field).
6	CEF	Checksum Error Flag	This bit is set by hardware and indicates that the received checksum does not match the hardware calculated checksum. Note: This bit needs to be cleared by software.
7	BEF	Bit Error Flag	This bit is set by hardware and indicates to the software that LINC has detected a bit error. This error can occur during response field transmission (Slave and Master modes) or during header transmission (in Master mode). Note: This bit needs to be cleared by software.
8	TOEF	Time Out Error Flag	Time out Error event occurs either for header time out or response time out.
9	LZEF	Long Zero Error Flag	This bit is set by hardware when the bus is dominant for more than a 256-bit time. If the dominant state continues. It is cleared by software.

0x018 LINTO1 LIN time out ctrl register

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name																HTO
Type																WI_R
Reset																0x18

Bit(s)	Mnemonic	Name	Description
[5:0]	HTOV	header timeout value	This field contains the header timeout duration (in bit time). This value does not include the Break and the Break Delimiter and the sync field. Note: if the written value is less than or equal to 11, the value will be overrode as 11

0x01C LINTO2 LIN timeout ctrl register

Bit	7	6	5	4	3	2	1	0
Name								TOEN
Type								WI
Reset								0

Bit(s)	Mnemonic	Name	Description
0	TOEN	Time out enable	Time out enable bit 0 : Disable 1 : Enable

0x020 LINIBR LIN integer baud rate register

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name																IBR
Type																WR
Reset																0

Bit(s)	Mnemonic	Name	Description
[11:0]	IBR	Integer Baud Rate divide value	Integer Baud Rate divide value This field can be written in Initialization mode only.

0x024 LINFBR LIN fractional baud rate register

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name																FBR
Type																WL_R
Reset																0

Bit(s)	Mnemonic	Name	Description
[3:0]	FBR	Fraction Baud Rate divide value	Fraction Baud Rate divide value This field can be written in Initialization mode only.

0x028 LINCS LIN Checksum field register

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name																CS
Type																RW
Reset																0

Bit(s)	Mnemonic	Name	Description
[7:0]	CS	Checksum Field	Checksum field Read only in auto checksum mode. Read and write in manual checksum mode if the LINCTRL1.CSM bit is set.

0x02C LINFRM LIN Frame Control register

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name							DFL	DIR	CST							ID
Type							RW	RW	RW							RW
Reset							0	0	0							0

Bit(s)	Mnemonic	Name	Description
[5:0]	ID	Identifier	Identifier part of the identifier field without the identifier parity.
8	CST	Checksum type	This bit controls the type of checksum applied on the current message. 0 : Enhanced Checksum covering Identifier and Data fields. This is compatible with LIN specification 2.0 and higher. 1 : Classic Checksum covering Data fields only. This is compatible with LIN specification 1.3 and earlier.
9	DIR	Direction	This bit controls the direction of the data field. 0 : LIN receives the data and copies them in the BDRL and BDRH registers. 1 : LIN transmits the data from the BDRL and BDRH registers.
[12:10]	DFL	Data field length	This field defines the number of data bytes in the response part of the frame. The value range 0-7 represents 1-8 bytes of data respectively. The master and slave need to configure this register.

0x030 BDLR Buffer data low register

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	DATA3								DATA2							
Type	RW								RW							
Reset	0								0							
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	DATA1								DATA0							
Type	RW								RW							
Reset	0								0							

0x034 BDHR LIN Buffer data high register

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	DATA7								DATA6							
Type	RW								RW							
Reset	0								0							
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	DATA5								DATA4							
Type	RW								RW							
Reset	0								0							

0x038 LIN_IFEN LIN ID filter enable register

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	EN															
Type	RW															
Reset	0															

Bit(s)	Mnemonic	Name	Description
[15:0]	EN	Filter Number Enable	Filter Number start from 0 to 15. In list mode, each EN corresponds to each filter. In mask mode, EN [2n+1] have no effect on the corresponding filters as they act as the masks for the filter 2n. 0 : filter is deactivated 1 : filter is activated

0x03C LIN_IFMR LIN ID filter mode register

Bit	7	6	5	4	3	2	1	0
Name	IFM							
Type	RW							
Reset	0							

Bit(s)	Mnemonic	Name	Description
[7:0]	IFM	ID filter Mode	<p>0 : Filters 2n and 2n + 1 are in identifier filter mode. 1 : Filters 2n and 2n + 1 are in mask mode (filter 2n + 1 is the mask for the filter 2n).</p> <p>Note : "n" ranges from 0 to 7 , corresponds to the bit index of IFM</p>

0x040 LIN_IFMI LIN ID filter match index register

Bit	7	6	5	4	3	2	1	0
Name	IFMI[4:0]							
Type	RW							
Reset	0							

Bit(s)	Mnemonic	Name	Description
[7:0]	IFMI	ID filter Match Index	<p>This register contains the index corresponding to the received identifier. It can be used to directly write or read the data in SRAM. When no filter matches, IFMI[4:0] = 0. When Filter n is matching, IFMI[4:0] = n + 1.</p>

0x044 LIN_IFCR2n LIN ID filter control register Offsets: 0x0044–0x007C (8 registers)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	DFL			DIR		CST	ID									
Type	RW			RW		RW	RW									
Reset	0			0		0	0									

Bit(s)	Mnemonic	Name	Description
[5:0]	ID	Identifier	Identifier part of the identifier field without the identifier parity.
8	CST	Checksum Type	<p>This bit controls the type of checksum applied on the current message.</p> <p>0 : Enhanced Checksum covering Identifier and Data fields. This is compatible with LIN specification 2.0 and higher. 1 : Classic Checksum covering Data fields only. This is compatible with LIN specification 1.3 and earlier.</p>
9	DIR	Direction	<p>This bit controls the direction of the data field.</p> <p>0 : LIN receives the data and copies them in the BDRL and BDRH registers. 1 : LIN transmits the data from the BDRL and BDRH registers.</p>
[12:10]	DFL	Data Field Length	This field defines the number of data bytes in the response part of the frame. The value range 0-7 represents 1-8 bytes of data respectively. The master and slave need to configure this register.

0x080 LIN_IFCR2n+1 LIN ID filter control register Offsets: 0x0048–0x0080 (8 registers)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name				DFL			DIR	CST			ID					
Type				RW			RW	RW			RW					
Reset				0			0	0			0					

Bit(s)	Mnemonic	Name	Description
[5:0]	ID	Identifier	Identifier part of the identifier field without the identifier parity.
8	CST	Checksum Type	This bit controls the type of checksum applied on the current message. 0 : Enhanced Checksum covering Identifier and Data fields. This is compatible with LIN specification 2.0 and higher. 1 : Classic Checksum covering Data fields only. This is compatible with LIN specification 1.3 and earlier.
9	DIR	Direction	This bit controls the direction of the data field. 0 : LIN receives the data and copies them in the BDRL and BDRH registers. 1 : LIN transmits the data from the BDRL and BDRH registers.
[12:10]	DFL	Data Field Length	This field defines the number of data bytes in the response part of the frame. The value range 0-7 represents 1-8 bytes of data respectively. The master and slave need to configure this register.

9 UART

9.1 Introduction

The UART (Universal Asynchronous Receiver/Transmitter) is a basic peripheral for serial communication. It operates to achieve many functions mainly by transmitter and receiver. In [Table 9-1](#), main functions are classified and listed.

Table 9-1 Function classification and configuration

Functions	LINEN	RS485EN	MULCOMEN
BASIC UART	0	0	0
RS485	0	1	0
LIN	1	0	0

Note1: Only UART1 support hardware flow control function; Only UART6 support soft LIN.

Note2: Hardware flow control function must be disabled when it is under RS485 mode.

Note3: LIN mode just supports 8-bit data format and 16 times oversample. Meanwhile, if auto baud rate function is enabled (LABAUDEN=1), the synchronous field data (0x55) will be discarded.

Note4: DMA function must operate with FIFO enable.

9.2 Features

- Full duplex, standard NRZ format.
- Programmable baud rate (16 bit divisor).
 - The baud rate range is 600 bps to 3 Mbps, baud rate error is less than 1%.
- Polling or interrupt driven
 - Transmitter data register empty, transmission complete.
 - Receiver data register full.
 - Receive overrun error, frame error, parity error.
 - Idle line detect.
 - Break detect supporting LIN.
 - Active edge detects for wake up from stop mode.
- Support DMA.
- Programmable 8-/9-bit data length, 1/2 stop bit, parity hardware generation and checking.
- Selectable transmitter output and receiver input polarity.
- Support hardware flow control.
- 13-bit break character generation and optional 10-/11-bit break character detection for LIN function.
- Support RS485 with direction auto-control.

The diagram illustrates the internal architecture of the UART peripheral, divided into TX (Transmit) and RX (Receive) control logic and shifters.

TX Control Logic and Transmitter Shifter:

- Inputs:**
 - UART_TX:** The transmit data input, which can be inverted (INVTX) before being shifted out.
 - TXEN:** Transmit Enable signal.
 - SDBRK:** Stop Delay Before Receive Enable signal.
 - Hardware Flow Control:** Receives **UART_CTS_n** and **UART_RTS_n** signals.
- Internal Structure:**
 - A **2 bytes FIFO** and a **THR** (Transmit Holding Register) interface with the **write** APB bus.
 - A **TX shift register** with 9 bits (start, 0-8, stop).
 - Control Logic:** Manages the TX shift register, including **LOOP** and **SUB** (subtract) operations.
 - Baudrate Generator:** Receives **DIV_L**, **DIV_H**, and **FRACDIV_L** from the APB **write** bus to generate the **tx clock**.
- Outputs:**
 - TXDMAEN:** TX DMA Enable signal.
 - TX fifo or THR empty:** Signal indicating the TX FIFO or THR is empty.
 - TX dma request:** DMA request signal.

RX Control Logic and Receiver Shifter:

- Inputs:**
 - UART_RX:** The receive data input, which can be inverted (INVRX) before being shifted in.
 - RXEN:** Receive Enable signal.
 - Hardware Flow Control:** Receives **UART_CTS_n** and **UART_RTS_n** signals.
- Internal Structure:**
 - A **2 bytes FIFO** and an **RBR** (Receive Buffer Register) interface with the **read** APB bus.
 - A **RX shift register** with 9 bits (stop, 8-1, 0, start).
 - Control Logic:** Manages the RX shift register and data flow.
 - Baudrate Generator:** Provides the **rx clock** to the RX shifter.
- Outputs:**
 - RXDMAEN:** RX DMA Enable signal.
 - Rx fifo or RBR not empty:** Signal indicating the RX FIFO or RBR is not empty.
 - RX dma request:** DMA request signal.

Error and Interrupt Logic:

- TX Errors:**
 - LBRKIE:** LIN break flag.
 - IDLEIE:** Mulcom idle flag.
 - ETXE:** TX fifo or THR empty.
 - ETC:** TX shifter empty.
 - EDCTS:** CTS_n changing.
- RX Errors:**
 - EOEBI:** Overflow or break error.
 - ENE:** Noise error.
 - EFE:** Frame error.
 - EPE:** Parity error.
 - ERXNE:** RX FIFO or RBR not empty.
- Interrupt:** A logic OR gate combines the TX and RX error signals to generate the **interrupt** signal.

9.4 Input & Output timing

Pin name	Corresponding signal	Width	I/O	Pull up or not	Timing limitation
UARTx_TX	uart_tx	1	O	no	None
UARTx_RX	uart_rx	1	I	no	None
UARTx_RTS	uart_rts_n	1	O	no	None
UARTx_CTS	uart_cts_n	1	1	no	None

Note: $x = 1$ to 6. Just UART1 has the full hardware flow control and its four signals (UART1_TX, UART1_RX, UARTx_RTS, UARTx_CTS) can all be found in PIN.

9.5 Basic UART functions

Basic UART function is to transmit and receive the serial data bit by bit. In Figure 9-2 and Figure 9-3, it illustrates the full data bits including start bit, data bits, parity bit, stop bits and guard time. But the bit6, bit7, bit8, bit9, parity, stop2 bit and guard time can be configured by user described in UARTn_LCR0 and UARTn_LCR1 in detail. One bit is corresponding to one bit time controlled by baud rate.

There exist many statuses for transmitting and receiving. User had better know when the statuses are generated in the transmitting or receiving process. In this way, user can use the UART function better. The status bits THRE and TC just occur in transmitting process illustrated in Figure 9-2. During the initialization condition after global reset or power on, THRE and TC turn to 1 just after TXEN is set to 1. But in the process of transmitting, THRE turns to 1 just after start bit and TC turns to 1 just after the last bit, such as the guard time if GUARDEN=1.

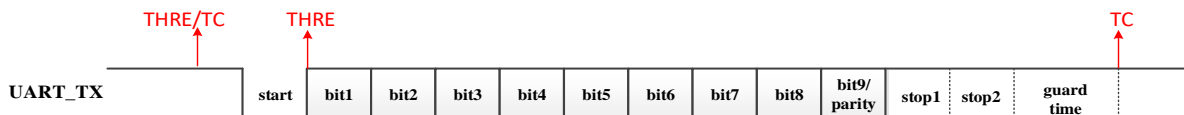


Figure 9-2 UART transmitter flow

The status bit DR, OE, PE, FE, BI and NE are set to 1 just after the stop1 bit if related events occur during receiving process as illustrated in Figure 9-3.



Figure 9-3 UART receiver flow

To point out, the status PE, FE and NE just aims at the current receiving data byte and will be cleared automatically just at the point the next data is received completely. For other statuses, they hold the value for all the time if they are not being cleared by reading or writing 1 to them.

9.5.1 Noise detection

For NE status, it is generated during receiving process when the noise exists in UART_RX signal. In order to detect the noise, UART_RX is sampled three times at the middle position as illustrated in Figure 9-4.

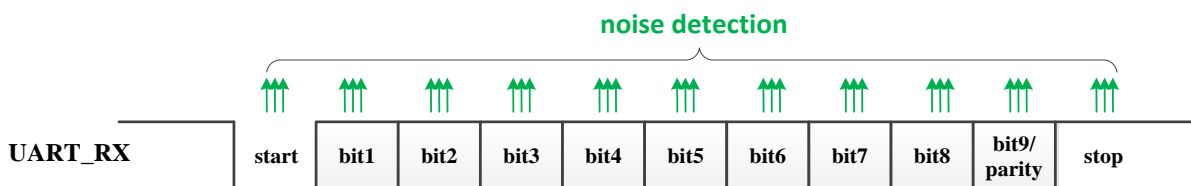


Figure 9-4 UART noise detection

To explain clearly and conveniently, the value of three times sample is called SM1, SM2 and SM3 respectively. If there are more than two '1' among SM1, SM2 and SM3 for start bit, the start bit is invalid and receiver is reset to receive again. Other than this case for start bit, if the value of SM1, SM2 and SM3 are different for each other, the noise is detected with the NE status turns to 1.

9.5.2 Baud rate description

UART Baud rate accuracy is determined by many aspects including UART clock, oversample time and so on. So, some specific and too high baud rate is not realized or realized with large error. The [Table 9-3](#) and [Table 9-4](#) describe the typical baud rate configuration and corresponding error at different system clocks.

Table 9-3 Typical baud rate and error @bclock=50MHz

Number	Theoretical value (Kbps)	Practical value(bps)	DIV_MAN [15:0]	DIV_FRAC [4:0]	Oversample times	Error
1	2.4	2399.98	1302	3	16	-0.001%
2	9.6	9599.69	325	17	16	-0.003%
3	19.2	19202.22	162	24	16	0.006%
4	57.6	57603.69	54	8	16	0.006%
5	115.2	115207.38	27	4	16	0.006%
6	230.4	230414.75	13	18	16	0.006%
7	460.8	460829.50	6	25	16	0.006%
8	921.6	917431.19	3	13	16	-0.452%
9	1843.2	1834862.38	3	13	8	-0.452%
10	4200	4166666.75	1	16	8	-0.794%

Table 9-4 Typical baud rate and error @bclock=36MHz

Number	Theoretical value (Kbps)	Practical value(bps)	DIV_MAN TI[15:0]	DIV_FRAC [4:0]	Oversample times	Error
1	2.4	2400	937	16	16	0
2	9.6	9600	234	12	16	0
3	19.2	19200	117	6	16	0
4	57.6	57600	39	2	16	0
5	115.2	115200	19	17	16	0
6	230.4	230769.23	9	24	16	0.16%
7	460.8	461538.47	4	28	16	0.16%
8	921.6	923076.94	2	14	16	0.16%
9	1843.2	1846153.88	1	7	16	0.16%
10	4200.0	no possible	-	-	-	-

9.6 Hardware flow control function

UART hardware flow controls the communication between two UART devices by RTS_n and CTS_n in order to reduce the CPU workload. In this way, the two communication sides can handle the data one by one leisurely. The practical application connection is described as [Figure 9-5](#).

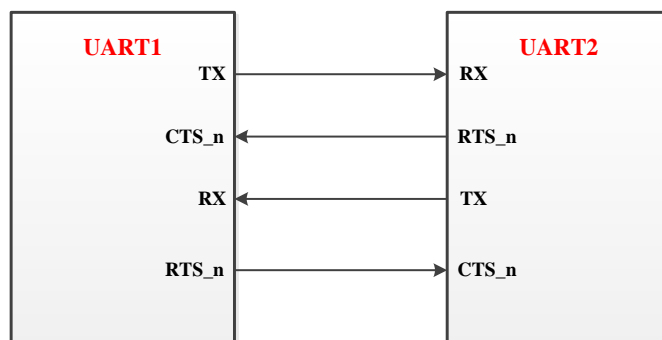


Figure 9-5 Hardware flow control connection

In [Figure 9-5](#), signal RTS_n of UART1 can inform UART2 not to transmit data because UART1 RX data register or FIFO has been full. When UART1 RX data register or FIFO turns to not full by reading, the UART1 RTS_n turns to low automatically. Then UART2 can transmit data by detecting the UART2 CTS_n low. In the similar way, the UART2 RTS_n and UART1 CTS_n operate to control the UART1 transmission.

Especially, if the UART2 CTS_n turns to high during the UART2 is transmitting data, the current data will be transmitted completely. So, user should usually check the CTS or CTS_n status to use the hardware flow control function in addition to other status bits.

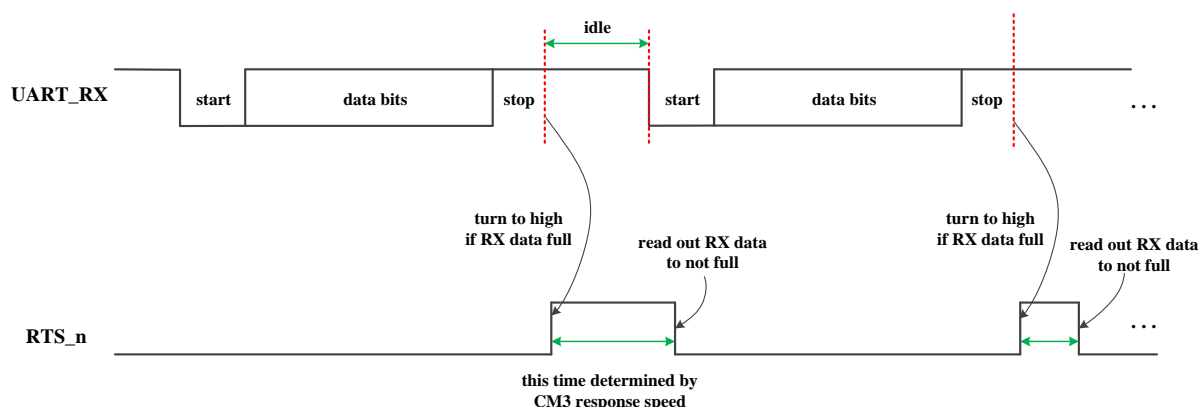


Figure 9-6 Hardware flow control principle

9.7 RS485 function

Compared with basic UART function, RS485 function generates an auto direction control signal UART_RTS, which is default low for receiving the data and high for transmitting data illustrated as [Figure 9-7](#) and [Figure 9-8](#). Because of the half-duplex, RS485 can implement one of the transmitting or receiving operation for just one moment. In the [Figure 9-7](#), there are two delays: delay1 is used for pulling up the UART_RTS signal before practical data transmission and guard time is used for pulling down the UART_RTS after the data bit transmission has absolutely been finished. Practical PCB

routing delay may lead to UART_RTS signal turning to high later than UART_TX so that the first data bits may be damaged. So, the delays help to guarantee the UART_RTS is high during the whole transmission. After transmission is finished, UART_RTS will go to low automatically to let the UART in receiving state.

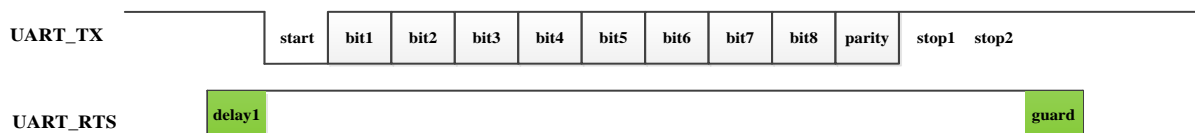


Figure 9-7 Single byte data transmission

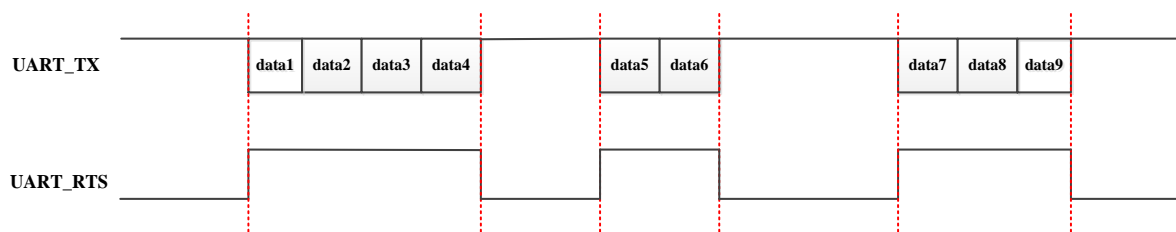


Figure 9-8 Multiple bytes data transmission

Figure 9-9 describes a typical case of the user application connection. UART_RTS acts as a direction control signal to MAX485. With this connection method, the default value of UART_RTS is low so that MAX485 is under default receiving condition. When UART wants to transmit data, the signal UART_RTS is set to high by hardware.

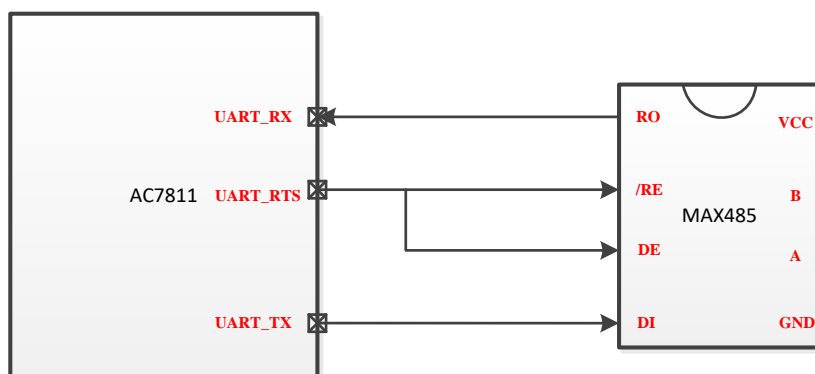


Figure 9-9 Practical circuit connection

9.8 LIN function

The UART LIN is just a soft LIN which transmits the break field, synchronous field and data respectively controlled by user as Figure 9-10. User can learn more details in the newest LIN protocol and the figure just illustrates a basic frame condition. User should pay more attention to the UARTn_LINCR register in addition to the basic UART registers. In UART module, there exists hardware logics for LIN detection and it is enabled when LINEN is configured to 1.

Firstly, description for receiving process are introduced based on the serial data stream appears on the UART_RX as Figure 9-10. Firstly, when UART receives 10 (LBRKDL=0) or 11 (LBRKDL=1) bit zero, UART LIN detection logics treat it as a valid break field for LIN frame and the LIN break flag

FBRK is set to 1 by hardware to indicate a valid LIN break field has occurred. To point out, the LIN break is not a data and not stored into the RX data register or FIFO. Then, delimiter bits with UART_RX high occurs following the break field. After delimiter bits period, synchronous field namely 0x55 acts as a normal data for receiving. If LABAUDEN is configured to 1, auto baud rate detection is in operation during synchronous field and the auto baud rate detection operation is finished just after the fifth rising edge illustrated in Figure 9-10. But the data 0x55 will not be stored into RX data register or FIFO. If LABAUDEN is configured to 0, auto baud rate detection is not performed and the data 0x55 is stored into the RX data register or FIFO. After synchronous field, the data streams are the useful data for user and received by UART without difference.

To avoid the module halting when exception condition occurs, time out mechanism is brought in. As illustrated in Figure 9-10, more than 25 bit time for break field, more than 14 bit time for delimiter and more than 13 bit time for synchronous field can lead the module to reset the receiver and LIN detection logics.

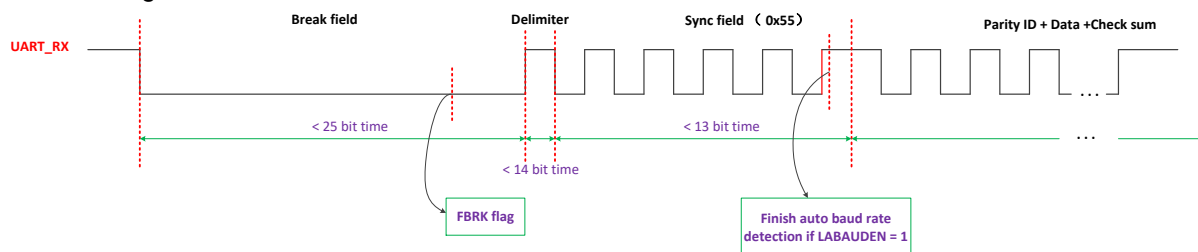


Figure 9-10 LIN frame flow

Then, introduction for transmitting LIN protocol frame is described in this paragraph. As software LIN, how to transmit the break field is the key procedure. When user wants to transmit the break field, user should check the UARTn_LSR0[THRE] status. If the UARTn_LSR0[THRE] value is 1, user can write 1 to the UARTn_LINCR[SBRK] to send a 13-bit time break field with other control bits not changed in LINCR. To point out, when UARTn_LINCR[SBRK] has been written to 1, the break field will be transmitted just after the current data has been transmitted completely or be transmitted immediately when the UART is in idle state. The UARTn_LINCR[SBRK] will be cleared automatically by hardware. And then the synchronous field(0x55) and other following data can be written to the THR data register and sent as normal data transmitting.

9.9 Two chip power mode

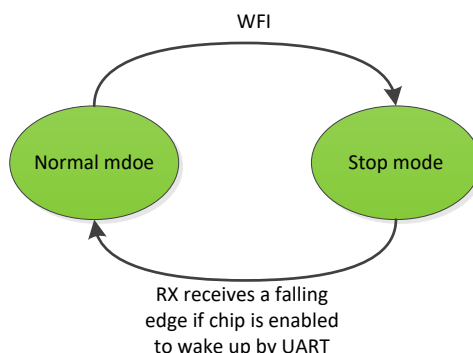


Figure 9-11 Chip normal mode and stop mode condition

There are two power modes designed for the chip. As the [Figure 9-11](#) illustrates, user can execute the WFI instruction to make the chip go into stop mode, in which the chip power consumption will be much less obviously and the UART module will power down and be reset by default. Certainly, all the UART configuration and other registers are reset for the stop mode condition.

In stop mode, the chip can wake up to normal mode by UART receiving a falling edge if the chip is enabled to wake up by UART. Because of the time required for the wake-up flow, UART can normally receive data just after the sum time of 5ms and re-configure UART time at least from receiving the wake-up falling edge. In details, TX1 can send 0xFF to act as a falling edge and actually other data is also ok. Specially, data will be lost in RX2 if TX1 send some data to RX2 during the wake up flow.

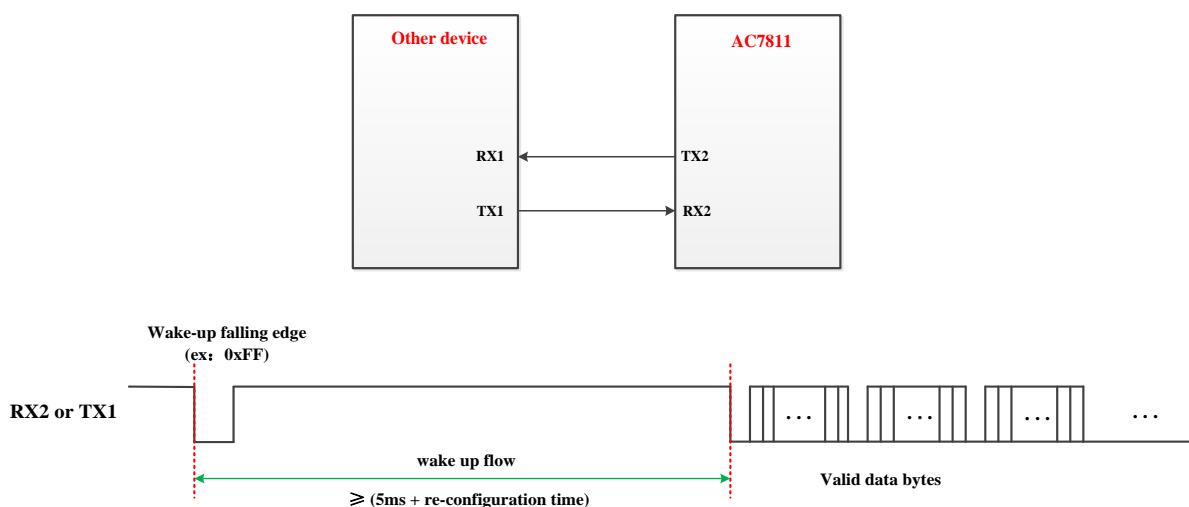


Figure 9-12 Typical flow for waking up the chip by UART

9.10 Baud rate configuration description

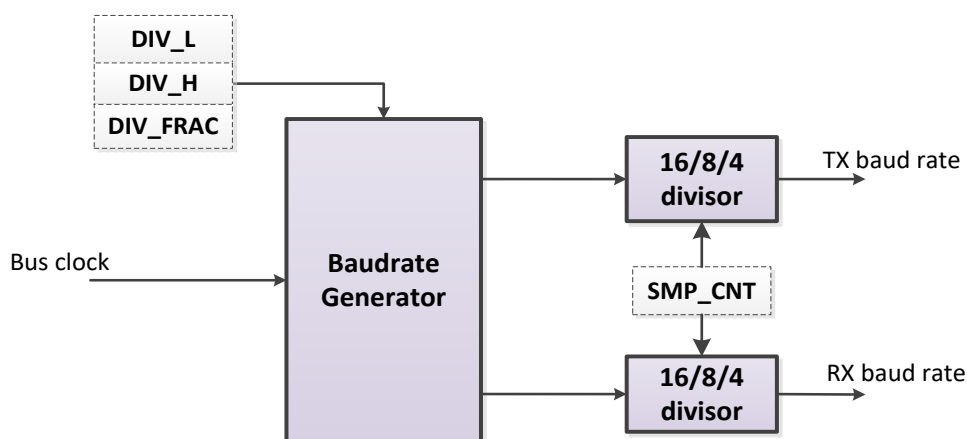


Figure 9-13 Diagram for baud rate generator

As illustrated in the [Figure 9-13](#), the baud rate related configuration registers are as follows: UARTn_SMP_CNT/UARTn_DIV_H/UARTn_DIV_L / UARTn_DIV_FRAC. And the detailed information is described in register map. For configuration, the formula below is used for baud rate configuration.

$$Baudrate = \frac{f_{clk}}{DIV * SMP_CNT}$$

Note: $DIV = \{UARTn_DIV_H, UARTn_DIV_L\}$; $SMP_CNT = UARTn_SMP_CNT$;

For example:

If user wants to get baud rate 230400 bps at 8 times sample mode with 50MHz bus frequency, so we can get the DIV configuration as follows. So, **UARTn_DIV_L=13, UARTn_DIV_H=0, UARTn_DIV_FRAC=32 * 0.5634=18.**

$$DIV = \frac{50000000}{Baudrate * SMP_CNT} = \frac{50000000}{230400 * 8} = 13.5634$$

Combined with the example above, UARTn_DIV_FRAC[4:0] can be explained clearly. In the expression above, DIV is not an integer. If user removes the fractional part, the accuracy of baud rate will decrease. Especially in large baud rate condition, the discard of DIV fractional part may reduce the accuracy to a low level so that the normal data transmission will make a mistake.

In order to keep high accuracy, UARTn_DIV_FRAC[4:0] is configured to represent the DIV fractional part. Because of 5 bits width, UARTn_DIV_FRAC ranges from 0 to 31. So, 32 multiplied by the DIV fractional part generates the configuration value for UARTn_DIV_FRAC[4:0].

9.11 Configure note

Configuration steps:

- (1) TX/RX data storage mode: UARTn_FCR.
- (2) Baud rate: UARTn_DIV_L/UARTn_DIV_H/ UARTn_DIV_FRAC/UARTn_SMP_CNT
- (3) DMA: UARTn_DMA_EN

Note: DMA function must operate with FIFO.

- (4) Data format: UARTn_LCR0/UARTn_LCR1

Note: SB bit must be taken care of.

- (5) Function configuration: UARTn_RS485CR/UARTn_LINCR/UARTn_MULCOMCR/UARTn_CNRT / UARTn_SLADDR and so on.

Note: This step is an option for different function.

- (6) Interrupt enable: UARTn_IER
- (7) Transmitter and receiver enable: UARTn_LCR1[TXEN] / UARTn_LCR1[RXEN] .
- (8) Transmit or receive data: UARTn_THR/UARTn_RBR

Note: this step is in normal transmitting or receiving data process actually.

Notes:

1. For LIN function, data format must be configured as 8 bit with no parity check with 16 times oversample.
2. For LIN function, sync field data(0x55) will be received and stored into FIFO or RX register when LABAUDEN=0 and will be received and not stored into FIFO or RX register when LABAUDEN=1.
3. For RS485 function, RTS_n PIN is used as the transmitting or receiving direction controlling signal.

9.12 Register Definition

Table 9-5 UART register map

Address = Base address + Offset address

Module name	Base address	Offset address	Comment
UART1	0x40018000	0x00 ~ 0x58	1. No UARTn_LCR 2. Part of the registers are reversed.
UART2	0x40019000	0x00 ~ 0x58	
UART3	0x4001a000	0x00 ~ 0x58	
UART4	0x4001b000	0x00 ~ 0x58	
UART5	0x4001c000	0x00 ~ 0x58	
UART6 (UART_LIN)	0x4001d000	0x00 ~ 0x5c	1. UARTn_LCR. 2. Part of the registers are reversed.

Offset Address	Name	Width	Register function
0x00	UARTn_RBR/THR	32	TX holding register /RX buffer register
0x04	UARTn_DIV_L	32	Divisor Low 8 bits
0x08	UARTn_DIV_H	32	Divisor High 8 bits
0x0C	UARTn_LCR0	32	UART supplementary control register 0
0x10	UARTn_LCR1	32	UART supplementary control register 1
0x14	UARTn_FCR	32	FIFO control register
0x18	UARTn_EFR	32	Hardware flow enable register
0x1c	UARTn_IER	32	Interrupt enable register
0x20	UARTn_LSR0	32	Status register0
0x24	UARTn_LSR1	32	Status register1
0x28	UARTn_SMP_CNT	32	UART sample counter register
0x34	UARTn_GUARD	32	Guard time added register
0x38	Reserved	32	
0x3c	UARTn_SLEEP_EN	32	Sleep enable register
0x40	UARTn_DMA_EN	32	DMA enable register
0x44	UARTn_DIV_FRAC	32	Fractional divider register
0x48	Reserved	32	
0x4c	UARTn_RS485CR	32	RS485 control register
0x50	UARTn_SLADDR	32	Address for RS485 or Multi-processor communication
0x54	UARTn_CNTR	32	Delay time for RS485
0x58	UARTn_MULCOMCR	32	Idle interrupt enable register
0x5c	UARTn_LINCR	32	Software LIN control register

0x00 UARTn_RBR/THR RX/TX Data Register 0000

Bit	32~15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name								RBR/THR								
Type								RO/WO								
Reset								0	0	0	0	0	0	0	0	0

Bit(s)	Mnemonic	Name	Description
8:0	RBR/THR	RBR/THR	RX/TX Data register The received data can be read by accessing this register and the transmitting data can be written into this register. The data width is not more than 9 bits.

0x04 UARTn_DIV_L Divisor low 8 bits register 0001

Bit	32~15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									DIV_L							
Type									RW							
Reset									0	0	0	0	0	0	0	1

Bit(s)	Mnemonic	Name	Description
7:0	DIV_L	DIV_L	Baud rate divisor Divisor low 8 bits.

0x08 UARTn_DIV_H Divisor high 8bits register 0000

Bit	32~15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									DIV_H							
Type									RW							
Reset									0	0	0	0	0	0	0	0

Bit(s)	Mnemonic	Name	Description
7:0	DIV_H	DIV_H	Baud rate divisor Divisor high 8 bits.

0x0C UARTn_LCR0 Control Register 0 0000

Bit	32~15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name										SUB	SP	EPS	PEN	STB	WLS1	WLS0
Type										RW	RW	RW	RW	RW		RW
Reset										0	0	0	0	0	0	0

Note: SB bit configuration must be 0, otherwise tx transmit '0' at any time.

Bit(s)	Mnemonic	Name	Description
6	SUB	SUB	Sets up break 0: No effect 1: SOUT signal is forced into the "0" state.
5	SP	SP	Stick parity 0: No effect. 1: The parity bit is forced into a defined state, depending on the states of EPS and PEN: If EPS = 1 & PEN = 1, the parity bit is set and checked =

Bit(s)	Mnemonic	Name	Description
			0: If EPS = 0 & PEN = 1, the parity bit is set and checked = 1.
4	EPS	EPS	Selects even parity 0: When EPS = 0, an odd number of ones is sent and checked. 1: When EPS = 1, an even number of ones is sent and checked.
3	PEN	PEN	Enables parity 0: The parity is neither transmitted nor checked. 1: The parity is transmitted and checked.
2	STB	STB	Number of STOP bits 0: One STOP bit is always added. 1: Two STOP bits are added after each character is sent; unless the character length is 5 when 1 STOP bit is added.
1:0	WLS1_WLS0	WLS1_WLS0	Selects word length 0: 5 bits 1: 6 bits 2: 7 bits 3: 8 bits 4: 9 bits (wls2)

0x10 UARTn_LCR1 Control Register 1 0000

Bit	32~15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									INVTX	INVRX	WLS2	LOOP			TXEN	RXEN
Type									RW	RW	RW	RW			RW	RW
Reset									0	0	0	0			0	0

Bit(s)	Mnemonic	Name	Description
7	INVTX	INVTX	Determine whether inverse the tx output, including idle, break, data bits,start bit, stop bit. 0: don't inverse tx output 1: inverse tx output
6	INVRX	INVRX	Determine whether inverse the tx output, including idle, break, data bits,start bit, stop bit. 0: don't inverse rx input 1: inverse rx input
5	WLS2	WLS2	Determine whether 9-bit data mode is available 0: not available 1: available
4	LOOP	LOOP	LOOP 0: for user normal use 1: control the uart into loop mode(can used for testing uart by itself)
1	TXEN	TXEN	UART Transmitter enable 0: disable 1: enable
0	RXEN	RXEN	UART Receiver enable 0: disable 1: enable

0x14 UARTn_FCR FIFO Control Register 0000

Bit	32~15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name																FIFOE
Type																RW
Reset																0

Bit(s)	Mnemonic	Name	Description
0	FIFOE	FIFOE	Enables FIFO 0: Disable both RX and TX FIFOs 1: Enable both RX and TX FIFOs.

0x18 UARTn_EFR 0000

Bit	32~15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									AUTO_CTS	AUTO_RTS						
Type									RW	RW						
Reset									0	0						

Overview: AUTOCTS=1 represents enabling the Hardware flow function of pin CTS_n, so if AUTOCTS=1, **user must make the n_CTS pin be tied to a fixed level, such as the other MCU's or device's pin.** If AUTOCTS=0, user needn't to care about the CTS_n pin.

Bit(s)	Mnemonic	Name	Description
7	AUTO_CTS	AUTO_CTS	Enables hardware transmission flow control 0: Disable 1: Enable
6	AUTO_RTS	AUTO_RTS	Enables hardware reception flow control 0: Disable 1: Enable.

0x1c UARTn_IER Interrupt Enable Register 0000

Bit	32~15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									EDCTS	EOEBI	ENE	EFE	EPE	ETC	ETXE	ERXNE
Type									RW	RW	RW	RW	RW	RW	RW	RW
Reset									0	0	0	0	0	0	0	0

Bit(s)	Mnemonic	Name	Description
7	EDCTS	EDCTS	CTS_n changing interrupt enable 0: disable 1: enable
6	EOEBI	EOEBI	Interrupt enable of overflow error or break error 0: disable 1: enable
5	ENE	ENE	Interrupt enable of noise error 0: disable 1: enable
4	EFE	EFE	Interrupt enable of overflow error or frame error 0: disable

Bit(s)	Mnemonic	Name	Description
			1: enable
3	EPE	ELSI	Interrupt enable of parity error 0: disable 1: enable
2	ETC	ETXTCI	Interrupt enable of transmitting completed 0: disable 1: enable
1	ETXE	ETBEI	Interrupt enable of transmitting data empty <i>Note: fifoe=1 represents fifo empty; fifoe=0 represent data register empty.</i> 0: disable 1: enable
0	ERXNE	ERBFI	Interrupt enable of receiving data not empty <i>Note: fifoe=1 represents fifo not empty; fifoe=0 represent data register not empty.</i> 0: disable 1: enable

0x20 UARTn_LSR0 Line Status Register

0020

Bit	32~15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									NE	TC	THRE	BI	FE	PE	OE	DR
Type									W1C	R	R	W1C	W1C	W1C	W1C	R
Reset									0	0	1	0	0	0	0	0

Note: NE/PE/FE error just aims at the current byte data. Meanwhile, OE/BI will exist until it is cleared.

Bit(s)	Mnemonic	Name	Description
7	NE	NE	Noise error flag <i>Note: write 1 to clear this flag to 0.</i> 0: No noise error. 1: Noise error has been occurred.
6	TC	TC	The flag of Transmitting finished <i>Note: write data to TX FIFO(fifoe=1)/TX register(fifoe=0) to clear this flag to 0. For LIN function, set SBRK bit also can clear this flag to 0.</i> 0: TX FIFO(fifoe=1) or TX register(fifoe=0) is not empty, or transmitter has not finished the data shifting. 1: TX FIFO(fifoe=1) or TX register(fifoe=0) is empty and transmitter has finished the data shifting.
5	THRE	THRE	The empty flag of TX holding register or TX FIFO <i>Note: write data to TX FIFO(fifoe=1)/TX register(fifoe=0) to clear this flag to 0.</i> 0: Reset whenever the contents of the TX FIFO are not empty, or whenever TX holding register is not empty (FIFOs are disabled). 1: Set whenever the contents of the TX FIFO are empty, or whenever TX holding register is empty (FIFOs are disabled).
4	BI	BI	Break error flag <i>Note: write 1 to clear this flag to 0.</i> 0: No break error 1: Break error has been occurred. If FIFOs are disabled, this bit is set whenever the SIN is held in the 0 state for more than one transmission time (START bit + DATA bits + PARITY + STOP bit). When a break occurs, only one zero character is loaded into FIFO

Bit(s)	Mnemonic	Name	Description
			or TX holding register.
3	FE	FE	Framing error flag Note: write 1 to clear this flag to 0. 0: No framing error. 1: Framing error has been occurred. This bit will be set if the received data do not have a valid STOP bit.
2	PE	PE	Parity error flag Note: write 1 to clear this flag to 0. 0: No parity error. 1: Parity error has been occurred. This bit will be set if the received data do not have a valid parity bit.
1	OE	OE	Overrun error flag Note: write 1 to clear this flag to 0. 0: No RX overflow error. 1: RX overflow error has been occurred. If FIFOs are disabled, this bit will be set if the RX buffer is not read by the CPU before the new data from the RX shift register overwrites the previous contents. If FIFOs are enabled, an overrun error occurs when RX FIFO is full and the RX shift register becomes full. OE is set as soon as this happens. The character in the shift register is then overwritten, but not transferred to FIFO.
0	DR	DR	Data ready flag Note: Read register UART_RBR/THR, or read all RX FIFO to clear this flag to 0. 0: Data not ready. 1: Data ready. Set by the RX buffer becoming full or RX FIFO not empty(at least one byte being transferred into the FIFO).

0x24 UARTn_LSR1 Line Status Register 0000

Bit	32~15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									RTS	CTS	UART_IDLE		DCTS	FBRK		IDLE
Type									R	R	R		W1C	W1C		W1C
Reset									0	0	0		0	0		0

Bit(s)	Mnemonic	Name	Description
7	RTS	RTS	Hardware flow status – RTS Note: RTS is opposite to the signal of PIN RTS_n. It is not an interrupt source. 0: Under hardware flow control function, it represents RX FIFO or RX register is already full. This signal can inform other device not transmitting data to the MCU. 1: Under hardware flow control function, it represents RX FIFO or RX register is not full.
6	CTS	CTS	Hardware flow status – CTS Note: CTS is opposite to the signal of PIN CTS_n. It is not an interrupt source. 0: Under hardware flow control function, it represents RX FIFO or RX register of other device is already full. This signal can inform the MCU not transmitting the next data. 1: Under hardware flow control function, it represents RX FIFO or RX register of other device is not full.
5	UART_IDLE	UART_IDLE	UART_IDLE

Bit(s)	Mnemonic	Name	Description
			0: uart is at operation. 1: uart is not at operation, namely, transmitter and receiver are not working or has finished the data transmission or reception.
3	DCTS	DCTS	The flag of pin CTS_n signal changing Note: write 1 to clear this flag to 0. 0: no change. 1: represents CTS_n pin signal changing from 1 to 0 or 0 to 1.
2	FBRK	FBRK	The flag of LIN BREAK occurred. Note: write 1 to clear this flag to 0. 0: has not detected the break field in LIN frame just in LIN function. 1: has detected the break field in LIN frame just in LIN function
0	IDLE	IDLE	IDLE flag Receiver has received the data followed by a high level maintaining at least one byte data time. IDLE status flag will be work after the MUI-Communication(MULCOMEN) is enabled. Note: write 1 to clear this flag to 0. 0: idle line has not been detected 1: idle line detected

0x28 **UARTn_SMP_CNT** Sample counter register 0000

Bit	32~15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name																SMP_CNT
Type																RW
Reset															0	0

Bit(s)	Mnemonic	Name	Description
			UART sample counter 0: Based on 16*baud_pulse, baud_rate = system clock frequency/16/{DLH, DLL} 1: Based on 8*baud_pulse, baud_rate = system clock frequency/8/{DLH, DLL} 2: Based on 4*baud_pulse, baud_rate = system clock frequency/4/{DLH, DLL} 3: Based on sampe_count * baud_pulse, baud_rate = system clock frequency/16 /{DLM, DLL}
1:0	SMP_CNT	SMP_CNT	

0x34 **UARTn_GUARD** Guard time register 000F

Bit	32~15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name												GUARD_EN				GUARD_CNT
Type												RW				RW
Reset												0	1	1	1	1

Notes: Adding the guard time can contribute to eliminate the accumulated error every byte, so it is significant to improve the accuracy of the baud rate by using the fraction divisor with the guard time.

Bit(s)	Mnemonic	Name	Description
4	GUARD_EN	GUARD_EN	Guard interval time added enabling signal

Bit(s)	Mnemonic	Name	Description
			0: disable 1: enable
3:0	GUARD_CNT	GUARD_CNT	Guard interval count value 0~15: 0 ~ 15 bit time

0x3C UARTn SLEEP_EN Sleep enable register 0000

Bit	32~15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name																SLEEP_EN
Type																RW
Reset																0

Bit(s)	Mnemonic	Name	Description
			Sleep function enable 0: Does not deal with sleep mode indication signal 1: Activate hardware flow control according to software initial settings when the chip enters the sleep mode. Release the hardware flow when the chip wakes up.
0	SLEEP_EN	SLEEP_EN	

0x40 UARTn DMA_EN 0000

Bit	32~15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name															TX_DMA_EN	RX_DMA_EN
Type															RW	RW
Reset															0	0

Bit(s)	Mnemonic	Name	Description
1	TX_DMA_EN	TX_DMA_EN	TX_DMA mechanism enabling signal 0: Does not use DMA in TX 1: Use DMA in TX. When this register is enabled.
0	RX_DMA_EN	RX_DMA_EN	RX_DMA mechanism enabling signal 0: Does not use DMA in RX 1: Use DMA in RX. When this register is enabled.

0x44 UARTn DIV_FRAC Fractional Divider Address 0000

Bit	32~15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name																DIV_FRAC
Type																RW
Reset																0

Bit(s)	Mnemonic	Name	Description
4:0	DIV_FRAC	DIV_FRAC	Fraction divisor : if actual divisor is 135.65, then DIV_FRAC is $0.65 \times 32 = [20.8] = 21$, and the DIV_L=135.

0x4c UARTn_RS485CR RS485 control register 0000

Bit	32~15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									RS485EN		INVPOL	DLYEN				
Type									RW		RW	RW				
Reset									0		0	0				

Notes:

1. Delay ahead of transmitting is corresponding to the `UARTn_CNTR` ; Delay after the transmitting can use the `UARTn_GUARD`.
2. RS485 function use `PIN RTS_n` as a transmitting or receiving direction control PIN.

Bit(s)	Mnemonic	Name	Description
7	RS485EN	RS485EN	0: disable rs485 mode 1: enable rs485 mode
5	INVPOL	INVPOL	INVPOL 0: don't inverse the polarity of rts_n 1: inverse the polarity of rts_n
4	DLYEN	DLYEN	A DELAY is inserted between the RS485 switch to output state and the actual START bit. The specific DELAY time is determined by the register <code>UART_CNTR</code> . That is, the delay1 corresponding to Figure 9-7. 0: disable delay 1: enable delay

0x50 UARTn_SLADDR ADDRESS FOR WAKE UP 0000

Bit	32~15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Type									RW	RW	RW	RW	RW	RW	RW	RW
Reset									0	0	0	0	0	0	0	0

Note: `RS485en=1` and `mulcomen=1` can't be available simultaneously.

Bit(s)	Mnemonic	Name	Description
7:0	SLADDR	ADDR	SLADDR 7:0: 0~255 as a slave address for RS485 when <code>rs485en=1</code> .

0x54 UARTn_CNTR 0000

Bit	32~15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									CNTR[7:0]							
Type									RW	RW	RW	RW	RW	RW	RW	RW
Reset													0	0	0	0

Bit(s)	Mnemonic	Name	Description
7:0	CNTR	CNTR	COUNTER 7:0: 0~255 bits time for time delay in RS485 mode

0x58 UARTn_MULCOMCR Idle interrupt enable register 0000

Bit	32~15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									MULCOMEN			IDLEIE				
Type									RW			RW				
Reset									0			0				

Bit(s)	Mnemonic	Name	Description
7	MULCOMEN	MULCOMEN	MUI-Communication enable 0: disabled 1: enabled
4	IDLEIE	IDLEIE	IDLE interrupt enable 0: disabled 1: enabled

0x5c UARTn_LINCR LIN control register 0000

Bit	32~15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									LINEN	LBRKIE	LBRKDL	SDBRK	LABAUDEN			
Type									RW	RW	RW	RW	RW			
Reset									0	0	0	0	0			

Bit(s)	Mnemonic	Name	Description
7	LINEN	LINEN	LIN Mode enable 0: disable 1: enable
6	LBRKIE	LBRKIE	LIN Break character detect interrupt enable 0: disable 1: enable
5	LBRKDL	LBRKDL	LIN Mode break detect length 0: 10 bits 1: 11bits
4	SDBRK	SDBRK	LIN Mode whether transmit 13 0s Note: set by software and cleared by MCU internal hardware during the stop bit of the break . 0: disable 1: enable transmit 13 '0's right now
3	LABAUDEN	LABAUDEN	LABAUDEN 0: 0x55 not used to auto baud rate detection 1: 0x55 used to auto baud rate detection

10 ADC

10.1 ADC introduction

Analog to digital converter (ADC) is a system used for translating the analog quantities, which are characteristic of most phenomena in the real world, to digital language, used in information processing, computing, data transmission, and control systems. ADC can operate under 2 power modes: normal mode and low power mode.

10.2 ADC features

- ADC channel input voltage range: $AVSS < V_{in} < AVDD$.
- Maximum conversion rate: 500K at maximum ADC operation clock.
- 18 channels with sample time configurable for each: 16 external channels, 2 internal channels (T-sensor, Bandgap).
- Conversion sequence classified as regular group and injection group.
 - Regular group: configurable maximum 16 channels.
 - Injection group: configurable maximum 4 channels.
- 8 operation modes (called mode x for convenience, x=1~8):
 - Single regular group channel single conversion (mode1).
 - Single regular group channel continuous conversion (mode2).
 - Multiple regular group channels single scan with injection trigger (mode3).
 - Multiple regular group channels single scan with automatically injection (mode4).
 - Multiple regular group channels continuous scan with injection trigger (mode5).
 - Multiple regular group channels continuous scan with automatically injection (mode6).
 - Multiple regular group channels under discontinuous conversion mode (mode7).
 - Multiple injection group channels under discontinuous conversion mode(mode8).
- Start ADC by internal software trigger or external hardware trigger.
- Analog monitor function:
 - Checking none, single or all channels voltage by configuration.
 - Monitor whether the channel voltage is less than low threshold or not less than high threshold.
 - If ADC operates under low power mode, configuration should refer to the table.
- Interrupts:
 - End Of Conversion (EOC) for regular or injection group.
 - End Of Injection Group Conversion (IEOC).
 - Analog Monitor event (AMO).
- DMA access: just for regular group channels.

10.3 ADC functional description

Figure 10-1 below shows the ADC block diagram:

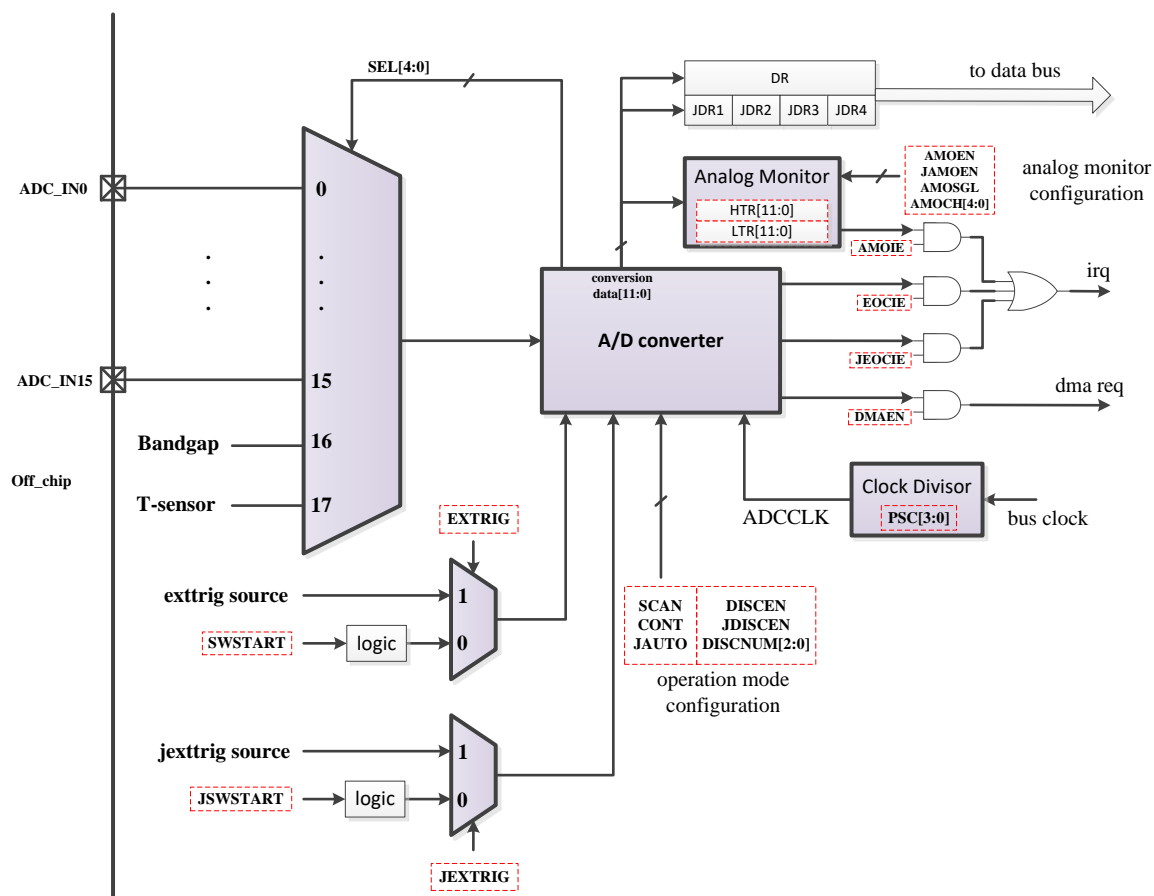


Figure 10-1 UART block diagram

In block diagram, ADC is mainly composed of ADC converter unit, input channel selector, clock divisor, and analog monitor and so on. As illustrated in [Figure 10-1](#), A/D converter unit operates at ADC clock, called ADCCLK for short and the other circuit parts operate at bus clock, called BCLK for short. A typical operation flow is introduced in the following paragraph.

Firstly, ADC should be powered on, which will be described in [10.3.1](#). And then, ADC can be triggered to start by the internal SWSTART or external trigger source, which is derived from the CTU module. After trigger, ADC converter unit starts to work and sends out the selecting signal to input channel selector to select the desired channel one by one based on the regular or injection group channels sequence. After one channel has finished the conversion, the conversion result is stored into the RDR or IDR_x based on which group the current conversion channel belongs to. Meanwhile, the analog monitor starts to work and the related statuses flag appear if the corresponding event occurs. Until now, a single channel conversion flow goes to the end. To point out, there exist some differences for different operation mode and detailed information will be illustrated later.

10.3.1 ADC power on sequence

Before starting all the functions, ADC should be powered on at first. Then a valid trigger can start the ADC to operate based on configured mode. The power on sequence is illustrated as follows.

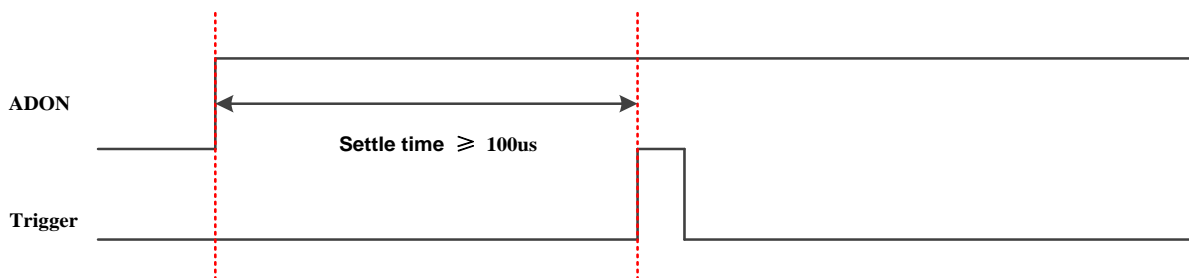


Figure 10-2 ADC Power on sequence

As illustrated in the [Figure 10-2](#), bit ADC_CTRL2[ADON] is set to 1 to control the power on process. After the ADON is set to 1, the settle time for A/D Converter unit power on is not less than 100us, which is the basic configuration.

10.3.2 ADC power modes

For ADC, there are two power modes available. One is the normal mode, the other is low power mode. In [Table 10-1](#), the ADC clock is lower at low power mode in order to get lower power. The low power mode for ADC is used when CPU runs into sleep mode by WFI instruction. The ADC analog monitor event under low power mode can wake up the CPU from sleep mode to normal mode. The normal mode for ADC is used when CPU runs at normal mode and all the ADC registers and functions can be achieved.

Table 10-1 Power modes

Modes	Normal mode	Low power mode
ADC clock (ADCCLK)	0.1 MHz to 10 MHz	0.1 MHz to 1.6 MHz
channel	0 to 17	0 to 17

To explain the ADC low power mode clearly, a simple flow for CPU operation mode switching is showed as [Figure 10-3](#). For example, user can let the CPU into sleep mode by WFI instruction if there is no need to make the CPU operate at normal mode and low power is an urgent requirement. And a desired condition that an analog monitor event occurs will wake up the CPU to work at normal mode.

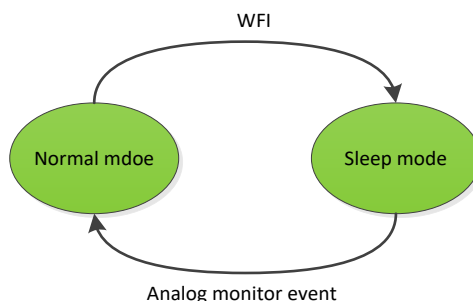


Figure 10-3 CPU normal mode and sleep mode condition

10.3.3 ADC operation modes

This chapter is the key section that introduces the detailed information for different ADC operation modes. Different modes can be used flexibly based on the practical application. After power on and a valid trigger, ADC operates at one of the following modes.

Table 10-2 ADC operation modes and its corresponding configuration

ADC operation mode	MODE_BITS	Trigger source	Channel
mode1	5'b0000x	regular trigger	single regular group channel
mode2	5'b0100x	regular trigger	single regular group channel
mode3	5'b10000	regular trigger (+injection trigger)	multiple regular group channels (+injection group channels)
mode4	5'b10001	regular trigger	multiple regular group channels + injection group channels
mode5	5'b11000	regular trigger (+injection trigger)	multiple regular group channels (+injection group channels)
mode6	5'b11001	regular trigger	multiple regular group channels + injection group channels
mode7	5'b1x10x	regular trigger	regular group channels
mode8	5'b1x01x	injection trigger	injection group channels

Note: $MODE_BITS = \{SCAN, CONT, DISCEN, JDISEN, JAUTO\}$.

Before describing each mode operation flow, it's necessary to introducing some terminologies, such as regular group, injection group and so on. For ADC input channels, they are called ch0 ~ ch17, of which ch0 ~ ch15 are the external input channels, ch16 is corresponding to thermal sensor channel, and ch17 represents the bandgap reference voltage channel.

Regular group is the input channels arranged to convert in order. Based on register ADC_RSQR1, ADC_RSQR2 and ADC_RSQR3, the regular group is composed of maximum 16 channels in sequence from RSQ1 to RSQ16.

For example, if the RSQ1 ~ RSQ16 are set to 9, 8,16,1,5,4,7,3,17,2,0,0,0,17,6,15 respectively, a regular group is arranged as [Figure 10-4](#).

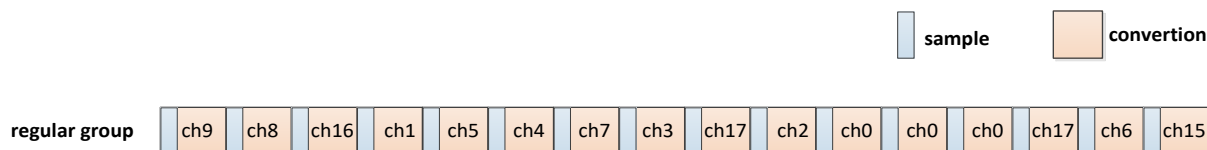


Figure 10-4 Regular group sequence

If the RSQL is set to 13, the last 3 channels will be invalid and not be converted. So, valid regular group sequence is illustrated as [Figure 10-5](#).

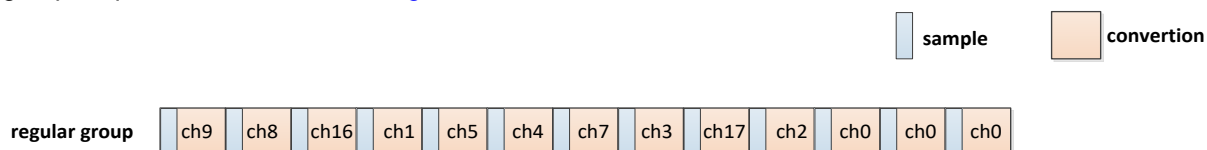


Figure 10-5 Valid regular group sequence

In the same way, injection group is the input channels arranged to convert in order. Based on register ADC_ISQR, the injection group is composed of maximum 4 channels in sequence from ISQ1 to ISQ4. For example, if the ISQ1 ~ ISQ4 are set to 16, 7, 17, 2 respectively, an injection group is arranged as [Figure 10-6](#).

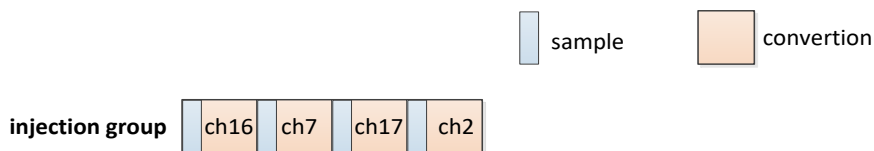


Figure 10-6 Injection group sequence

If the ISQL is set to 3, the last 1 channel will be invalid and not be converted. So, valid injection group sequence is illustrated as [Figure 10-7](#).

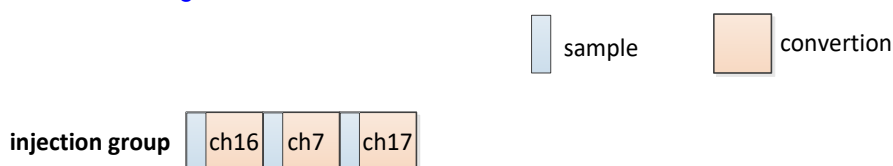


Figure 10-7 Valid injection group sequence

Obviously, regular group trigger and injection group trigger are the corresponding signals to start conversion of the regular and injection group sequence. The trigger is derived from the internal SWSTART or external trigger source illustrated in the ADC block diagram. And the regular trigger is invalid when ADC is in the process of regular group channel conversion. Based on the basic introduction, detailed information for each mode is described as follows. Meanwhile, the [Figure 10-4](#) and [Figure 10-6](#) will be used to explain the operation flow for each mode.

10.3.3.1 Mode 1

This mode just aims at the first channel in regular group no matter what the RSQL is. After the mode is configured as [Table 10-2](#), a valid trigger can make the ADC work at this mode.

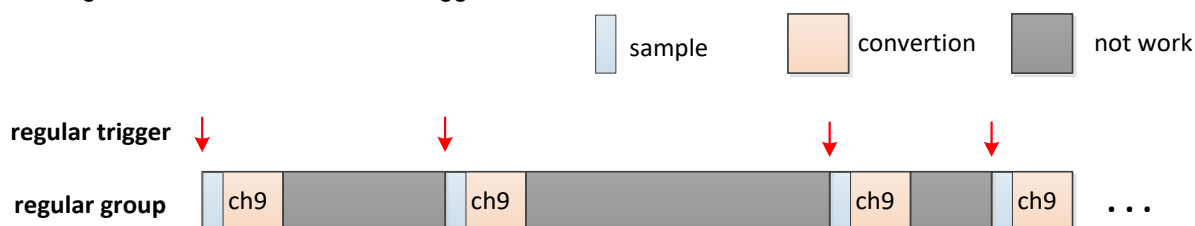


Figure 10-8 Mode 1 operation flow

As the [Figure 10-8](#) shows, the first channel in regular group is converted once after a valid regular trigger. Then the ADC goes to idle until the next valid regular trigger for the next conversion.

10.3.3.2 Mode 2

This mode also just aims at the first channel in regular group no matter what the RSQL is. After the mode configuration as [Table 10-2](#), a valid trigger can make the ADC work at this mode.

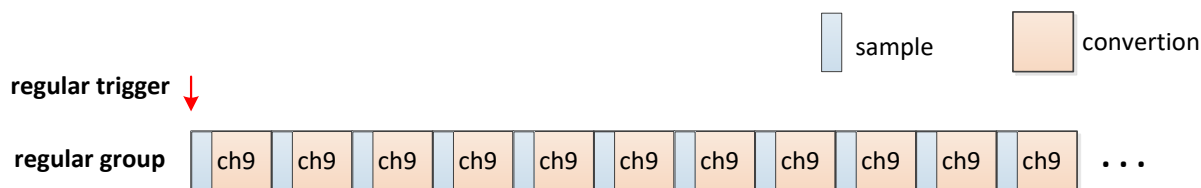


Figure 10-9 Mode 2 operation flow

As the [Figure 10-9](#) shows, the first channel in regular group is converted forever except power down, top reset or mode change after a valid regular trigger.

10.3.3.3 Mode 3

This mode aims at the regular group channels and injection group channels, which must be triggered by injection trigger. The valid regular and injection group channel number are decided by RSQL and ISQL respectively. With the mode configuration as [Table 10-2](#), a valid trigger can make the ADC work at this mode. For example, RSQL is set to 7 and ISQL is set to 3. A typical operation shows as [Figure 10-10](#). The initial regular trigger starts conversion of the first 7 channels in group. When ADC converts the channel 1 in regular group, an injection trigger switches the conversion to 3 injection group channels after the end of channel 1 conversion. After 3 injection group channels has been converted completely, conversion switches back to regular group channels automatically and the last 3 channels start to convert. When finishing the valid regular channels conversion, ADC runs to idle until the next trigger.

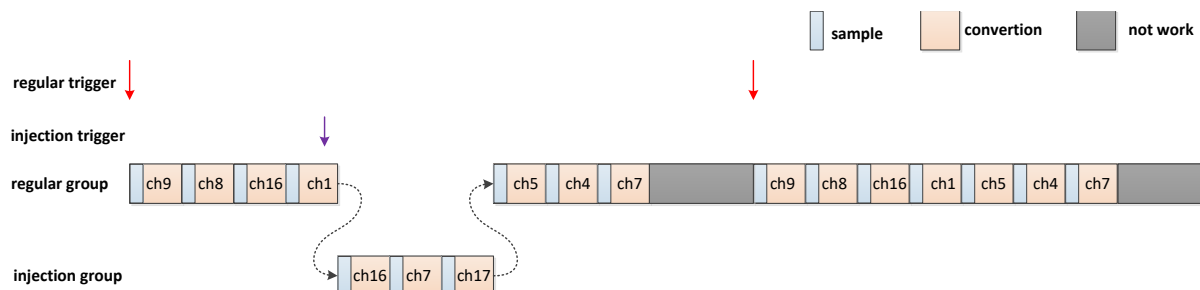


Figure 10-10 Mode 3 typical operation flow

Especially, if the injection trigger occurs when the ADC is idle, the ADC will finish the conversion of the valid injection group channels as the following [Figure 10-11](#). Basically, the regular or injection group channels are converted once by a valid trigger at this mode.

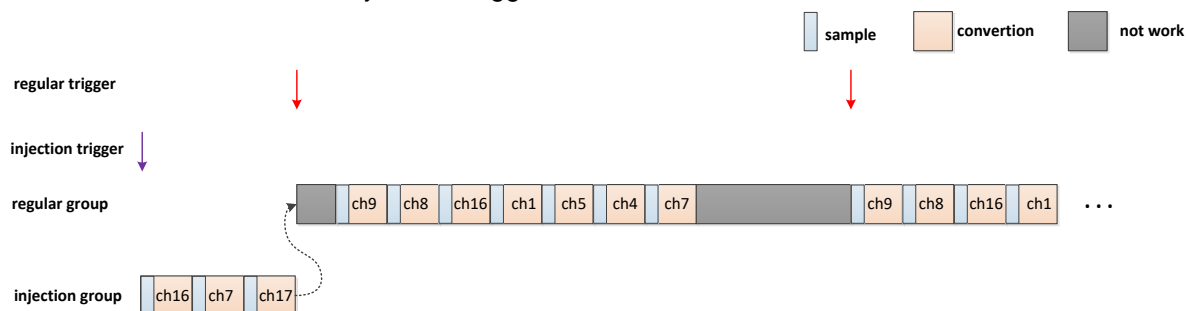


Figure 10-11 Mode 3 operation flow with injection trigger at ADC idle state

10.3.3.4 Mode 4

This mode aims at a single conversion for all the regular group channels and injection group channels, which are converted following the regular group channel automatically. The valid regular channels and injection group channels are decided by RSQL and ISQL respectively. With the mode configuration as [Table 10-2](#), a valid trigger can make the ADC work at this mode. For example, SQL is set to 7 and ISQL is set to 3. A typical operation shows as [Figure 10-12](#). A regular trigger starts conversion of the first 7 regular group channels followed by 3 injection group channels automatically. After the total 10 channels has been converted completely, ADC runs to idle state until the next valid regular trigger.

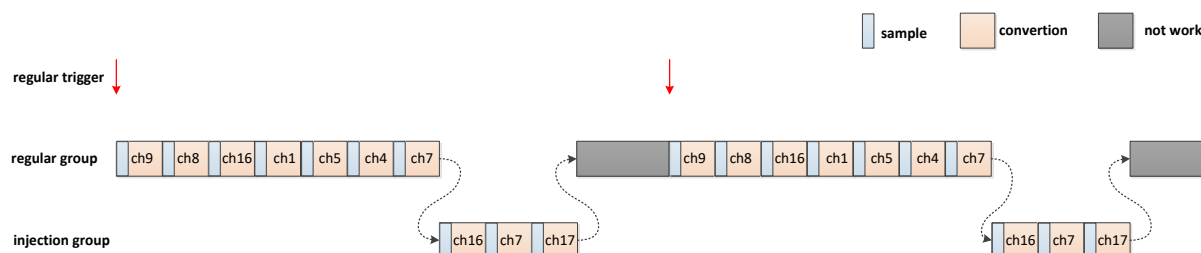


Figure 10-12 Mode 4 operation flow with auto injection trigger

10.3.3.5 Mode 5

This mode just aims at the regular group channels and injection group channels, which must be triggered by injection trigger. The valid regular channels and injection group channels are decided by SQL and ISQL respectively. With the mode configuration as [Table 10-2](#), a valid trigger can make the ADC work all the time except power down, top reset and mode change. For example, RSQL is set to 7 and ISQL is set to 3. A typical operation shows as [Figure 10-13](#). The ADC works on regular group channels in order circularly after a regular trigger or works on the injection group channels if an injection trigger occurs.

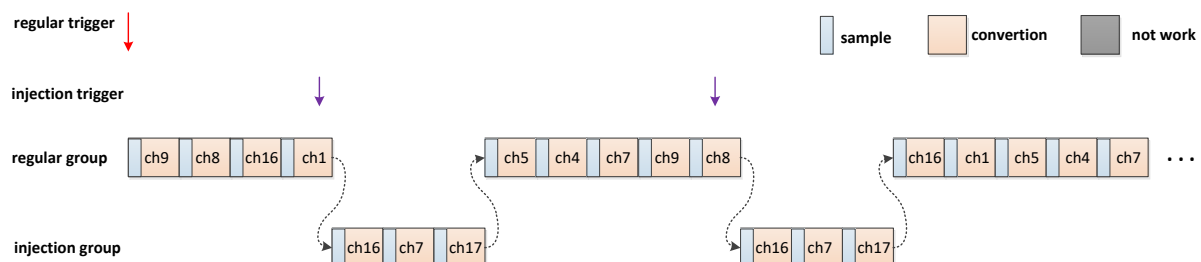


Figure 10-13 Mode 5 typical operation flow

Especially, if the injection trigger occurs when the ADC is idle, the ADC will finish the conversion of the valid injection group channels at first as the following [Figure 10-14](#).

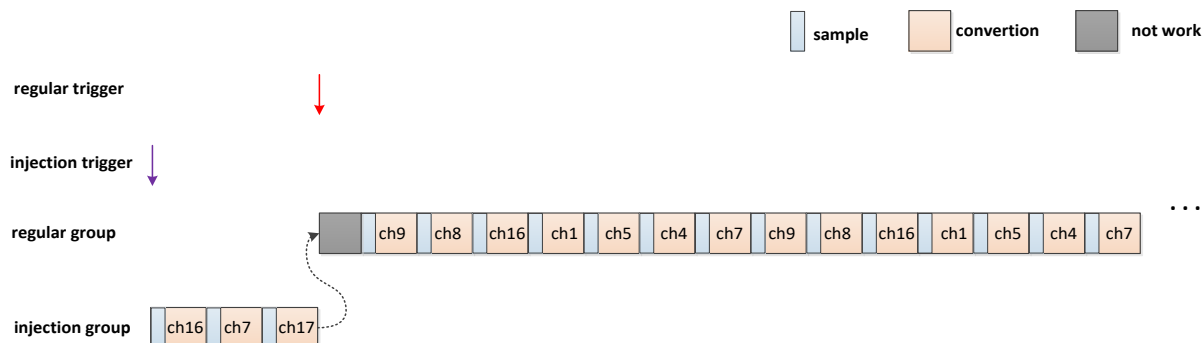


Figure 10-14 Mode 5 operation flow with injection trigger at ADC idle state

10.3.3.6 Mode 6

This mode aims at the regular group channels and injection group channels. The valid regular channels and injection group channels are decided by RSQL and ISQL respectively. With the mode configuration as [Table 10-2](#), a valid regular trigger can make the ADC work at this mode. A key feature for this mode is that a single regular trigger can make the ADC work all the time except power down, top reset and mode change. For example, RSQL is set to 7 and ISQL is set to 3. An operation flow shows as [Figure 10-15](#). The ADC works on regular group channels in order followed by injection group channels circularly after a regular trigger.

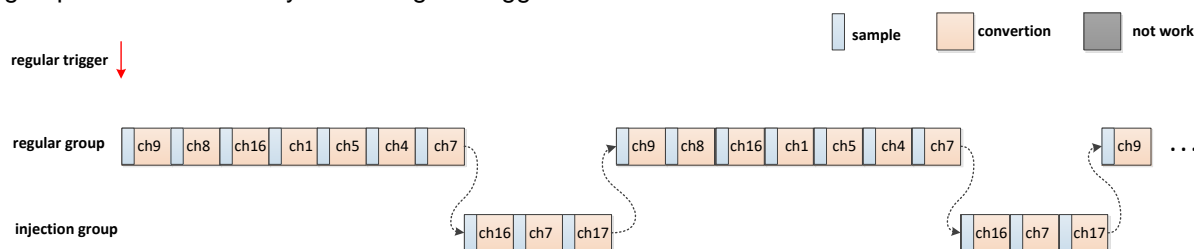


Figure 10-15 Mode 6 operation flow

10.3.3.7 Mode 7

This mode just aims at the regular group channels. The valid regular channels group channels are decided by RSQL. With the mode configuration as [Table 10-2](#), ADC can work at this mode. The valid regular channels are divided into several subgroups every DISCNUM channels.

For example, RSQL is set to 7 and DISCNUM is set to 2.

First regular trigger: ch9, ch8;

Second regular trigger: ch16, ch1;

Third regular trigger: ch5, ch4;

Fourth regular trigger: ch7, generates EOC flag;

Fifth regular trigger: ch9, ch8;

Sixth regular trigger: ch16, ch1;

...

So, the practical conversion flow is illustrated as [Figure 10-16](#).

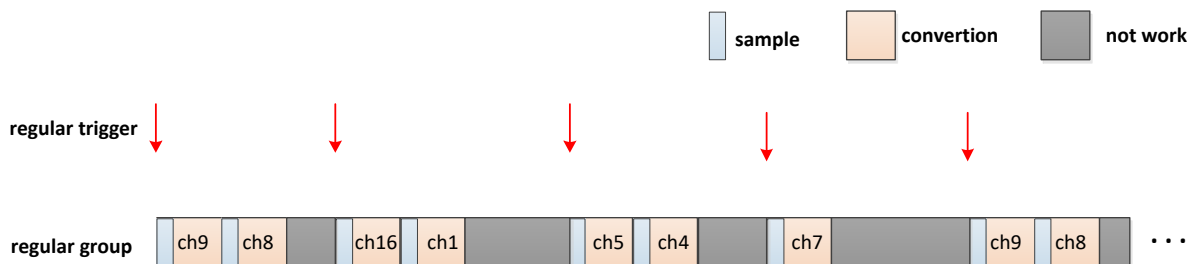


Figure 10-16 Mode 7 operation flow

10.3.3.8 Mode 8

This mode just aims at the injection group channels. The valid injection channels group channels are decided by ISQL. With the mode configuration as [Table 10-2](#), ADC can work at this mode. The valid injection channels are divided into several subgroups every one channel.

For example, ISQL is set to 3.

First regular trigger: ch16;

Second regular trigger: ch7;

Third regular trigger: ch17, generates EOC and IEOC flag;

Fourth regular trigger: ch16;

Fifth regular trigger: ch9, ch7;

...

So, the practical conversion flow is illustrated as [Figure 10-17](#).

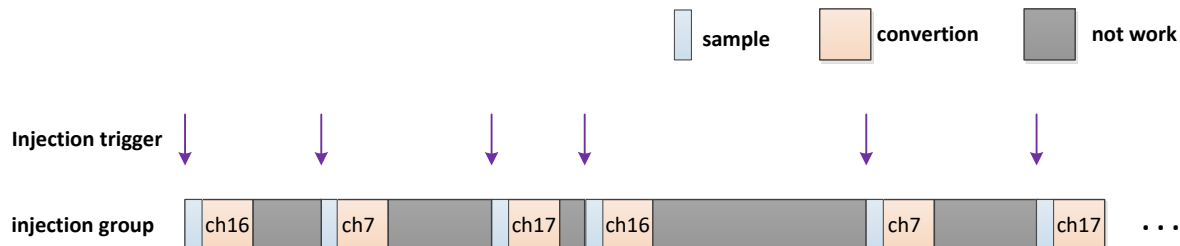


Figure 10-17 Mode 8 operation flow

10.4 Analog monitor

If the monitored channels voltage is more than the high threshold or less than the low threshold, analog monitor set the AMO flag to 1, which is cleared by writing 0 to it. Meanwhile, the interrupt occurs if AMO flag is 1 and AMOIE is configured to 1.

The high and low threshold is decided by AWDH and AWDL respectively. Analog monitor can be used to monitor none, single channel, multiple channels based on bits AMOEN, IAMOEN, AMOSGL and AMOCH.

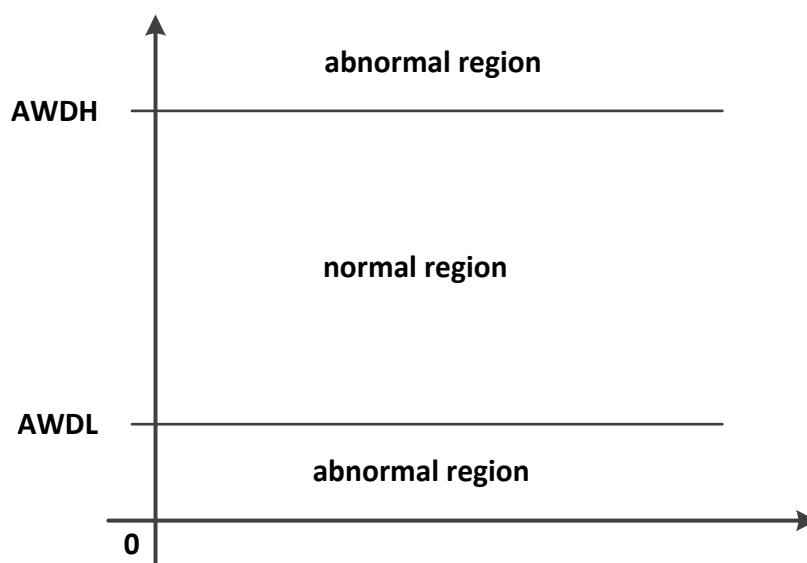


Figure 10-18 Analog monitor detecting region

Table 10-3 Analog monitor configuration

Analog monitor channel	{AMOEN,IAMOEN,AMOSGL}	Configurable operation mode	Comment
None	3'b00x	All modes	-
All injection group channels	3'b010	mode3 ~ mode6, mode8	-
All regular group channels	3'b100	Except mode8	-
All channels	3'b110	All modes	-
Single injection group channel	3'b011	mode3 ~ mode6, mode8	Conversion sequence must contain the injection channel determined by AMDCH[4:0].
Single regular group channel	3'b101	mode1 ~ mode7	Conversion sequence must contain the regular channel determined by AMDCH[4:0].
Single regular or injection group channel	3'b111	All modes	Conversion sequence must contain the regular or injection channel determined by AMDCH[4:0].

10.5 Status flag description

For ADC, there are three flags to indicate the conversion status. One is the EOC, another is IEOC, the last is AMO. The EOC flag indicates the end of the conversion for both regular and injection group channels. The IEOC flag indicates whether all the injection group channels are converted completely. The AMO flag indicates whether the analog monitor event occurs. The analog monitor event is whether the current conversion result is more than high threshold or less than low threshold based on the configuration. The flag EOC and IEOC are generated at different time for different modes, but can be classified to three conditions. The flag AMO is generated at the same time for all the modes. The following descriptions are introduced with the assumption that ch5 is less than AWDL, ch7 is more than the AWDH, and the analog monitor is configured to check all the channels including regular group channels and injection group channels for example.

The flags EOC and IEOC is generated at the same time for mode 1 to mode 6. The flag AMO is generated at the same time for all the 8 modes. The detailed information about three flags is illustrated as [Figure 10-19](#) based on mode 6.

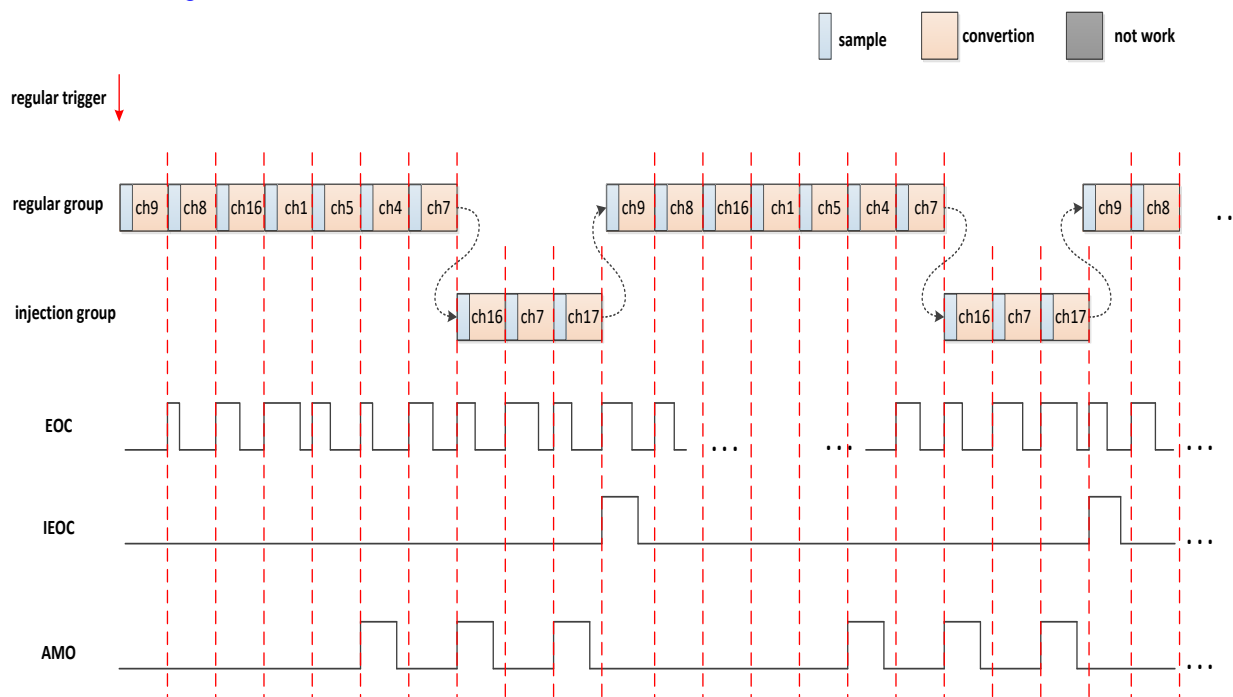


Figure 10-19 Three flags under condition 1

In [Figure 10-19](#), EOC is set to 1 when a channel conversion is finished for both regular and injection group channels. And the EOC is cleared by writing 0 to it or reading the ADC_RDR register. For IEOC flag, it is set to 1 when all the valid injection group channels have been converted completely and cleared by writing 0 to IEOC bit. AMO flag is set to 1 when the channel voltage is out of the analog monitor normal region, such as ch5, ch7. To point out, the duration of the flag maintaining 1 is decided by the CPU response time, which is related with the current loading of the CPU at that moment.

The time when the flags are generated is different between mode7 and mode 1 to mode 6. The detailed information about three flags is illustrated as [Figure 10-20](#).

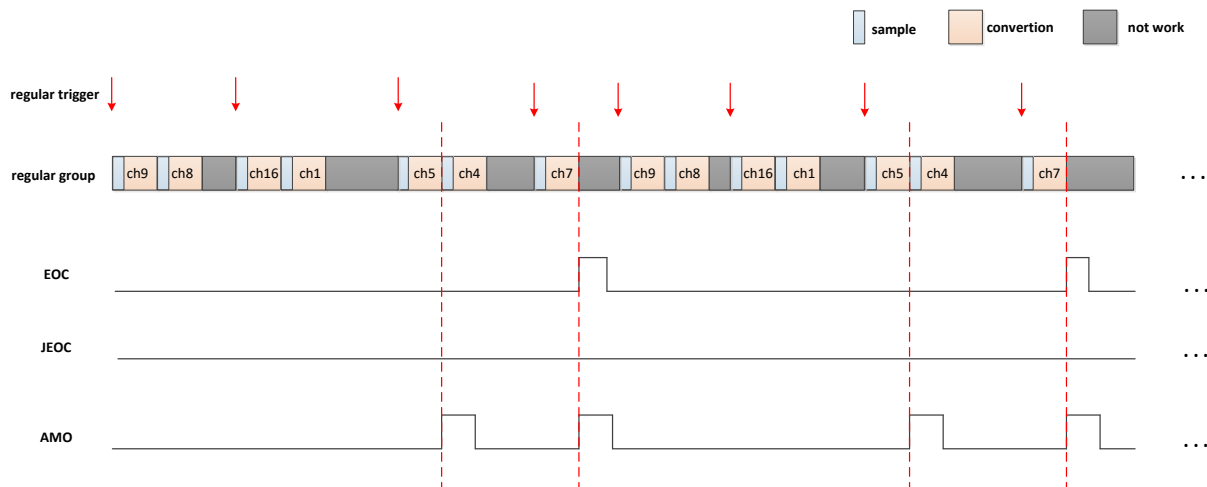


Figure 10-20 Three flags under condition 2

In [Figure 10-20](#), EOC is set to 1 when the channel conversions are finished for regular group channels. And the EOC is cleared by writing 0 to it or reading the ADC_RDR register. For IEOC flag, it is still 0 all the time for this mode. AMO flag is set to 1 when the channel voltage is out of the analog monitor normal region, such as ch5, ch7. To point out, the duration of the flag maintaining 1 is decided by the CPU response time, which is related with the current loading of the CPU at that moment. The flags also show a different scenario under mode 8. It's easy to describe the flag information because ADC conversion just aims at injection group channels.

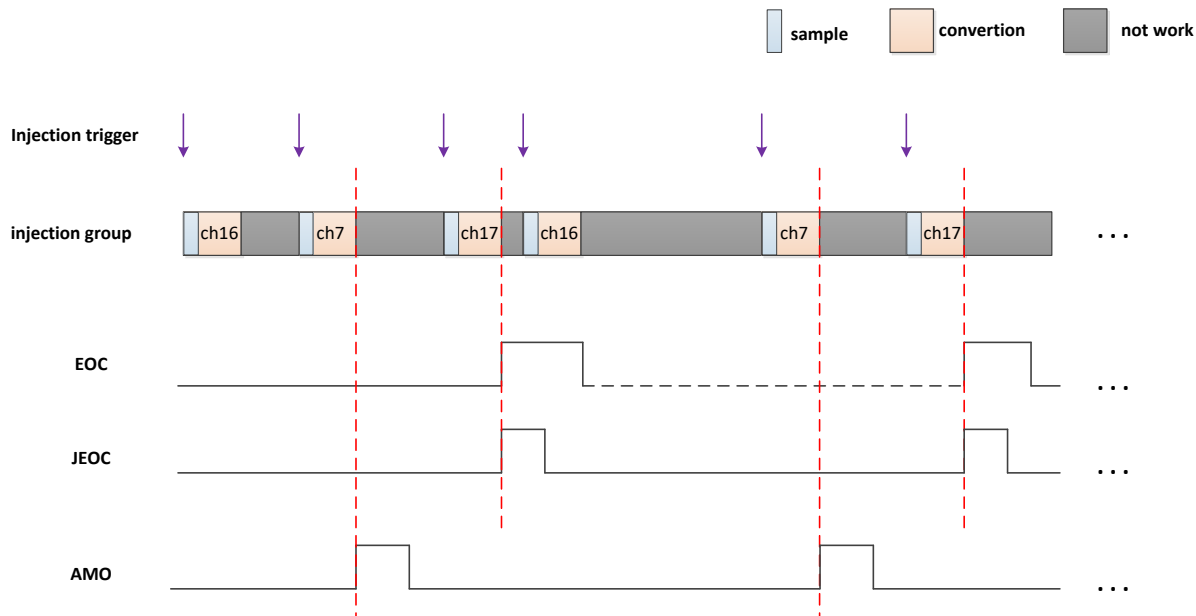


Figure 10-21 Three flags under condition 3

In [Figure 10-21](#), EOC and IEOC are set to 1 when the channel conversions are finished for all the injection group channels. And the EOC is cleared by writing 0 to it or reading the ADC_RDR register. For IEOC flag, it is cleared by writing 0 to IEOC bit. AMO flag is set to 1 when the channel voltage is out of the analog monitor normal region, such as ch7. To point out, the duration of the flag maintaining

1 is decided by the CPU response time, which is related with the current loading of the CPU at that moment.

10.6 Calibration

The calibration function can make the ADC conversion result more accurate and correct the errors caused by GE (Gain Error) and OE (Offset Error).

During chip production, machine testing is required, and the measured GE & OE values are stored in a specific area of the chip. When the calibration function is enabled, the data register gives the final result of using GE & OE for calibration.

10.7 Sampling conversion time

The ADC uses several ADC_CLK cycles to sample the input voltage. The number of sampling cycles can be changed by the SPT [2:0] bits in the ADC_SPT register. Each channel can be sampled at different times.

Total conversion time: (SPT+ 12) * ADC cycles + 5 APB cycles

For example, if APB=48MHz, ADCCLK=8MHz, SPT=3 ADCCLK, total conversion time= (3+12)/8+(5/48) ≈ 1.98μs.

10.8 Temperature sensor

The temperature sensor can be used to measure the ambient temperature (TA) of the device. The temperature sensor is internally connected to the ADC channel which is used to convert the sensor output voltage into a digital value.

The internal temperature sensor is more suited to applications that detect temperature variations instead of absolute temperatures. If accurate temperature readings are needed, an external temperature sensor part should be used.

Temperature using the following formula:

Temperature (°C) = {(VTEMP25 - VSENSE) / Slope} + 25

VTEMP25: VSENSE value for 25°C

VSENSE: VSENSE value

Slope = Average slope for Temperature (mV/° C).

Refer to the Electrical characteristics section for the actual values of VTEMP25 and Slope.

10.9 DMA access

Because of just one data register for regular group channels, DMA function is recommended to avoid conversion result lose when there are many regular group channels for conversion. Therefore, DMA function just is dedicated for regular group channels.

10.10 Program guide

10.10.1 Normal power mode

The basic programming steps are described as three steps, which briefly introduces the configuration process at macro level. And the detailed configuration can be found in the following example.

Basic three steps at macro level:

- First step: ADC operation clock configuration and power on.
- Second step: mode configuration and other configuration.
- Last step: a valid trigger.

For example, ADC works with the following configuration:

- (1) Mode 5, ADC operation clock:8.33MHz @ bus clock = 50MHz.
- (2) Regular sequence: ch9, ch8, ch16, ch1, ch5, ch4, ch7, ch3, ch17, ch2, ch0, ch0, ch0, ch17, ch6, chf.
RSQL=7;
- (3) Injection sequence: ch16, ch7, ch17, ch2.
ISQL=3;
- (4) Analog monitor: for all the channels.
- (5) Others: default.

10.10.2 Low power mode

ADC configuration for low power mode is the same with the normal mode configuration, and ADC enters low power mode and wakes up from low power mode just following the [Figure 10-3](#).

10.11 ADC register definition

Table 10-4 ADC register map

BaseAddress = 0x40003000.

Address = BaseAddress + OffsetAddress.

Offset Address	Name	Width	Register function
00000000	ADC_STR	32	ADC status register
00000004	ADC_CTRL1	32	ADC control register1
00000008	ADC_CTRL2	32	ADC control register2
0000000C	ADC_SPT1	32	ADC sample time selection register 1
00000010	ADC_SPT2	32	ADC sample time selection register 2
00000014	ADC_IOFR1	32	ADC injection group offset register 1
00000018	ADC_IOFR2	32	ADC injection group offset register 2
0000001C	ADC_IOFR3	32	ADC injection group offset register 3
00000020	ADC_IOFR4	32	ADC injection group offset register 4
00000024	AWDH	32	ADC AWD high threshold register

Offset Address	Name	Width	Register function
00000028	AWDL	32	ADC AWD low threshold register
0000002C	ADC_RSQR1	32	ADC regular group sequence configure register 1
00000030	ADC_RSQR2	32	ADC regular group sequence configure register 2
00000034	ADC_RSQR3	32	ADC regular group sequence configure register 3
00000038	ADC_ISQR	32	ADC injection group sequence configure register
0000003C	ADC_IDR1	32	ADC injection group data register 1
00000040	ADC_IDR2	32	ADC injection group data register 2
00000044	ADC_IDR3	32	ADC injection group data register 3
00000048	ADC_IDR4	32	ADC injection group data register 4
0000004C	ADC_RDR	32	ADC regular group data register

00000000 ADC_STR
ADC Status Register
00000000

Bit	31~18							7	6	5	4	3	2	1	0
Name											ADC_IDLE		IEOC	EOC	AMO
Type											R		R	R	R

Bit(s)	Mnemonic	Name	Description
4	ADC_IDLE		ADC idle state indicates (is useful for sleep function) 0: ADC not idle 1: ADC idle
2	IEOC		Injection group conversion completed flag 0: injection group conversion not completed 1: injection group conversion completed, write 0 to clear
1	EOC		Regular group conversion completed flag 0: Regular group conversion not completed 1: Regular group conversion completed, write 0 or read ADC_RDR to clear
0	AMO		Analog monitor event occurs 0: no analog monitor event 1: analog monitor event occurs, write 0 to clear

00000004 **ADC_CTRL1**
ADC Control Register 1

00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	SW STA RT	ISW STA RT								AL IG N	IEX TTR IG	EXT TRIG	D M AE N	AM OIE	IEO CIE	EOC IE
Type										R W	RW	RW	R W	RW	RW	RW
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	SCA N	CO NT	DIS CEN	IDIS CEN	IA UT O	DISCNUM[2:0]			AM OE N	IA M OE N	AM OS GL	AMOC[4:0]				
Type	RW	RW	RW	RW	R W	R W	R W	RW				RW	R W	RW	RW	RW

Bit(s)	Mnemonic	Name	Description
31	SWSTART	SWSTART	Software trigger for regular channels write 1 to trigger; read as 0
30	ISWSTART	ISWSTART	Software trigger for inject channels write 1 to trigger; read as 0
22	ALIGN	ALIGN	Data alignment 0: right alignment. 1: left alignment.
21	IEXTTRIG	IEXTTRIG	Inject group trig source select 1: external 0: internal(software trig)
20	EXTTRIG	EXTTRIG	Regular group trig source select 1: external 0: internal(software trig)
19	DMAEN	DMAEN	DMA function enable 1: enable 0: disable
18 ~ 16	AMOIE,EOCIE,IEOCIE	AMOIE,EOCIE,IEOCIE	Interrupt function enable 1: enable 0: disable
15 ~ 11	Modes control bits	-	ADC operation mode detailed configuration in table Table 10-2
10 ~ 8	DISCNUM	DISCNUM	Discontinuous conversion length of channel 0 ~ 7: decide the subgroup length under mode 7
7 ~ 5	Analog monitor control bits	-	Analog monitor function configuration detailed configuration in Table 10-3 .
4 ~ 0	AMOC	AMOC	Analog monitor detecting channel Specify the monitored channel when analog monitor is configured to detect just single channel

00000008 **ADC_CTRL2**
ADC Control Register 2

0000F002

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

Bit(s)	Mnemonic	Name	Description
15 ~ 12	PSC	PSC	Bus clock(Tbusclk) Prescale to get ADC operation clock(Tadcclk). NOTE: psc value must guarantee that ADC operation clock is less than 10 MHz. 0 ~ 15 : 1 ~ 16 divisor.
0	ADON	ADON	ADC Power on Write 1: ADC power on. Write 0: ADC power down and reset the ADC (but the configure registers are not be reset).

0000000C **ADC_SPT1** **ADC**
Sample Time Register 1

00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name									SPT17[2:0]			SPT16[2:0]			SPT15[2:0]	
Type									RW							
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name		SPT14[2:0]			SPT13[2:0]			SPT12[2:0]			SPT11[2:0]			SPT10[2:0]		
Type	RW															

Note: x=10 to 17

Bit(s)	Mnemonic	Name	Description
23:0	SPTx	SPTx	Sample time selection for each channels 000 to 111: 6/14/29/42/56/72/215/3 ADCCLK.

00000010 **ADC_SPT2**
ADC Sample Time Register 2

00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name			SPT9[2:0]			SPT8[2:0]			SPT7[2:0]			SPT6[2:0]			SPT5[2:0]	
Type			RW													
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name		SPT4[2:0]			SPT3[2:0]			SPT2[2:0]			SPT1[2:0]			SPT0[2:0]		
Type	RW															

Note: x=0 to 9

Bit(s)	Mnemonic	Name	Description
23:0	SPTx	SPT	Sample time selection for each channels 000 to 111: 6/14/29/42/56/72/215/3 ADCCLK.

00000014~20 ADC_IOFRx(x=1~4)

ADC Injection group offset Register

00000000

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

Note: x= 1 to 4

Bit(s)	Mnemonic	Name	Description
11:0	IOFRx	IOFRx	Injection group offset value Final conversion result will be minus by Offset value

00000024 AWDH

ADC AWD High threshold Register

00000000

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

Bit(s)	Mnemonic	Name	Description
11:0	AWDH	AWDH	High threshold value for analog Watch dog Define the high threshold value .

00000028 AWDL

ADC AWD Low Threshold Register

00000000

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

Bit(s)	Mnemonic	Name	Description
11:0	AWDL	AWDL	Low threshold value for analog Watch dog Define the low threshold value .

0000002C ADC_RSQR1

ADC regular group sequence configure register 1

00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name									RSQL[3:0]				RSQ16[4:0]			
Type																
Bit	15	14	13	12	11	10	9	8								
Name	RSQ15[4:0]				RSQ14[4:0]								RSQ13[4:0]			
Type																

Bit(s)	Mnemonic	Name	Description
23:20	RSQL	RSQL	Length of the regular group Note: length must be less than the number of actual valid regular group sequence 0 ~ 15:define the regular group length.
19:0	RSQx	RSQx	Channel selection for regular group 0~15: external channels 16: BG voltage 17: Thermal sensor voltage

00000030 ADC_RSQR2

ADC regular group sequence configure register 2

00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
Name			RSQ12[4:0]						RSQ11[4:0]						RSQ10[4:0]			
Type			RW															
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Name			RSQ9[4:0]						RSQ8[4:0]						RSQ7[4:0]			
Type	RW																	

Bit(s)	Mnemonic	Name	Description
29:0	RSQx	RSQx	Channel selection for regular group 0~15: external channels 16: BG voltage 17: Thermal sensor voltage

00000034 ADC_RSQR3

ADC regular group sequence configure register 3

00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
Name			RSQ6[4:0]						RSQ5[4:0]						RSQ4[4:0]			
Type			RW															
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Name			RSQ3[4:0]						RSQ2[4:0]						RSQ1[4:0]			
Type			RW															

Bit(s)	Mnemonic	Name	Description
29:0	RSQx	RSQx	Channel selection for regular group 0~15: external channels 16: BG voltage 17: Thermal sensor voltage

00000038 ADC_ISQR ADC injection group sequence configure register

00000000

AutoChips Confidential

© 2013 - 2021 AutoChips Inc.

Page 140 of 360

This document contains information that is proprietary to AutoChips Inc.
 Unauthorized reproduction or disclosure of this information in whole or in part is strictly prohibited.

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name											ISQL[1:0]		ISQ4[4:0]			
Type																
Bit	15	14	13	12	11	10	9	8	7	6						
Name		ISQ3[4:0]					ISQ2[4:0]					ISQ1[4:0]				
Type	RW															

Bit(s)	Mnemonic	Name	Description
21:20	ISQL	ISQL	Length of the injection group <i>Note: The length must be less than the number of actual valid injection group sequence.</i> 0 ~ 15: define the regular group length.
19:0	ISQx	ISQx	Channel selection for regular group 0~15: external channels 16: BG voltage 17: Thermal sensor voltage

0000003C~48 ADC IDR_x(x=1~4)

ADC Injection group data register

00000000

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

Note: x=1 to 4

Bit(s)	Mnemonic	Name	Description
11:0	IDRx	IDR	Data registers for injection group <i>Note: there are 4 data registers for injection group.</i>

0000004C ADC RDR

ADC Regular group data register

00000000

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

Bit(s)	Mnemonic	Name	Description
11:0	RDR	RDR	Data register for regular group <i>Note: there is just one register for regular group. If ADC is operating at high speed and the cpu can't handle in time, so user must open the DMA function for ADC to avoid the data from being lost.</i>

11 ACMP

11.1 Introduction

The ACMP module includes ACMP0 and ACMP1. Both ACMP0 and ACMP1 contain a comparator and a 6-bit DAC. The analog MUX provides a circuit for selecting an analog input signal from six channels. One channel provided by the 6-bit DAC, and others by external input.

The polling mode and hall output function of ACMP0 are designed for motor application. ACMP1 is a normal module without polling and hall output.

11.2 Features

- On-chip 6-bit resolution DAC with selectable reference voltage from VDD or internal bandgap.
- Configurable hysteresis.
- Selectable interrupt on rising edge, falling edge, or both rising or falling edges of comparator output.
- Up to six selectable comparator inputs.
- Support low power mode wakeup.
- Support CTU trigger.
- ACMP0 support polling mode.
- ACMP0 support hall output.

11.2.1 Block diagram

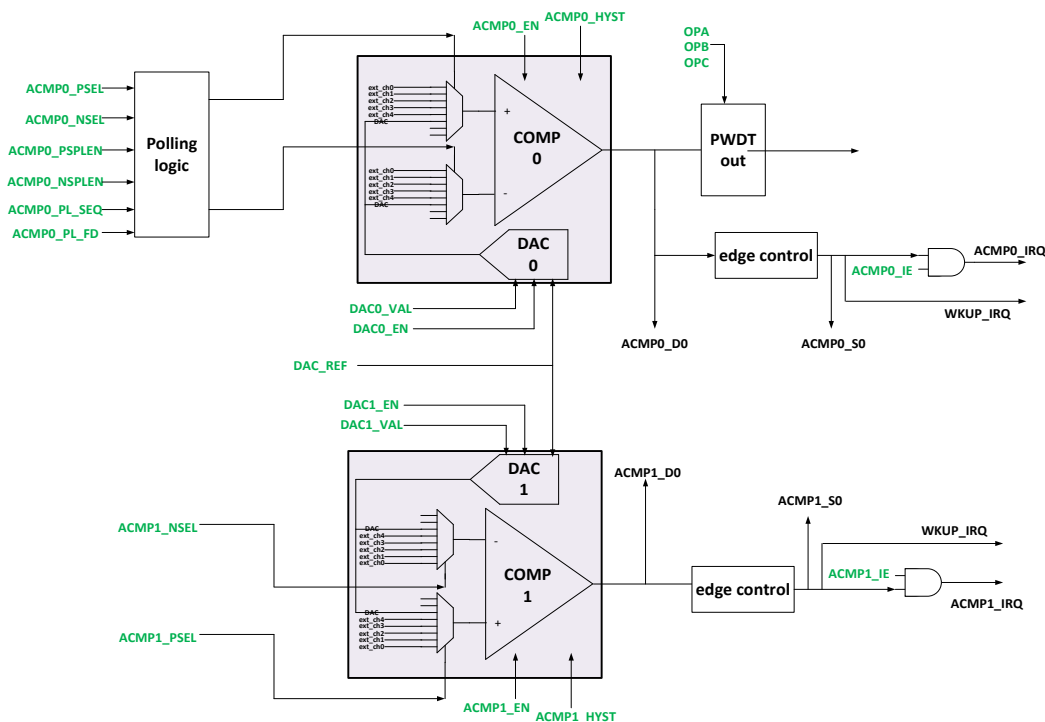


Figure 11-1 ACMP block diagram

11.3 Functional description

The ACMP0/ACMP1 module is functionally composed of two parts: digital-to-analog (DAC) and comparator (CMP).

The DAC includes a 64-level DAC (digital to analog converter) and relevant control logic. DAC can select one of two reference inputs, Vdd or on-chip bandgap, as the DAC input V_{in} by setting DAC_REF. After the DAC is enabled, it converts the data set in DAC_VAL to a stepped analog output, which is fed into ACMP as an internal reference input.

The ACMP0/ACMP1 can achieve the analog comparison between positive input and negative input, and then give out a digital output and relevant interrupt. Both the positive input and negative input of analog comparator can be selected from the six common inputs: five external reference inputs and one internal reference input from the DAC output.

11.3.1 ACMP0

11.3.1.1 Normal mode

After the ACMP0 is enabled by setting ACMP0_CR0[ACMP0_EN], the comparison result appears as a digital output. Whenever a valid edge defined in ACMP0_CR0[ACMP0_MOD] occurs, ACMP0_SR[ACMP0_F] is asserted. If ACMP0_CR0[ACMP0_IE] is set, a CPU interrupt occurs.

The ACMP0 output is synchronized by the bus clock to generate ACMP0_DR[ACMP0_O] so that CPU can read the comparison. ACMP0_DR[ACMP0_O] changes following the comparison result, so it can serve as a tracking flag that continuously indicates the voltage delta on the inputs.

11.3.1.2 Polling mode

ACMP0 can switch the input channel for comparator's positive or negative input. The switch sequence is defined in ACMP0_CR4[ACMP0_PL_SEQ], and the frequency is controlled by ACMP0_FD[ACMP0_PL_FD]. ACMP0_PS_PLEN and ACMP0_NS_PLEN is the enable bit of polling mode. ACMP0_PS_PLEN and ACMP0_NS_PLEN can't be enabled simultaneously. Both ACMP0_PS_PLEN and ACMP0_NS_PLEN enabled will not trigger the polling mode. So software must ensure that one of the two fields above is enabled.

This page gives an example of polling mode now. ACMP0 positive input polling, polling frequency is source_clk/100, external channel 1-4 and DAC out act polling, ACMP0 negative input selects external input 0, falling edge trigger interrupt.

Step1: ACMP0_IE = 1'b1, ACMP0_MOD = 2'b01.

Step2: set DAC0_VAL, DAC0_EN = 1'b1.

Step3: ACMP0_PS_PLEN = 1'b1, ACMP0_NS_PLEN = 1'b0.

Step4: ACMP0_PL_FD = 2'b01, ACMP0_PL_SEQ = 6'b111110.

Step5: ACMP0_NSEL = 3'b000.

Step6: ACMP0_EN = 1'b1.

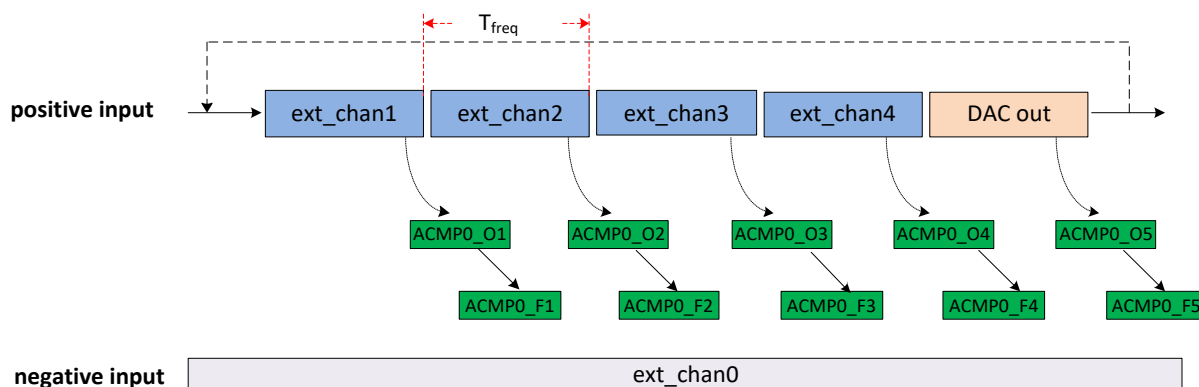


Figure 11-2 Polling mode

11.3.1.3 HALL output mode

ACMP0 has three hall outputs, `acmp0_out_pwt_a`, `acmp0_out_pwt_b` and `acmp0_out_pwt_c`. These signals are connected to PWD module inside the chip. These hall outputs complete the motor control cooperated with the polling function. Every hall output can be selected one of the six channels with polling mode.

For example, if polling mode `ACMP0_PL_SEQ = 6'b001110`. The polling sequence is external input 1 -> external input 2 -> external input 3 (channel 1 -> channel 2 -> channel 3).

Set `ACMP0_OPA[ACMP0_OPA_SEL] = 3'b010`, then the `acmp0_out_pwt_a` is the `ACMP0_DO[ACMP0_O3]`.

11.3.2 ACMP1

ACMP1 is normal function module without polling mode and hall out similar to ACMP0.

11.3.3 Low power wakeup

When in low power mode, a valid edge on ACMP output generates an asynchronous interrupt that can wake the MCU from low power mode. This interrupt can be cleared by writing a 1 to the `ACMP0_WUF/ACMP1_WUF`. Note that the wakeup function is valid for normal mode only.

Note: When Using ACMP to Wakeup MCU, it is recommended to select rising edge. Since the ACMP clock is disabled in the MCU low power mode to save power consumption, it is sensitive to external signals and the MCU will be woken up incorrectly if there is noise in the external signal when the falling edge is used.

11.4 Memory map and register definition

Table 11-1 ACMP register map

Module name: ACMP Base address: (+40005000h)

Address	Name	Width	Register Function
40005000	ACMP0_CR0	32	ACMP0 configuration register 0
40005004	ACMP0_CR1	32	ACMP0 configuration register 1
40005008	ACMP0_CR2	32	ACMP0 configuration register 2

4000500C	ACMP0_CR3	32	ACMP0 configuration register 3
40005010	ACMP0_CR4	32	ACMP0 configuration register 4
40005014	ACMP0_DR	32	ACMP0 data output register 0
40005018	ACMP0_SR	32	ACMP0 status register 0
4000501C	ACMP0_FD	32	ACMP0 polling frequency divider register
40005020	ACMP0_OPA	32	ACMP0 hall output A set register
40005024	ACMP0_OPB	32	ACMP0 hall output B set register
40005028	ACMP0_OPC	32	ACMP0 hall output C set register
4000502C	ACMP_DACSR	32	ACMP DAC reference select register
40005030	ACMP1_CR0	32	ACMP1 configuration register 0
40005034	ACMP1_CR1	32	ACMP1 configuration register 1
40005038	ACMP1_CR2	32	ACMP1 configuration register 2
4000503C	ACMP1_DSR	32	ACMP1 data and status register
40008820	ACMPDAC_CFG	32	ACMPDAC CFG0 register

40005000 ACMP0_CR0 ACMP0 configuration register 0 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									ACMP0_EN	ACMP0_HYST		ACMP0_IE		ACMP0_OPE	ACMP0_MOD	
Type									RW	RW		RW		RW	RW	
Reset									0	0		0		0	0	0

Bit(s)	Name	Description
7	ACMP0_EN	ACMP0 enable 0: disable 1: enable
6	ACMP0_HYST	analog comparator 0 hysteresis selection 0: 10mV 1: 20mV
4	ACMP0_IE	ACMP0 interrupt enable 0: disable 1: enable
2	ACMP0_OPE	ACMP0 hall output enable 0: disable 1: enable
1:0	ACMP0_MOD	determines the ACMP0 sensitivity modes of the interrupt trigger 00: ACMP0 interrupt on output falling edge. 01: ACMP0 interrupt on output rising edge. 10: ACMP0 interrupt on output falling edge. 11: ACMP0 interrupt on output falling or rising edge.

40005004 ACMP0_CR1 ACMP0 configuration register 1 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name										ACMP0_PSEL				ACMP0_NSEL		
Type										RW				RW		
Reset										0	0	0		0	0	0

Bit(s)	Name	Description
6:4	ACMP0_PSEL	ACMP0 positive input select 000: external input 0 001: external input 1 010: external input 2 011: external input 3 100: external input 4 101: DAC0 output 110: ACMP0 output tie 0 111: ACMP0 output tie 0
2:0	ACMP0_NSEL	ACMP0 negative input select 000: external input 0 001: external input 1 010: external input 2 011: external input 3 100: external input 4 101: DAC0 output 110: ACMP0 output tie 0 111: ACMP0 output tie 0

40005008 ACMP0_CR2 ACMP0 configuration register 2 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name										DAC0_EN		DAC0_VAL				
Type										RW		RW				
Reset										0		0	0	0	0	0

Bit(s)	Name	Description
7	DAC0_EN	DAC0 enable 0: disable 1: enable
5:0	DAC0_VAL	DAC0 output level selection

4000500C ACMP0_CR3 ACMP0 configuration register 3 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									ACMP0_PSPL_EN				ACMP0_NSPL_EN			
Type									RW				RW			
Reset									0				0			

Bit(s)	Name	Description
7	ACMP0_PSPL_EN	ACMP0 positive input polling mode enable 0: disable 1: enable
3	ACMP0_NSPL_EN	ACMP0 negative input polling mode enable 0: disable 1: enable

40005010 ACMP0_CR4 ACMP0 configuration register 4 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name											ACMP0_PL_SEQ					
Type											RW					
Reset											0	0	0	0	0	0

Bit(s)	Name	Description
5:0	ACMP0_PL_SEQ	ACMP0 polling channel sequence set 0: disable corresponding channel 1: enable corresponding channel

40005014 ACMP0_DR ACMP0 data output register 0 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									ACMP0_O		ACMP0_O5	ACMP0_O4	ACMP0_O3	ACMP0_O2	ACMP0_O1	ACMP0_O0
Type									RO		RO	RO	RO	RO	RO	RO
Reset									0		0	0	0	0	0	0

Bit(s)	Name	Description
7	ACMP0_O	ACMP0 normal mode output
5	ACMP0_O5	ACMP0 polling mode channel 5 output
4	ACMP0_O4	ACMP0 polling mode channel 4 output

Bit(s)	Name	Description
3	ACMP0_O3	ACMP0 polling mode channel 3 output
2	ACMP0_O2	ACMP0 polling mode channel 2 output
1	ACMP0_O1	ACMP0 polling mode channel 1 output
0	ACMP0_O0	ACMP0 polling mode channel 0 output

40005018 ACMP0_SR ACMP0 status register 0 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									ACMP0_F	ACMP0_WPF	ACMP0_F5	ACMP0_F4	ACMP0_F3	ACMP0_F2	ACMP0_F1	ACMP0_F0
Type									RW	RW	RW	RW	RW	RW	RW	RW
Reset									0	0	0	0	0	0	0	0

Bit(s)	Name	Description
7	ACMP0_F	ACMP0 normal mode interrupt flag write 1 clear
6	ACMP0_WPF	ACMP0 low power mode wakeup interrupt flag write 1 clear
5	ACMP0_F5	ACMP0 polling mode channel 5 interrupt flag write 1 clear
4	ACMP0_F4	ACMP0 polling mode channel 4 interrupt flag write 1 clear
3	ACMP0_F3	ACMP0 polling mode channel 3 interrupt flag write 1 clear
2	ACMP0_F2	ACMP0 polling mode channel 2 interrupt flag write 1 clear
1	ACMP0_F1	ACMP0 polling mode channel 1 interrupt flag write 1 clear
0	ACMP0_F0	ACMP0 polling mode channel 0 interrupt flag write 1 clear

4000501C ACMP0_FD ACMP0 polling frequency divider register 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name																ACMP0_PL_FD
Type																RW
Reset															0	0

Bit(s)	Name	Description
1:0	ACMP0_PL_FD	ACMP0 polling mode frequency divider this divider controls the switch frequency of polling channels 00: source_clk/256 01: source_clk/100

Bit(s)	Name	Description
10:	source_clk/70	
11:	source_clk/50	

40005020 ACMP0_OPA ACMP0 hall output A set register 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name														ACMP0_OPA_SEL		
Type														RW		
Reset														0	0	0

Bit(s)	Name	Description
2:0	ACMP0_OPA_SEL	ACMP0 hall output A set 000: polling channel 0 001: polling channel 1 010: polling channel 2 011: polling channel 3 100: polling channel 4 101: polling channel 5 110: output tie0 111: output tie0

40005024 ACMP0_OPB ACMP0 hall output B set register 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name														ACMP0_OPB_SEL		
Type														RW		
Reset														0	0	0

Bit(s)	Name	Description
2:0	ACMP0_OPB_SEL	ACMP0 hall output B set 000: polling channel 0 001: polling channel 1 010: polling channel 2 011: polling channel 3 100: polling channel 4 101: polling channel 5 110: output tie 0 111: output tie 0

40005028 ACMP0_OPC ACMP0 hall output C set register 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name														ACMP0_OPC_SEL		
Type														RW		
Reset														0	0	0

Bit(s)	Name	Description
2:0	ACMP0_OPC_SEL	ACMP0 hall output C set 000: polling channel 0 001: polling channel 1 010: polling channel 2 011: polling channel 3 100: polling channel 4 101: polling channel 5 110: output tie 0 111: output tie 0

4000502C ACMP_DACSR ACMP DAC reference select register 00000000

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

Bit(s)	Name	Description
0	DAC_REF	DAC reference select 0: The DAC selects bandgap as the reference 1: The DAC selects Vdd as the reference

40005030 ACMP1_CR0 ACMP1 configuration register 0 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									ACMP1_EN	ACMP1_HYST		ACMP1_IE			ACMP1_MOD	
Type									RW	RW		RW			RW	
Reset									0	0		0			0	0

Bit(s)	Name	Description
7	ACMP1_EN	ACMP1 enable 0: disable 1: enable

Bit(s)	Name	Description
6	ACMP1_HYST	analog comparator 1 hysteresis selection 0: 20mV 1: 30mV
4	ACMP1_IE	ACMP1 interrupt enable 0: disable 1: enable
1:0	ACMP1_MOD	determines the ACMP1 sensitivity modes of the interrupt trigger 00: ACMP1 interrupt on output falling edge. 01: ACMP1 interrupt on output rising edge. 10: ACMP1 interrupt on output falling edge. 01: ACMP1 interrupt on output falling or rising edge.

40005034 ACMP1_CR1 ACMP1 configuration register 1 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name										ACMP1_PSEL				ACMP1_NSEL		
Type										RW				RW		
Reset										0	0	0		0	0	0

Bit(s)	Name	Description
6:4	ACMP1_PSEL	ACMP1 positive input select 000: external input 0 001: external input 1 010: external input 2 011: external input 3 100: external input 4 101: DAC1 output 110: ACMP1 output tie 0 111: ACMP1 output tie 0
2:0	ACMP1_NSEL	ACMP1 negative input select 000: external input 0 001: external input 1 010: external input 2 011: external input 3 100: external input 4 101: DAC1 output 110: ACMP0 output tie 0 111: ACMP0 output tie 0

40005038 ACMP1_CR2 ACMP1 configuration register 2 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name										DAC1_EN	DAC1_VAL					
Type										RW	RW					
Reset										0	0	0	0	0	0	0

Bit(s)	Name	Description
7	DAC1_EN	DAC1 enable

Bit(s)	Name	Description
		0: disable 1: enable
5:0	DAC1_VAL	DAC1 output level selection

4000503C ACMP1_DSR ACMP1 data and status register 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name										ACMP1_WUF		ACMP1_F				ACMP1_O
Type										RW		RW				RW
Reset										0		0				0

Bit(s)	Name	Description
6	ACMP1_WUF	ACMP1 low power wakeup flag write 1 clear
4	ACMP1_F	ACMP1 interrupt flag write 1 clear
0	ACMP1_O	ACMP1 output

40008820 ACMPDAC_CFG0 ACMPDAC Configure register 0 01FFFC00

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

Bit(s)	Name	Description
		ACMP LPF Select ACMP0 and ACMP1 share the Same LPF Setting, Suggest Select 1MHz for Better performance and Select 200KHz to filter external noise
12:10	LPFSEL	111 1 MHz 110 1 MHz 101 750 kHz 100 750 kHz 011 500 kHz 010 500 kHz 001 200 kHz 000 200 kHz
13	HYST0	ACMP0 Hysteresis Enable Bit 0: Disable 1: Enable
14	HYST1	ACMP1 Hysteresis Enable Bit 0: Disable

Bit(s)	Name	Description
		1: Enable
15	MODE	ACMP Mode Select 0: Low Power Mode (for Saving Power Consumption) 1: Normal Mode

11.5 Interrupts

Table 11-2 ACMP interrupt table

Interrupt	Bit field	Enable	Clear
ACMP0	ACMP0_SR[ACMP0_F0]	ACMP0_IE	Write 1
ACMP0	ACMP0_SR[ACMP0_F1]	ACMP0_IE	Write 1
ACMP0	ACMP0_SR[ACMP0_F2]	ACMP0_IE	Write 1
ACMP0	ACMP0_SR[ACMP0_F3]	ACMP0_IE	Write 1
ACMP0	ACMP0_SR[ACMP0_F4]	ACMP0_IE	Write 1
ACMP0	ACMP0_SR[ACMP0_F5]	ACMP0_IE	Write 1
ACMP0	ACMP0_SR[ACMP0_WUF]	ACMP0_IE	Write 1
ACMP1	ACMP1_DSR[ACMP1_F]	ACMP1_IE	Write 1
ACMP1	ACMP1_DSR[ACMP1_WUF]	ACMP1_IE	Write 1

12 PWM

12.1 Introduction

The PWM module is a two-to-six channel timer that supports input capture, output compare, and the generation of PWM signals to control electric motor and power management applications. The PWM time reference is a 16-bit counter.

The device contains up to four PWM modules of one 6-channel PWM with Full functions and three 2-channel PWM with basic TPM functions. Each PWM module can use independent external clock input. The table below summarizes the configuration of PWM modules.

Table 12-1 PWM modules configuration

Feature	PWM0/PWM1/PWM3	PWM2
Number of Channels	2	6
Periodic TOF	Yes	Yes
Input Capture Mode	Yes	Yes
Channel input filter	No	Yes
Output Compare mode	Yes	Yes
Edge-Aligned PWM	Yes	Yes
Center-Aligned PWM	Yes	Yes
Combine Mode	Yes	Yes
Complementary Mode	Yes	Yes
Inverting	Yes	Yes
Software Output Control	Yes	Yes
Deadtime insertion	Yes	Yes
Output Mask	Yes	Yes
Fault Control	No	Yes
Number of fault inputs	No	Yes
Fault input filter	No	Yes
Polarity Control	Yes	Yes
Capture Test Mode	Yes	Yes
Dual edge capture mode	Yes	Yes
Quadrature decode mode	Yes	Yes
Quadrature decode input filter	Yes	Yes

12.1.1 PWM features

The PWM features include:

- PWM source clock is selectable.
 - Source clock can be the system clock, the internal RC clock, or an external clock.
 - Selecting external clock connects PWM clock to a chip level input pin, therefore allowing to synchronize the PWM counter with an off chip clock source.
- 16-bit Prescaler divide-by 1, 2, 3 to 65535.
- 16-bit counter.
 - It can be a free-running counter or a counter with initial and final value.
 - The counting can be up or up-down.
- Each channel can be configured for input capture, output compare, or edge-aligned PWM mode.
- In Input Capture mode, the capture can occur on rising edges, falling edges or both edges.
- An input filter can be selected for some channels.

- In Output Compare mode, the output signal can be set, cleared, or toggled on match.
- All channels can be configured for center-aligned PWM mode.
- Each pair of channels can be combined to generate a PWM signal with independent control of both edges of PWM signal.
- The PWM channels can operate as pairs with equal outputs, pairs with complementary outputs, or independent channels with independent outputs.
- The deadtime insertion is available for each complementary pair.
- Generation of match triggers.
- Software control of PWM outputs.
- Up to 4 fault inputs for global fault control.
- The polarity of each channel is configurable.
- The generation of an interrupt per channel.
- The generation of an interrupt when the counter overflows.
- The generation of an interrupt when the fault condition is detected.
- Synchronized loading of write buffered PWM registers.
- Write protection for critical registers.
- Dual edge capture for pulse and period width measurement.
- Quadrature decoder with input filters, relative position counting, and interrupt on position count or capture of position count on external event.

12.1.2 Block diagram

The PWM uses one input/output (I/O) pin per channel, CHn (PWM channel (n)), where n is the channel number (0–5).

The following figure shows the PWM structure. The central component of the PWM is the 16-bit counter with programmable initial and final values and its counting can be up or up-down.

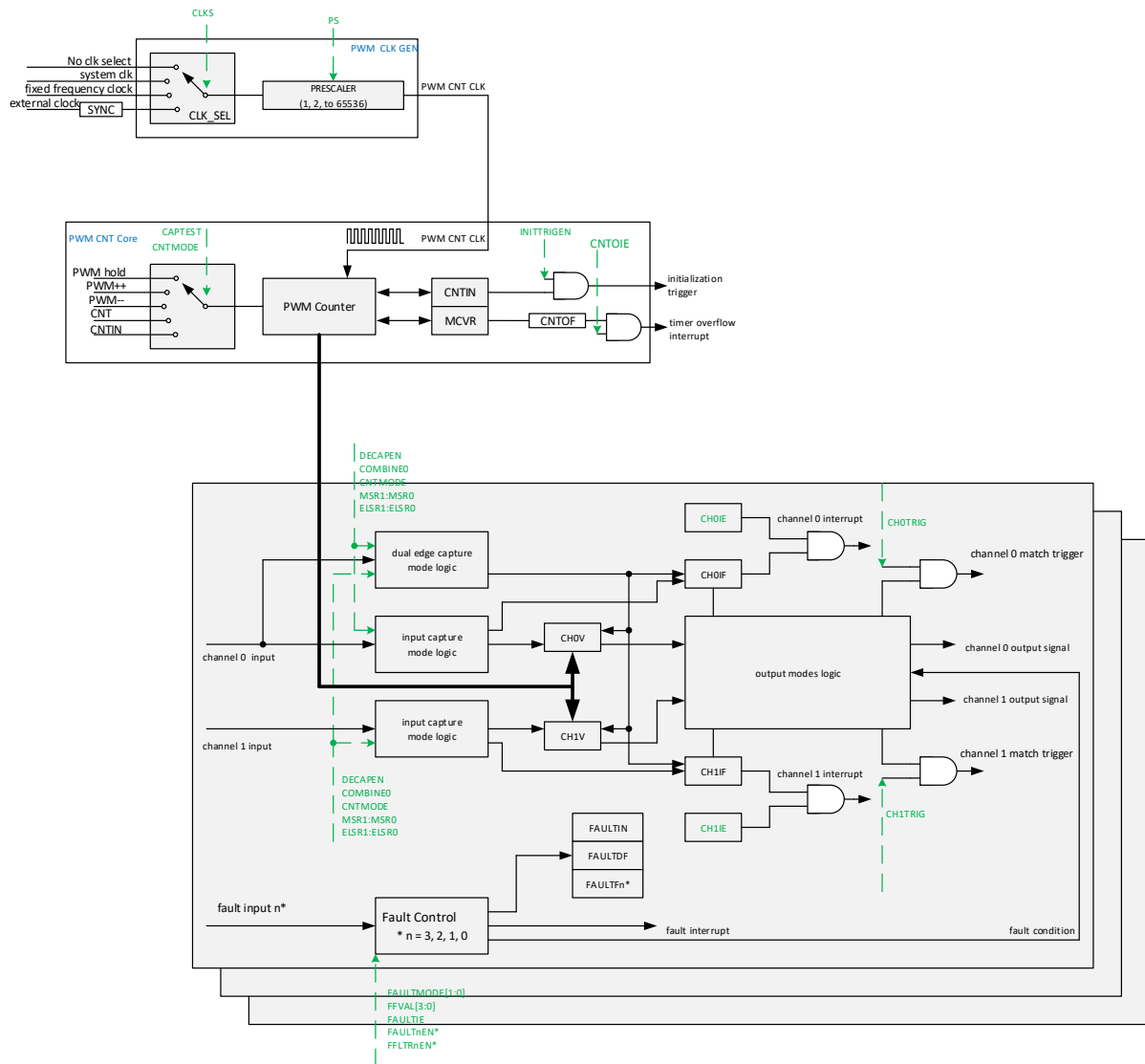


Figure 12-1 PWM block diagram

12.2 Memory map and register definition

Table 12-2 PWM register map and reset values

PWM0 Base address: (0x40013000h)

PWM1 Base address: (0x40014000h)

PWM2 Base address: (0x40015000h)

PWM3 Base address: (0x4001e000h)

Address = Base address + Offset address

Module name	Base address	Offset address
PWM0	0x40013000	0x00 ~ 0x94
PWM1	0x40014000	0x00 ~ 0x94
PWM2	0x40015000	0x00 ~ 0x94
PWM3	0x4001e000	0x00 ~ 0x94

Offset Address	Name	Width	Register Function
0x00	PWMx_INIT	32	PWM Initialization Register
0x04	PWMx_CNT	32	PWM counter value
0x08	PWMx_MCVR	32	Max Count Value Register
0x0C	PWMx_CH0SCR	32	Channel (0) Status And Control Register
0x10	PWMx_CH0V	32	Channel (0) Value
0x14	PWMx_CH1SCR	32	Channel (1) Status And Control Register
0x18	PWMx_CH1V	32	Channel (1) Value
0x1C	PWMx_CH2SCR	32	Channel (2) Status And Control Register
0x20	PWMx_CH2V	32	Channel (2) Value
0x24	PWMx_CH3SCR	32	Channel (3) Status And Control Register
0x28	PWMx_CH3V	32	Channel (3) Value
0x2C	PWMx_CH4SCR	32	Channel (4) Status And Control Register
0x30	PWMx_CH4V	32	Channel (4) Value
0x34	PWMx_CH5SCR	32	Channel (5) Status And Control Register
0x38	PWMx_CH5V	32	Channel (5) Value
0x4C	PWMx_CNTIN	32	Counter Initial Value
0x50	PWMx_STR	32	Capture And Compare Status Register
0x54	PWMx_FUNCSEL	32	Features Mode Selection
0x58	PWMx_SYNC	32	Synchronization
0x5C	PWMx_OUTINIT	32	Initial State For Channels Output
0x60	PWMx_OMCR	32	Output Mask Control Register
0x64	PWMx_MODESEL	32	Mode Selection Register
0x68	PWMx_DTSET	32	Deadtime Setting Register
0x6C	PWMx_EXTTRIG	32	PWM External Trigger
0x70	PWMx_CHOPOLCR	32	Channel Output Polarity Control Register
0x74	PWMx_FDSR	32	Fault Detect Status Register
0x78	PWMx_CAPFILTER	32	Input Capture Mode Filter Control
0x7C	PWMx_FFAFER	32	Fault Filter and Fault Enable Register
0x80	PWMx_QEI	32	Quadrature Decoder Control And Status
0x84	PWMx_CONF	32	Configuration

PWM1 Base address: (0x40014000h)

PWM2 Base address: (0x40015000h)

PWM3 Base address: (0x4001e000h)

Address = Base address + Offset address

Module name	Base address	Offset address
PWM0	0x40013000	0x00 ~ 0x94
PWM1	0x40014000	0x00 ~ 0x94
PWM2	0x40015000	0x00 ~ 0x94
PWM3	0x4001e000	0x00 ~ 0x94

0x88	PWMx_FLTPOL	32	PWM Fault Input Polarity
0x8C	PWMx_SYNCONF	32	Synchronization Configuration
0x90	PWMx_INVCR	32	PWM Inverting Control
0x94	PWMx_CHOSWCR	32	Channel Software Output Control Register

0x00 PWMx_INIT PWM Initialization Register 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name									CLKPSC[15:8]							
Type									RW							
Reset									0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CLKPSC[7:0]								CN TO F	CN TO I E	CN TM OD E	CLKSRC				
Type	RW								RO	RW	RW	RW				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0			

Bit(s)	Name	Description
23:8	CLKPSC	PWM CLK prescaler The new prescaler factor affects the clock source on the next system clock cycle after the new value is updated into the register bits. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.
7	CNTOF	Timer Overflow Flag Set by hardware when the PWM counter passes the value in the MCVR register. The CNTOF bit is cleared by reading the INIT register while CNTOF is set and then writing a 0 to CNTOF bit. Writing a 1 to CNTOF has no effect. If another PWM overflow occurs between the read and write operations, the write operation has no effect. Therefore, CNTOF remains set indicating an overflow has occurred. In this case, a CNTOF interrupt request is not lost due to the clearing sequence for a previous CNTOF. 0 : PWM counter has not overflowed. 1: PWM counter has overflowed.
6	CNTOIE	Timer Overflow Interrupt Enable Enables PWM overflow interrupts. 0 : Disable CNTOF interrupts. Use software polling. 1 : Enable CNTOF interrupts. An interrupt is generated when CNTOF equals one.
5	CNTMODE	Center-Aligned PWM Select Select CPWM mode. This mode configures the PWM to operate in Up-Down

Bit(s)	Name	Description
		Counting mode. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1. 0 : PWM counter operates in Up Counting mode. 1 : PWM counter operates in Up-Down Counting mode.
4:3	CLKSRC	Clock Source Selection Select one of the three PWM counter clock sources. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1. 00: No clock selected. This in effect disables the PWM counter. 01: System clock 10 :Fixed frequency clock 11: External clock

0x04 PWMx_CNT PWM counter value 00000000

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

Bit(s)	Name	Description
15:0	COUNT	PWM counter value. The CNT register contains the PWM counter value. Reset clears the CNT register. Writing any value to COUNT updates the counter with its initial value, CNTIN.

0x08 PWMx_MCVR Max Count Value Register 00000000

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

Bit(s)	Name	Description
15:0	MCVR	Max Count Value Register The MCVR register contains the modulo value for the PWM counter. After the PWM counter reaches the MCVR value, the overflow flag (CNTOF) becomes set at the next clock, and the next value of PWM counter depends on the selected counting method. Writing to the MCVR register latches the value into a buffer. The MCVR register is updated with the value of its write buffer according to Registers updated from write buffers. Initialize the PWM counter, by writing to CNT, before writing to the MCVR register to avoid confusion about when the first counter overflow will occur.

0x0C PWMx_CH0SCR Channel (0) Status And Control Register 00000000

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

AutoChips Confidential

© 2013 - 2021 AutoChips Inc.

Page 159 of 360

0x0C PWMx_CH0SCR Channel (0) Status And Control Register 00000000

Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									CHI F	CHI E	MS R1	MS R0	ELS R1	ELS R0		
Type									RO	RW	RW	RW	RW	RW		
Reset									0	0	0	0	0	0		

Bit(s)	Name	Description
7	CHIF	Channel Interrupt Flag Set by hardware when an event occurs on the channel. CHIF is cleared by reading the CSC register while CHnIF is set and then writing a 0 to the CHIF bit. Writing a 1 to CHIF has no effect. If another event occurs between the read and write operations, the write operation has no effect. Therefore, CHIF remains set indicating an event has occurred. In this case, a CHIF interrupt request is not lost due to the clearing sequence for a previous CHIF. 0 : No channel event has occurred. 1 : A channel event has occurred.
6	CHIE	Channel Interrupt Enable Enables channel interrupts. 0 : Disable channel interrupts. Use software polling. 1: Enable channel interrupts.
5	MSR1	Channel Mode Select Register 1 Used for further selections in the channel logic. Its functionality is dependent on the channel mode. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.
4	MSR0	Channel Mode Select Register 0 Used for further selections in the channel logic. Its functionality is dependent on the channel mode. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.
3	ELSR1	Edge or Level Select Register 1 The functionality of ELSR1 and ELSR0 depends on the channel mode. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.
2	ELSR0	Edge or Level Select Register 0 The functionality of ELSR1 and ELSR0 depends on the channel mode. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.

0x10 PWMx_CH0V Channel (0) Value 00000000

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

Bit(s)	Name	Description
15:0	CHCVAL	Channel Count Value These registers contain the captured PWM counter value for the input modes or the match value for the output modes. In Input Capture, Capture Test, and Dual Edge Capture modes, any write to a CHnV register is ignored. In output modes, writing to a CHnV register latches the value into a buffer. A CHnV

Bit(s)	Name	Description
		register is updated with the value of its write buffer according to Registers updated from write buffers.

0x4C PWMx_CNTIN Counter Initial Value 00000000

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

Bit(s)	Name	Description
15:0	CNTINIT	Counter Initial Value The Counter Initial Value register contains the initial value for the PWM counter. Writing to the CNTIN register latches the value into a buffer. The CNTIN register is updated with the value of its write buffer according to Registers updated from write buffers. When the PWM clock is initially selected, by writing a non-zero value to the CLKS bits, the PWM counter starts with the value 0x0000. To avoid this behavior, before the first write to select the PWM clock, write the new value to the CNTIN register and then initialize the PWM counter by writing any value to the CNT register.

0x50 PWMx_STR Capture And Compare Status Register 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name											CH 5SF	CH 4SF	CH 3SF	CH 2SF	CH 1SF	CH 0SF
Type											RO	RO	RO	RO	RO	RO
Reset											0	0	0	0	0	0

Bit(s)	Name	Description
5	CH5SF	Channel 5 Status Flag 0 : No channel event has occurred. 1: A channel event has occurred.
4	CH4SF	Channel 4 Status Flag 0 : No channel event has occurred. 1: A channel event has occurred.
3	CH3SF	Channel 3 Status Flag 0: No channel event has occurred. 1: A channel event has occurred.
2	CH2SF	Channel 2 Status Flag 0: No channel event has occurred. 1: A channel event has occurred.
1	CH1SF	Channel 1 Status Flag 0: No channel event has occurred. 1: A channel event has occurred.

Bit(s)	Name	Description
0	CH0SF	Channel 0 Status Flag 0: No channel event has occurred. 1: A channel event has occurred.

0x54	PWMx_FUNCSEL								Features Mode Selection								00000004
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Name																	
Type																	
Reset																	
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Name									FA UL TIE	FAULTMO DE		CA PT ES T	PW MS YN C	WP DIS	INI T	PW ME N2	
Type									RW	RW		RW	RW	RW	RW	RW	
Reset									0	0	0	0	0	1	0	0	

Bit(s)	Name	Description
7	FAULTIE	Fault Interrupt Enable Enables the generation of an interrupt when a fault is detected by PWM and the PWM fault control is enabled. 0: Fault control interrupt is disabled. 1: Fault control interrupt is enabled.
6:5	FAULTMODE	Fault Control Mode Defines the PWM fault control mode. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1. 00 : Fault control is disabled for all channels. 01: Fault control is enabled for even channels only (channels 0, 2, 4, and 6), and the selected mode is the manual fault clearing. 10: Fault control is enabled for all channels, and the selected mode is the manual fault clearing. 11: Fault control is enabled for all channels, and the selected mode is the automatic fault clearing.
4	CAPTEST	Capture Test Mode Enable Enables the capture test mode. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1. 0: Capture test mode is disabled. 1: Capture test mode is enabled.
3	PWMSYNC	PWM Synchronization Mode Selects which triggers can be used by MCVR, CHnV, OMCR, and PWM counter synchronization. The PWMSYNC bit configures the synchronization when SYNCMODE is zero. 0 : No restrictions. Software and hardware triggers can be used by MCVR, CHnV, OMCR and PWM counter synchronization. 1: Software trigger can only be used by MCVR and CHnV synchronization, and hardware triggers can only be used by OMCR and PWM counter synchronization.
2	WPDIS	Write Protection Disable When write protection is enabled (WPDIS = 0), write protected bits cannot be written. When write protection is disabled (WPDIS = 1), write protected bits can be written. The WPDIS bit is the negation of the WPEN bit. WPDIS is

Bit(s)	Name	Description
		cleared when 1 is written to WPEN. WPDIS is set when WPEN bit is read as a 1 and then 1 is written to WPDIS. Writing 0 to WPDIS has no effect. 0: Write protection is enabled. 1: Write protection is disabled.
1	INIT	Initialize The Channels Output When a 1 is written to INIT bit, the channels output is initialized according to the state of their corresponding bit in the OUTINIT register. Writing a 0 to INIT bit has no effect. The INIT bit is always read as 0.
0	PWMEN2	PWM Enable This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1. 0: Only the TPM-compatible registers (first set of registers) can be used without any restriction. Do not use the PWM-specific registers. 1: All registers including the PWM-specific registers (second set of registers) are available for use with no restrictions.

0x58	PWMx_SYNC				Synchronization												00000000
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Name																	
Type																	
Reset																	
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Name					PWM_SYNCPOL	PWM_TRI_G2_CLEA R_CNT	PWM_TRI_G1_CLEA R_CNT	PWM_TRI_G0_CLEA R_CNT	SW_SYNC	SW_TRI_G	PWM0_TRI_G	ACMP_TRI_G	OMSYNCP	REINIT	MAXSYNCP	MINSYNCP	
Type					RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	
Reset					0	0	0	0	0	0	0	0	0	0	0	0	

Bit(s)	Name	Description
11	PWM_SYNCPOL	PWM_SYNCPOL Selects when the POL register is updated with the value of its buffer. 0: POL register is updated with the value of its buffer in all rising edges of the system clock. 1: POL register is updated with the value of its buffer only by the PWM synchronization.
10	PWM_TRIG2_CLEAR_CNT	PWM_TRIG2_CLEAR_CNT 0: when TRIG 2 event happened, not clear the CNT core 1: when TRIG2 event happened, clear the CNT core
9	PWM_TRIG1_CLEAR_CNT	PWM_TRIG1_CLEAR_CNT 0: when TRIG1 event happened, not clear the CNT core 1: when TRIG1 event happened, clear the CNT core
8	PWM_TRIG0_CLEAR_CNT	PWM_TRIG0_CLEAR_CNT 0: when TRIG0 event happened, not clear the CNT core 1: when TRIG0 event happened, clear the CNT core
7	SWSYNC	PWM Synchronization Software Trigger Selects the software trigger as the PWM synchronization trigger. The software trigger happens when a 1 is written to SWSYNC bit. 0 : Software trigger is not selected.

Bit(s)	Name	Description
		1 : Software trigger is selected.
6	SWTRIG	PWM Synchronization Hardware Trigger 2 Enables hardware trigger 2 to the PWM synchronization. Hardware trigger 2 happens when a rising edge is detected at the trigger 2 input signal. 0: Trigger is disabled. 1: Trigger is enabled.
5	PWM0TRIG	PWM Synchronization Hardware Trigger 1 Enables hardware trigger 1 to the PWM synchronization. Hardware trigger 1 happens when a rising edge is detected at the trigger 1 input signal. 0: Trigger is disabled. 1: Trigger is enabled.
4	ACMPTRIG	PWM Synchronization Hardware Trigger 0 Enables hardware trigger 0 to the PWM synchronization. Hardware trigger 0 happens when a rising edge is detected at the trigger 0 input signal. 0: Trigger is disabled. 1 : Trigger is enabled.
3	OMSYNCP	Output Mask Synchronization Selects when the OMCR register is updated with the value of its buffer. 0 : OMCR register is updated with the value of its buffer in all rising edges of the system clock. 1: OMCR register is updated with the value of its buffer only by the PWM synchronization.
2	REINIT	PWM Counter Reinitialization by Synchronization Determines if the PWM counter is reinitialized when the selected trigger for the synchronization is detected. The REINIT bit configures the synchronization when SYNCMODE is zero. 0 : PWM counter continues to count normally. 1: PWM counter is updated with its initial value when the selected trigger is detected.
1	MAXSYNCP	Maximum Loading Point Enable Selects the maximum loading point to PWM synchronization. If CNTMAX is one, the selected loading point is when the PWM counter reaches its maximum value (MCVR register). 0: The maximum loading point is disabled. 1 : The maximum loading point is enabled.
0	MINSYNCP	Minimum Loading Point Enable Selects the minimum loading point to PWM synchronization. If CNTMIN is one, the selected loading point is when the PWM counter reaches its minimum value (CNTIN register). 0 : The minimum loading point is disabled. 1 ; The minimum loading point is enabled.

0x5C	PWMx_OUTINIT				Initial State For Channels Output										00000000	
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name											CH 5OI V	CH 4OI V	CH 3OI V	CH 2OI V	CH 1OI V	CH 0OI V

0x5C PWMx_OUTINIT Initial State For Channels Output 00000000

Type											RW	RW	RW	RW	RW	RW
Reset											0	0	0	0	0	0

Bit(s)	Name	Description
5	CH5OIV	Channel 5 Output Initialization Value Selects the value that is forced into the channel output when the initialization occurs. 0: The initialization value is 0. 1: The initialization value is 1.
4	CH4OIV	Channel 4 Output Initialization Value Selects the value that is forced into the channel output when the initialization occurs. 0 : The initialization value is 0. 1: The initialization value is 1.
3	CH3OIV	Channel 3 Output Initialization Value Selects the value that is forced into the channel output when the initialization occurs. 0 : The initialization value is 0. 1 : The initialization value is 1.
2	CH2OIV	Channel 2 Output Initialization Value Selects the value that is forced into the channel output when the initialization occurs. 0: The initialization value is 0. 1: The initialization value is 1.
1	CH1OIV	Channel 1 Output Initialization Value Selects the value that is forced into the channel output when the initialization occurs. 0 : The initialization value is 0. 1 : The initialization value is 1.
0	CH0OIV	Channel 0 Output Initialization Value Selects the value that is forced into the channel output when the initialization occurs. 0 : The initialization value is 0. 1: The initialization value is 1.

0x60 PWMx_OMCR Output Mask Control Register 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name											CH5OMEN	CH4OMEN	CH3OMEN	CH2OMEN	CH1OMEN	CH0OMEN
Type											RW	RW	RW	RW	RW	RW
Reset											0	0	0	0	0	0

Bit(s)	Name	Description
5	CH5OMEN	Channel 5 Output Mask

Bit(s)	Name	Description
		Defines if the channel output is masked or unmasked. 0: Channel output is not masked. It continues to operate normally. 1: Channel output is masked. It is forced to its inactive state.
4	CH4OMEN	Channel 4 Output Mask Defines if the channel output is masked or unmasked. 0: Channel output is not masked. It continues to operate normally. 1: Channel output is masked. It is forced to its inactive state.
3	CH3OMEN	Channel 3 Output Mask Defines if the channel output is masked or unmasked. 0: Channel output is not masked. It continues to operate normally. 1: Channel output is masked. It is forced to its inactive state.
2	CH2OMEN	Channel 2 Output Mask Defines if the channel output is masked or unmasked. 0: Channel output is not masked. It continues to operate normally. 1: Channel output is masked. It is forced to its inactive state.
1	CH1OMEN	Channel 1 Output Mask Defines if the channel output is masked or unmasked. 0: Channel output is not masked. It continues to operate normally. 1: Channel output is masked. It is forced to its inactive state.
0	CH0OMEN	Channel 0 Output Mask Defines if the channel output is masked or unmasked. 0: Channel output is not masked. It continues to operate normally. 1: Channel output is masked. It is forced to its inactive state.

0x64 PWMx MODESEL PWM Function Mode Selection 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name										PAI R2 FA UL TE N	PAI R2 SY NC EN	PAI R2 DT EN	PAI R2 DE CA P	PAI R2 DE CA PE N	PAI R2 CO MP EN	PAI R2 CO MB IN EN
Type										R W	R W	R W	R W	R W	R W	R W
Reset										0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name		PAI R1 FA UL TE N	PAI R1 SY NC EN	PAI R1 DT EN	PAI R1 DE CA P	PAI R1 DE CA PE N	PAI R1 CO MP EN	PAI R1 CO MB IN EN		PAI R0 FA UL TE N	PAI R0 SY NC EN	PAI R0 DT EN	PAI R0 DE CA P	PAI R0 DE CA PE N	PAI R0 CO MP EN	PAI R0 CO MB IN EN
Type		R W	R W	R W	R W	R W	R W	R W		R W	R W	R W	R W	R W	R W	R W
Reset		0	0	0	0	0	0	0		0	0	0	0	0	0	0

Bit(s)	Name	Description
22	PAIR2FAULTEN	Fault Control Enable for n = 4 Enables the fault control in channels (n) and (n+1). This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1. 0: The fault control in this pair of channels is disabled.

Bit(s)	Name	Description
		1: The fault control in this pair of channels is enabled.
21	PAIR2SYNCEN	Synchronization Enable for n = 4 Enables PWM synchronization of registers CH(n)V and CH(n+1)V. 0: The PWM synchronization in this pair of channels is disabled. 1: The PWM synchronization in this pair of channels is enabled.
20	PAIR2DTEN	Deadtime Enable for n = 4 Enables the deadtime insertion in the channels (n) and (n+1). This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1. 0: The deadtime insertion in this pair of channels is disabled. 1: The deadtime insertion in this pair of channels is enabled.
19	PAIR2DECAP	Dual Edge Capture Mode Captures for n = 4 Enables the capture of the PWM counter value according to the channel (n) input event and the configuration of the dual edge capture bits. This field applies only when PWMEN2 = 1 and DECAPEN = 1. DECAP bit is cleared automatically by hardware if dual edge capture one-shot mode is selected and when the capture of channel (n+1) event is made. 0: The dual edge captures are inactive. 1: The dual edge captures are active.
18	PAIR2DECAPEN	Dual Edge Capture Mode Enable for n = 4 Enables the Dual Edge Capture mode in the channels (n) and (n+1). This bit reconfigures the function of MSnR0, ELSnR1:ELSnR0 and ELS(n+1) R1: ELS(n+1) R0 bits in Dual Edge Capture mode. This field applies only when PWMEN2 = 1. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1. 0: The Dual Edge Capture mode in this pair of channels is disabled. 1: The Dual Edge Capture mode in this pair of channels is enabled.
17	PAIR2COMPEN	Complement of Channel (n) for n = 4 Enables Complementary mode for the combined channels. In Complementary mode the channel (n+1) output is the inverse of the channel (n) output. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1. 0: The channel (n+1) output is the same as the channel (n) output. 1: The channel (n+1) output is the complement of the channel (n) output.
16	PAIR2COMBINEN	Combine Channels for n = 4 Enables the combine feature for channels (n) and (n+1). This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1. 0: Channels (n) and (n+1) are independent. 1: Channels (n) and (n+1) are combined.
14	PAIR1FAULTEN	Fault Control Enable for n = 2 Enables the fault control in channels (n) and (n+1). This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1. 0: The fault control in this pair of channels is disabled. 1: The fault control in this pair of channels is enabled.
13	PAIR1SYNCEN	Synchronization Enable for n = 2 Enables PWM synchronization of registers CH(n)V and CH(n+1) V. 0: The PWM synchronization in this pair of channels is disabled. 1: The PWM synchronization in this pair of channels is enabled.
12	PAIR1DTEN	Deadtime Enable for n = 2 Enables the deadtime insertion in the channels (n) and (n+1). This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1. 0: The deadtime insertion in this pair of channels is disabled. 1: The deadtime insertion in this pair of channels is enabled.
11	PAIR1DECAP	Dual Edge Capture Mode Captures for n = 2

Bit(s)	Name	Description
		<p>Enables the capture of the PWM counter value according to the channel (n) input event and the configuration of the dual edge capture bits. This field applies only when PWMEN2 = 1 and DECAPEN = 1. DECAP bit is cleared automatically by hardware if dual edge capture one-shot mode is selected and when the capture of channel (n+1) event is made.</p> <p>0: The dual edge captures are inactive.</p> <p>1: The dual edge captures are active.</p>
10	PAIR1DECAPEN	<p>Dual Edge Capture Mode Enable for n = 2</p> <p>Enables the Dual Edge Capture mode in the channels (n) and (n+1). This bit reconfigures the function of MSnR0, ELSnR1:ELSnR0 and ELS(n+1) R1: ELS(n+1) R0 bits in Dual Edge Capture mode This field applies only when PWMEN2 = 1. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.</p> <p>0: The Dual Edge Capture mode in this pair of channels is disabled.</p> <p>1: The Dual Edge Capture mode in this pair of channels is enabled.</p>
9	PAIR1COMPEN	<p>Complement of Channel (n) for n = 2</p> <p>Enables Complementary mode for the combined channels. In Complementary mode the channel (n+1) output is the inverse of the channel (n) output. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.</p> <p>0: The channel (n+1) output is the same as the channel (n) output.</p> <p>1: The channel (n+1) output is the complement of the channel (n) output.</p>
8	PAIR1COMBINEN	<p>Combine Channels for n = 2</p> <p>Enables the combine feature for channels (n) and (n+1). This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.</p> <p>0: Channels (n) and (n+1) are independent.</p> <p>1: Channels (n) and (n+1) are combined.</p>
6	PAIR0FAULTEN	<p>Fault Control Enable for n = 0</p> <p>Enables the fault control in channels (n) and (n+1). This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.</p> <p>0: The fault control in this pair of channels is disabled.</p> <p>1: The fault control in this pair of channels is enabled.</p>
5	PAIR0SYNCEN	<p>Synchronization Enable for n = 0</p> <p>Enables PWM synchronization of registers CH(n)V and CH(n+1) V.</p> <p>0: The PWM synchronization in this pair of channels is disabled.</p> <p>1: The PWM synchronization in this pair of channels is enabled.</p>
4	PAIR0DTEN	<p>Deadtime Enable for n = 0</p> <p>Enables the deadtime insertion in the channels (n) and (n+1). This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.</p> <p>0: The deadtime insertion in this pair of channels is disabled.</p> <p>1: The deadtime insertion in this pair of channels is enabled.</p>
3	PAIR0DECAP	<p>Dual Edge Capture Mode Captures for n = 0</p> <p>Enables the capture of the PWM counter value according to the channel (n) input event and the configuration of the dual edge capture bits. This field applies only when PWMEN2 = 1 and DECAPEN = 1. DECAP bit is cleared automatically by hardware if dual edge capture one-shot mode is selected and when the capture of channel (n+1) event is made.</p> <p>0: The dual edge captures are inactive.</p> <p>1: The dual edge captures are active.</p>
2	PAIR0DECAPEN	<p>Dual Edge Capture Mode Enable for n = 0</p> <p>Enables the Dual Edge Capture mode in the channels (n) and (n+1). This bit</p>

Bit(s)	Name	Description
		reconfigures the function of MSnR0, ELSnR1:ELSnR0 and ELS(n+1) R1: ELS(n+1) R0 bits in Dual Edge Capture mode This field applies only when PWMEN2 = 1. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1. 0: The Dual Edge Capture mode in this pair of channels is disabled. 1: The Dual Edge Capture mode in this pair of channels is enabled.
1	PAIR0CMPEN	Complement of Channel (n) for n = 0 Enables Complementary mode for the combined channels. In Complementary mode the channel (n+1) output is the inverse of the channel (n) output. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1. 0 The channel (n+1) output is the same as the channel (n) output. 1 The channel (n+1) output is the complement of the channel (n) output.
0	PAIR0COMBINEN	Combine Channels for n = 0 Enables the combine feature for channels (n) and (n+1). This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1. 0: Channels (n) and (n+1) are independent. 1: Channels (n) and (n+1) are combined.

0x68	PWMx_DTSET				Deadtime Insertion Control										00000000	
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									DTPSC		DTVAL					
Type									RW		RW					
Reset									0	0	0	0	0	0	0	0

Bit(s)	Name	Description
7:6	DTPSC	Deadtime Prescaler Value Selects the division factor of the system clock. This prescaled clock is used by the deadtime counter. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1. 0x: Divide the system clock by 1. 10: Divide the system clock by 4. 11: Divide the system clock by 16.
5:0	DTVAL	Deadtime Value Selects the deadtime insertion value for the deadtime counter. The deadtime counter is clocked by a scaled version of the system clock. See the description of DTPS. Deadtime insert value = (DTPSC x DTVAL). DTVAL selects the number of deadtime counts inserted as follows: When DTVAL is 0, no counts are inserted. When DTVAL is 1, 1 count is inserted. When DTVAL is 2, 2 counts are inserted. This pattern continues up to a possible 63 counts. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.

0x6C PWMx_EXTTRIG PWM External Trigger 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									TRIGF	INITTRIGEN	CH5TRIG	CH4TRIG	CH3TRIG	CH2TRIG	CH1TRIG	CH0TRIG
Type									RW	RW	RW	RW	RW	RW	RW	RW
Reset									0	0	0	0	0	0	0	0

Bit(s)	Name	Description
7	TRIGF	Channel Trigger Flag Set by hardware when a channel trigger is generated. Clear TRIGF by reading EXTTRIG while TRIGF is set and then writing a 0 to TRIGF. Writing a 1 to TRIGF has no effect. If another channel trigger is generated before the clearing sequence is completed, the sequence is reset so TRIGF remains set after the clear sequence is completed for the earlier TRIGF. 0: No channel trigger was generated. 1: A channel trigger was generated.
6	INITTRIGEN	Initialization Trigger Enable Enables the generation of the trigger when the PWM counter is equal to the CNTIN register. 0: The generation of initialization trigger is disabled. 1: The generation of initialization trigger is enabled.
5	CH5TRIG	Channel 5 Trigger Enable Enable the generation of the channel trigger when the PWM counter is equal to the CHnV register. 0: The generation of the channel trigger is disabled. 1: The generation of the channel trigger is enabled.
4	CH4TRIG	Channel 4 Trigger Enable Enable the generation of the channel trigger when the PWM counter is equal to the CHnV register. 0: The generation of the channel trigger is disabled. 1: The generation of the channel trigger is enabled.
3	CH3TRIG	Channel 3 Trigger Enable Enable the generation of the channel trigger when the PWM counter is equal to the CHnV register. 0: The generation of the channel trigger is disabled. 1: The generation of the channel trigger is enabled.
2	CH2TRIG	Channel 2 Trigger Enable Enable the generation of the channel trigger when the PWM counter is equal to the CHnV register. 0: The generation of the channel trigger is disabled. 1: The generation of the channel trigger is enabled.
1	CH1TRIG	Channel 1 Trigger Enable Enable the generation of the channel trigger when the PWM counter is equal to the CHnV register. 0: The generation of the channel trigger is disabled. 1: The generation of the channel trigger is enabled.
0	CH0TRIG	Channel 0 Trigger Enable

Bit(s)	Name	Description
--------	------	-------------

Enable the generation of the channel trigger when the PWM counter is equal to the CHnV register.

0: The generation of the channel trigger is disabled.

1: The generation of the channel trigger is enabled.

0x70

PWMx_CHOPOLCR

Channels Output Polarity Control Register

00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name											CH5POL	CH4POL	CH3POL	CH2POL	CH1POL	CH0POL
Type											RW	RW	RW	RW	RW	RW
Reset											0	0	0	0	0	0

Bit(s)	Name	Description
--------	------	-------------

Channel 5 Polarity
Defines the polarity of the channel output. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.
0: The channel polarity is active high.
1: The channel polarity is active low.

Channel 4 Polarity
Defines the polarity of the channel output. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.
0: The channel polarity is active high.
1: The channel polarity is active low.

Channel 3 Polarity
Defines the polarity of the channel output. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.
0: The channel polarity is active high.
1: The channel polarity is active low.

Channel 2 Polarity
Defines the polarity of the channel output. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.
0: The channel polarity is active high.
1: The channel polarity is active low.

Channel 1 Polarity
Defines the polarity of the channel output. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.
0: The channel polarity is active high.
1: The channel polarity is active low.

Channel 0 Polarity
Defines the polarity of the channel output. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1.
0: The channel polarity is active high.
1: The channel polarity is active low.

0x74 PWMx_FDSR Fault Detect Status Register 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									FA ULT DF	WP EN	FA ULT IN		FAU LTD F3	FA ULT DF2	FA ULT DF1	FA ULT DF0
Type									RO	RW	RO		RO	RO	RO	RO
Reset									0	0	0		0	0	0	0

Bit(s)	Name	Description
7	FAULTDF	Fault Detection Flag Represents the logic OR of the individual FAULTDF j bits where j = 3, 2, 1, 0. Clear FAULTDF by reading the FDSR register while FAULTDF is set and then writing a 0 to FAULTDF while there is no existing fault condition at the enabled fault inputs. Writing a 1 to FAULTDF has no effect. If another fault condition is detected in an enabled fault input before the clearing sequence is completed, the sequence is reset so FAULTDF remains set after the clearing sequence is completed for the earlier fault condition. FAULTDF is also cleared when FAULTDF j bits are cleared individually. 0: No fault condition was detected. 1: A fault condition was detected.
6	WPEN	Write Protection Enable The WPEN bit is the negation of the WPDIS bit. WPEN is set when 1 is written to it. WPEN is cleared when WPEN bit is read as a 1 and then 1 is written to WPDIS. Writing 0 to WPEN has no effect. 0: Write protection is disabled. Write protected bits can be written. 1: Write protection is enabled. Write protected bits cannot be written.
5	FAULTIN	Fault Inputs Represents the logic OR of the enabled fault inputs after their filter (if their filter is enabled) when fault control is enabled. 0: The logic OR of the enabled fault inputs is 0. 1: The logic OR of the enabled fault inputs is 1.
3	FAULTDF3	Fault Detection Flag 3 Set by hardware when fault control is enabled, the corresponding fault input is enabled and a fault condition is detected at the fault input. Clear FAULTDF 3 by reading the FDSR register while FAULTDF 3 is set and then writing a 0 to FAULTDF 3 while there is no existing fault condition at the corresponding fault input. Writing a 1 to FAULTDF 3 has no effect. FAULTDF 3 bit is also cleared when FAULTDF bit is cleared. If another fault condition is detected at the corresponding fault input before the clearing sequence is completed, the sequence is reset so FAULTDF 3 remains set after the clearing sequence is completed for the earlier fault condition. 0: No fault condition was detected at the fault input. 1: A fault condition was detected at the fault input.
2	FAULTDF2	Fault Detection Flag 2 Set by hardware when fault control is enabled, the corresponding fault input is enabled and a fault condition is detected at the fault input. Clear FAULTDF 2 by reading the FDSR register while FAULTDF 2 is set and then writing a 0 to FAULTDF 2 while there is no existing fault condition at the corresponding fault input. Writing a 1 to FAULTDF 2 has no effect. FAULTDF 2 bit is also cleared when FAULTDF bit is cleared. If another fault condition is detected at the corresponding fault input before the clearing sequence is completed, the sequence is reset so FAULTDF 2 remains set after the clearing sequence is completed for the earlier fault condition. 0: No fault condition was detected at the fault input. 1: A fault condition was detected at the fault input.

Bit(s)	Name	Description
1	FAULTDF1	Fault Detection Flag 1 Set by hardware when fault control is enabled, the corresponding fault input is enabled and a fault condition is detected at the fault input. Clear FAULTDF 1 by reading the FDSR register while FAULTDF 1 is set and then writing a 0 to FAULTDF 1 while there is no existing fault condition at the corresponding fault input. Writing a 1 to FAULTDF 1 has no effect. FAULTDF 1 bit is also cleared when FAULTDF bit is cleared. If another fault condition is detected at the corresponding fault input before the clearing sequence is completed, the sequence is reset so FAULTDF 1 remains set after the clearing sequence is completed for the earlier fault condition. 0: No fault condition was detected at the fault input. 1: A fault condition was detected at the fault input.
0	FAULTDF0	Fault Detection Flag 0 Set by hardware when fault control is enabled, the corresponding fault input is enabled and a fault condition is detected at the fault input. Clear FAULTDF 0 by reading the FDSR register while FAULTDF 0 is set and then writing a 0 to FAULTDF 0 while there is no existing fault condition at the corresponding fault input. Writing a 1 to FAULTDF 0 has no effect. FAULTDF 0 bit is also cleared when FAULTDF bit is cleared. If another fault condition is detected at the corresponding fault input before the clearing sequence is completed, the sequence is reset so FAULTDF 0 remains set after the clearing sequence is completed for the earlier fault condition. 0: No fault condition was detected at the fault input. 1: A fault condition was detected at the fault input.

0x78 PWMx CAPFILTER Input Capture Filter Control 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name													CH3CAPFVAL[4:1]			
Type													RW			
Reset													0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	CH3FVAL[0:0]	CH2CAPFVAL					CH1CAPFVAL					CH0CAPFVAL				
Type	RW	RW					RW					RW				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit(s)	Name	Description
19:15	CH3CAPFVAL	Channel 3 Input Filter Selects the filter value for the channel input. The filter is disabled when the value is zero.
14:10	CH2CAPFVAL	Channel 2 Input Filter Selects the filter value for the channel input. The filter is disabled when the value is zero.
9:5	CH1CAPFVAL	Channel 1 Input Filter Selects the filter value for the channel input. The filter is disabled when the value is zero.
4:0	CH0CAPFVAL	Channel 0 Input Filter Selects the filter value for the channel input. The filter is disabled when the value is zero.

0x7C PWMx_FFAFER Fault Filter and Fault Enable Register 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name					FFVAL				FF3 EN	FF2 EN	FF1 EN	FF0 EN	FE R3 EN	FE R2 EN	FE R1 EN	FE R0 EN
Type					RW				RW	RW	RW	RW	RW	RW	RW	RW
Reset					0	0	0	0	0	0	0	0	0	0	0	0

Bit(s)	Name	Description
11:8	FFVAL	Fault Input Filter Selects the filter value for the fault inputs. The fault filter is disabled when the value is zero.
7	FF3EN	Fault Input 3 Filter Enable Enables the filter for the fault input. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1. 0: Fault input filter is disabled. 1: Fault input filter is enabled.
6	FF2EN	Fault Input 2 Filter Enable Enables the filter for the fault input. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1. 0: Fault input filter is disabled. 1: Fault input filter is enabled.
5	FF1EN	Fault Input 1 Filter Enable Enables the filter for the fault input. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1. 0: Fault input filter is disabled. 1: Fault input filter is enabled.
4	FF0EN	Fault Input 0 Filter Enable Enables the filter for the fault input. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1. 0: Fault input filter is disabled. 1 ; Fault input filter is enabled.
3	FER3EN	Fault Input 3 Enable Enables the fault input. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1. 0: Fault input is disabled. 1: Fault input is enabled.
2	FER2EN	Fault Input 2 Enable Enables the fault input. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1. 0: Fault input is disabled. 1 ; Fault input is enabled.
1	FER1EN	Fault Input 1 Enable Enables the fault input. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1. 0: Fault input is disabled. 1: Fault input is enabled.

Bit(s)	Name	Description
0	FER0EN	Fault Input 0 Enable Enables the fault input. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1. 0: Fault input is disabled. 1: Fault input is enabled.

0x80 **PWMx_QEI** **Quadrature Encoder/Decoder Interface Configuration Register** 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									PH AFL TR EN	PH BFL TR EN	PH AP OL	PH BP OL	QU AD MO DE	QU ADI R	TO FDI R	QEI EN
Type									RW	RW	RW	RW	RW	RO	RO	RW
Reset									0	0	0	0	0	0	0	0

Bit(s)	Name	Description
7	PHAFLTREN	Phase A Input Filter Enable Enables the filter for the quadrature decoder phase A input. The filter value for the phase A input is defined by the CH0FVAL field of FILTER. The phase A filter is also disabled when CH0FVAL is zero. 0: Phase A input filter is disabled. 1: Phase A input filter is enabled.
6	PHBFLTREN	Phase B Input Filter Enable Enables the filter for the quadrature decoder phase B input. The filter value for the phase B input is defined by the CH1FVAL field of FILTER. The phase B filter is also disabled when CH1FVAL is zero. 0: Phase B input filter is disabled. 1: Phase B input filter is enabled.
5	PHAPOL	Phase A Input Polarity Selects the polarity for the quadrature decoder phase A input. 0: Normal polarity. Phase A input signal is not inverted before identifying the rising and falling edges of this signal. 1: Inverted polarity. Phase A input signal is inverted before identifying the rising and falling edges of this signal.
4	PHBPOL	Phase B Input Polarity Selects the polarity for the quadrature decoder phase B input. 0: Normal polarity. Phase B input signal is not inverted before identifying the rising and falling edges of this signal. 1: Inverted polarity. Phase B input signal is inverted before identifying the rising and falling edges of this signal.
3	QUADMODE	Quadrature Decoder Mode Selects the encoding mode used in the Quadrature Decoder mode. 0: Phase A and phase B encoding mode. 1: Count and direction encoding mode.
2	QUADIR	PWM Counter Direction in Quadrature Decoder Mode Indicates the counting direction. 0: Counting direction is decreasing (PWM counter decrement).

Bit(s)	Name	Description
		1: Counting direction is increasing (PWM counter increment).
1	TOFDIR	Timer Overflow Direction in Quadrature Decoder Mode Indicates if the TOF bit was set on the top or the bottom of counting. 0: TOF bit was set on the bottom of counting. There was an PWM counter decrement and PWM counter changes from its minimum value (CNTIN register) to its maximum value (MCVR register). 1: TOF bit was set on the top of counting. There was an PWM counter increment and PWM counter changes from its maximum value (MCVR register) to its minimum value (CNTIN register).
0	QEIEN	Quadrature Decoder Mode Enable Enables the Quadrature Decoder mode. In this mode, the phase A and B input signals control the PWM counter direction. The Quadrature Decoder mode has precedence over the other modes. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1. 0: Quadrature Decoder mode is disabled. 1: Quadrature Decoder mode is enabled.

0x84 PWMx CONF Configuration 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name					EVENT5_P		EVENT4_P		EVENT3_P		EVENT2_P		EVENT1_P		EVENT0_P	
Type					SC		SC		SC		SC		SC		SC	
Reset					RW		RW		RW		RW		RW		RW	
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name																
Type																
Reset																

Bit(s)	Name	Description
27:26	EVENT5_PSC	Channel 5 input event PSC 00:1 01:2 10:4 11:8
25:24	EVENT4_PSC	Channel 4 input event PSC 00:1 01:2 10:4 11:8
23:22	EVENT3_PSC	Channel 3 input event PSC 00:1 01:2 10:4 11:8
21:20	EVENT2_PSC	Channel 2 input event PSC 00:1 01:2 10:4 11:8
19:18	EVENT1_PSC	Channel 1 input event PSC 00:1 01:2

Bit(s)	Name	Description
		10:4 11:8
17:16	EVENT0_PSC	Channel 0 input event PSC 00:1 01:2 10:4 11:8
6:0	CNTOFNUM	TOF Frequency Selects the ratio between the number of counter overflows to the number of times the CNTOF bit is set. NUMTOF = 0: The CNTOF bit is set for each counter overflow. CNTOFNUM = 1: The CNTOF bit is set for the first counter overflow but not for the next overflow. CNTOFNUM = 2: The CNTOF bit is set for the first counter overflow but not for the next 2 overflows. CNTOFNUM = 3: The CNTOF bit is set for the first counter overflow but not for the next 3 overflows.

0x88	PWMx FLTPOL				PWM Fault Input Polarity								00000000			
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name													FLT 3P OL	FLT 2P OL	FLT 1P OL	FLT 0P OL
Type													RW	RW	RW	RW
Reset													0	0	0	0

Bit(s)	Name	Description
3	FLT3POL	Fault Input 3 Polarity Defines the polarity of the fault input. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1. 0: The fault input polarity is active high. A one at the fault input indicates a fault. 1: The fault input polarity is active low. A zero at the fault input indicates a fault.
2	FLT2POL	Fault Input 2 Polarity Defines the polarity of the fault input. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1. 0: The fault input polarity is active high. A one at the fault input indicates a fault. 1: The fault input polarity is active low. A zero at the fault input indicates a fault.
1	FLT1POL	Fault Input 1 Polarity Defines the polarity of the fault input. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1. 0: The fault input polarity is active high. A one at the fault input indicates a fault. 1: The fault input polarity is active low. A zero at the fault input indicates a fault.

Bit(s)	Name	Description
0	FLT0POL	Fault Input 0 Polarity Defines the polarity of the fault input. This field is write protected. It can be written only when FUNCSEL[WPDIS] = 1. 0: The fault input polarity is active high. A one at the fault input indicates a fault. 1: The fault input polarity is active low. A zero at the fault input indicates a fault.

0x8C	PWMx SYNCONF				Synchronization Configuration								00000000			
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name										HW POL	SW POL	SW VH WSY NC	IN VH WSY NC	OMV H WSY NC	PW MS VH WSY NC	CN TV H WSY NC
Type										R W	R W	R W	R W	R W	R W	R W
Reset										0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name				SWVS WSY NC	IN VS W SY NC	OMV S W SY NC	PW MS VS W SY NC	CN TV S W SY NC	SY NC M OD E		SW OC	IN VC		CN TI NC		HT RI GM OD ES EL
Type				R W	R W	R W	R W	R W	R W		R W	R W		R W		R W
Reset				0	0	0	0	0	0		0	0		0		0

Bit(s)	Name	Description
22	HWPOL	Channel CHPOLCR synchronization is activated by a hardware trigger. 0: A hardware trigger does not activate the CHPOLCR register synchronization. 1: A hardware trigger activates CHPOLCR register synchronization.
21	SWPOL	Channel CHPOLCR synchronization is activated by the software trigger. 0: The software trigger does not activate the CHPOLCR register synchronization. 1: The software trigger activates CHPOLCR register synchronization.
20	SWVHWSYNC	Software output control synchronization is activated by a hardware trigger. 0: A hardware trigger does not activate the CHOSWCR register synchronization. 1: A hardware trigger activates the CHOSWCR register synchronization.
19	INVHWSYNC	Inverting control synchronization is activated by a hardware trigger. 0: A hardware trigger does not activate the INVCR register synchronization. 1: A hardware trigger activates the INVCR register synchronization.
18	OMVHWSYNC	Output mask synchronization is activated by a hardware trigger. 0: A hardware trigger does not activate the OMCR register synchronization. 1: A hardware trigger activates the OMCR register synchronization.
17	PWMSVHWSYNC	MCVR, CNTIN, and CV registers synchronization is activated by a hardware trigger. 0: A hardware trigger does not activate MCVR, CNTIN, and CHV registers synchronization. 1: A hardware trigger activates MCVR, CNTIN, and CHV registers

Bit(s)	Name	Description
		synchronization.
16	CNTVHWSYNC	PWM counter synchronization is activated by a hardware trigger. 0: A hardware trigger does not activate the PWM counter synchronization. 1: A hardware trigger activates the PWM counter synchronization.
12	SWVSWSYNC	Software output control synchronization is activated by the software trigger. 0: The software trigger does not activate the CHOSWCR register synchronization. 1: The software trigger activates the CHOSWCR register synchronization.
11	INVSWSYNC	Inverting control synchronization is activated by the software trigger. 0: The software trigger does not activate the INVCR register synchronization. 1: The software trigger activates the INVCR register synchronization.
10	OMVSWSYNC	Output mask synchronization is activated by the software trigger. 0: The software trigger does not activate the OMCR register synchronization. 1: The software trigger activates the OMCR register synchronization.
9	PWMSVSWSYNC	MCVR, CNTIN, and CV registers synchronization is activated by the software trigger. 0: The software trigger does not activate MCVR, CNTIN, and CHV registers synchronization. 1: The software trigger activates MCVR, CNTIN, and CHV registers synchronization.
8	CNTVSWSYNC	PWM counter synchronization is activated by the software trigger. 0: The software trigger does not activate the PWM counter synchronization. 1: The software trigger activates the PWM counter synchronization.
7	SYNCMODE	Synchronization Mode Selects the PWM Synchronization mode. 0: Legacy PWM synchronization is selected. 1: Enhanced PWM synchronization is selected.
5	SWOC	CHOSWCR Register Synchronization 0: CHOSWCR register is updated with its buffer value at all rising edges of system clock. 1: CHOSWCR register is updated with its buffer value by the PWM synchronization.
4	INVC	INVCR Register Synchronization 0: INVCR register is updated with its buffer value at all rising edges of system clock. 1: INVCR register is updated with its buffer value by the PWM synchronization.
2	CNTINC	CNTIN Register Synchronization 0: CNTIN register is updated with its buffer value at all rising edges of system clock. 1: CNTIN register is updated with its buffer value by the PWM synchronization.
0	HWTRIGMODESEL	Hardware Trigger Mode 0: PWM clears the TRIGj bit when the hardware trigger j is detected, where j = 0, 1, 2. 1: PWM does not clear the TRIGj bit when the hardware trigger j is detected, where j = 0, 1, 2.

0x90

PWMx_INVCR

PWM Inverting Control Register

00000000

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

0x90 PWMx_INVCR PWM Inverting Control Register 00000000

Reset	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bit													PAI R3I NV EN	PAI R2I NV EN	PAI R1I NV EN	PAI R0I NV EN
Name																
Type													RW	RW	RW	RW
Reset													0	0	0	0

Bit(s)	Name	Description
3	PAIR0INVEN	Pair Channels 3 Inverting Enable 0: Inverting is disabled. 1: Inverting is enabled.
2	PAIR1INVEN	Pair Channels 2 Inverting Enable 0: Inverting is disabled. 1: Inverting is enabled.
1	PAIR2INVEN	Pair Channels 1 Inverting Enable 0: Inverting is disabled. 1: Inverting is enabled.
0	PAIR3INVEN	Pair Channels 0 Inverting Enable 0: Inverting is disabled. 1: Inverting is enabled.

0x94 PWMx_CHOSWCR PWM Channel Software Output Control Register 00000000

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name			CH5S WCV	CH4S WCV	CH3S WCV	CH2S WCV	CH1S WCV	CH0S WCV			CH5S WEN	CH4S WEN	CH3S WEN	CH2S WEN	CH1S WEN	CH0S WEN
Type			RW	RW	RW	RW	RW	RW			RW	RW	RW	RW	RW	RW
Reset			0	0	0	0	0	0			0	0	0	0	0	0

Bit(s)	Name	Description
13	CH5SWCV	Channel 5 Software Output Control Value 0: The software output control forces 0 to the channel output. 1: The software output control forces 1 to the channel output.
12	CH4SWCV	Channel 4 Software Output Control Value 0: The software output control forces 0 to the channel output. 1: The software output control forces 1 to the channel output.
11	CH3SWCV	Channel 3 Software Output Control Value 0: The software output control forces 0 to the channel output. 1: The software output control forces 1 to the channel output.
10	CH2SWCV	Channel 2 Software Output Control Value 0: The software output control forces 0 to the channel output. 1: The software output control forces 1 to the channel output.
9	CH1SWCV	Channel 1 Software Output Control Value 0: The software output control forces 0 to the channel output.

Bit(s)	Name	Description
		1: The software output control forces 1 to the channel output.
8	CH0SWCV	Channel 0 Software Output Control Value 0: The software output control forces 0 to the channel output. 1: The software output control forces 1 to the channel output.
5	CH5SWEN	Channel 5 Software Output Control Enable 0: The channel output is not affected by software output control. 1: The channel output is affected by software output control.
4	CH4SWEN	Channel 4 Software Output Control Enable 0: The channel output is not affected by software output control. 1: The channel output is affected by software output control.
3	CH3SWEN	Channel 3 Software Output Control Enable 0: The channel output is not affected by software output control. 1: The channel output is affected by software output control.
2	CH2SWEN	Channel 2 Software Output Control Enable 0: The channel output is not affected by software output control. 1: The channel output is affected by software output control.
1	CH1SWEN	Channel 1 Software Output Control Enable 0: The channel output is not affected by software output control. 1: The channel output is affected by software output control.
0	CH0SWEN	Channel 0 Software Output Control Enable 0: The channel output is not affected by software output control. 1: The channel output is affected by software output control.

12.3 Functional description

12.3.1 Clock source

The PWM has only one clock domain: the system clock (Timer clock).

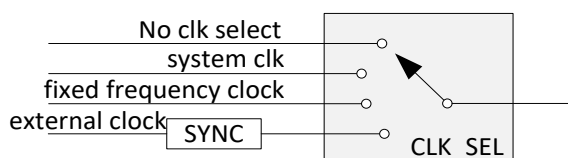


Figure 12-2 PWM clock source

The CLKSRC[1:0] bits in the PWMx_INIT register select one of three possible clock sources for the PWM counter or disable the PWM counter. After any MCU reset, CLKSRC[1:0] = 00, so no clock source is selected. The CLKSRC[1:0] bits may be read or written at any time. Disabling the PWM counter by writing 00 to the CLKSRC[1:0] bits does not affect the PWM counter value or other registers.

The fixed frequency clock is an internal LFOSC clock source for the PWM counter that allows the selection of a clock other than the system clock or an external clock. This clock frequency are almost equal 8 MHz.

The external clock passes through a synchronizer clocked by the system clock to assure that counter transitions are properly aligned to system clock transitions. Therefore, to meet Nyquist criteria

considering also jitter, the frequency of the external clock source must not exceed 1/4 of the system clock frequency.

12.3.2 Prescaler

The selected counter clock source passes through a prescaler that is a 16-bit counter. The value of the prescaler is selected by the PWMx_INIT register PWM_PSC[15:0] bits. The following figure shows an example of the prescaler counter and PWM counter.

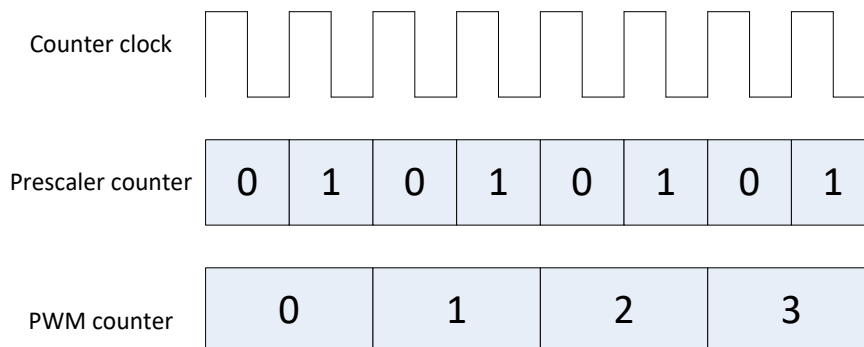


Figure 12-3 Prescaler

12.3.3 Counter

The PWM has a 16-bit counter that is used by the channels either for input or output modes. The PWM counter clock is the selected clock divided by the prescaler. The PWM counter has these modes of operation:

- Up counting.
- Up-down counting.
- Quadrature Decoder mode.

12.3.3.1 Up counting

Up counting is selected when QEIEN=0 and CNTMODE=0. CNTIN defines the starting value of the count and MCVR defines the final value of the count, see the following figure. The value of CNTIN is loaded into the PWM counter, and the counter increments until the value of MCVR is reached, at which point the counter is reloaded with the value of CNTIN. The PWM period when using up counting is $(MCVR - CNTIN + 0x0001) \times \text{period of the PWM counter clock}$. The TOF bit is set to 1 when the PWM counter changes from MCVR to CNTIN.

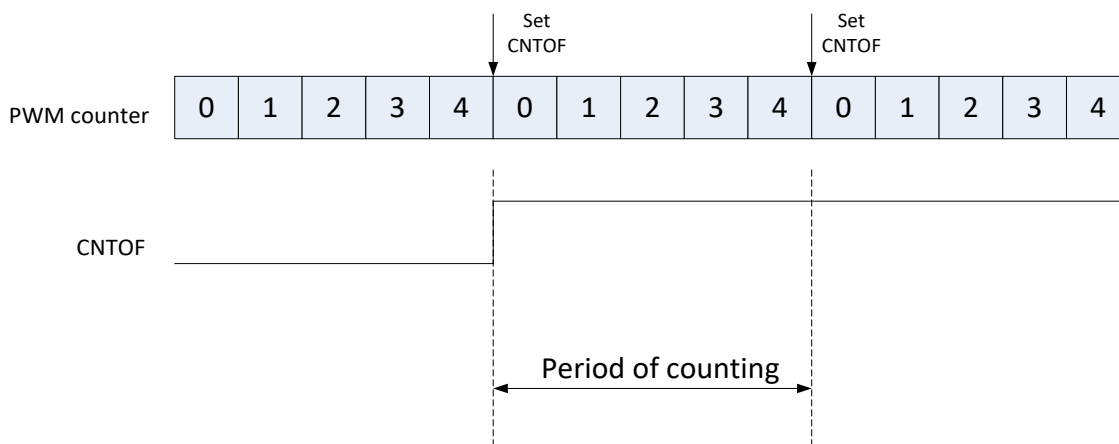


Figure 12-4 Up counting

12.3.3.2 Up-Down counting

Up-down counting is selected when QEIEN = 0 and CNTMODE = 1. CNTIN defines the starting value of the count and MCVR defines the final value of the count. The value of CNTIN is loaded into the PWM counter, and the counter increments until the value of MCVR is reached, at which point the counter is decremented until it returns to the value of CNTIN and the up-down counting restarts.

The PWM period when using up-down counting is $2 \times (\text{MCVR} - \text{CNTIN}) \times \text{period of the PWM counter clock}$. The TOF bit is set to 1 when the PWM counter changes from MCVR to (MCVR – 1). See the following figure.

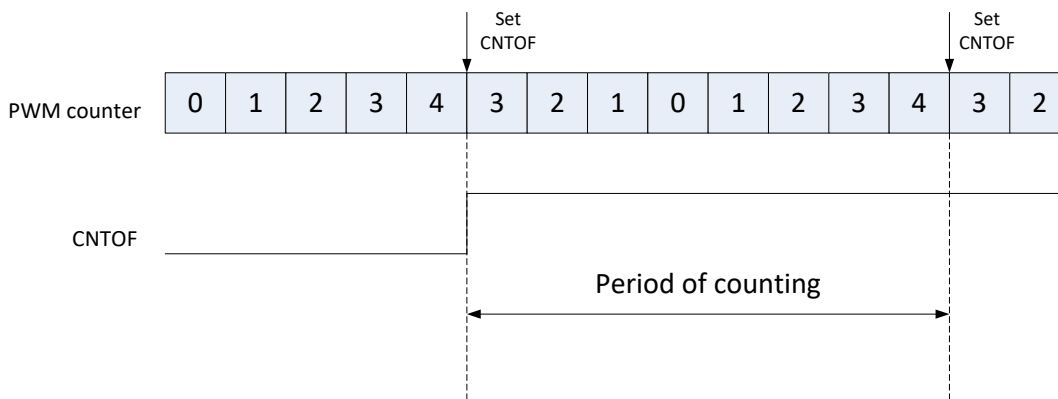


Figure 12-5 Up-Down counting

12.3.4 Operation mode

PWM can be configured for input capture, output compare, or edge-aligned PWM mode, center-aligned PWM mode, combination mode, Quadrature decoder mode. Please refer to the following table in detail.

Table 12-3 Operation mode configuration

PAIRn DECAPEN	PAIRn COMBINEN	CNTMODE	MSR1: MSR0	ELSR1: ELSR0	Operation mode	Configuration
0	0	0	00	01	Input	Capture on Rising

PAIRn DECAPEN	PAIRn COMBINEN	CNTMODE	MSR1: MSR0	ELSR1: ELSR0	Operation mode	Configuration
					capture	Edge Only
				10		Capture on Falling Edge Only
				11		Capture on Rising or Falling Edge
			01	01	Output compare	Toggle Output on match
				10		Clear Output on match
				11		Set Output on match
			1X	10	Edge-aligned PWM	High-true pulses (Clear Output on match)
				X1		High-true pulses (Set Output on match)
		1	XX	10	Center-aligned PWM	High-true pulses (Clear Output on match-up)
				X1		Low-true pulses (Set Output on match-up)
	1	0	XX	10	Combine PWM	High-true pulses (Set on channel (n) match, and clear on channel (n+1) match)
				X1		Low-true pulses (Clear on channel (n) match, and set on channel (n +1) match)
1	0	0	X0	See the following table	Dual edge capture	One-Shot Capture mode
			X1			Continuous Capture mode

ELSR1	ELSR0	Channel Port Enable	Detected Edges
0	0	Disabled	No edge
0	1	Enabled	Rising edge

ELSR1	ELSR0	Channel Port Enable	Detected Edges
1	0	Enabled	Falling edge
1	1	Enabled	Rising and falling edges

12.3.5 Input capture mode

The Input Capture mode is selected when:

- DECAPEN = 0
- COMBINE = 0
- CNTMODE = 0
- MSnR1:MSnR0 = 0:0
- ELSnR1:ELSnR0 != 0:0

When a selected edge occurs on the channel input, the current value of the PWM counter is captured into the CHnV register. At the same time, the CHnIF bit is set and the channel interrupt is generated if enabled by CHnIE = 1. When a channel is configured for input capture, the PWMxCHn pin is an edge-sensitive input. ELSnR1:ELSnR0 control bits determine which edge, falling or rising, triggers input capture event. Note that the maximum frequency for the channel input signal to be detected correctly is system clock divided by 4, which is required to meet Nyquist criteria for signal sampling. Writes to the CHnV register is ignored in Input Capture mode.

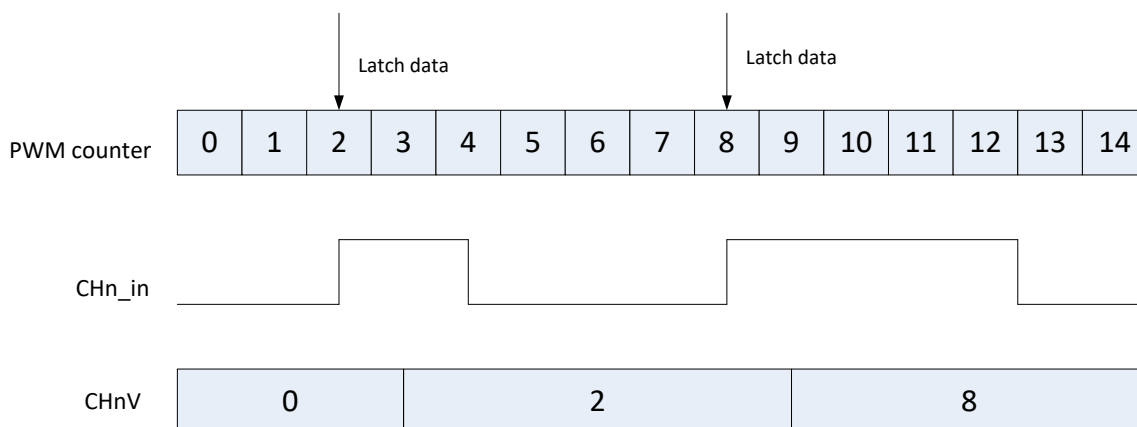


Figure 12-6 Input Capture mode

For PWM2 Channel0 to Channel3, there exist additional capture filter to make input signal clean. The filter is 5-bit counter and can be configured through register CHnCAPFVAL(n = 0,1,2,3). When the CHnCAPFVAL[4:0] = 0, the filter function is disabled, if CHnCAPFVAL[4:0] ≠ 00000, the input signal will be delayed CHnCAPFVAL[4:0] x 4 system Clock, and then be transported to the input capture function.

Please note that only PWM2 exists input capture filter function.

12.3.6 Output Compare mode

The Output Compare mode is selected when:

- DECAPEN = 0
- COMBINE = 0
- CNTMODE = 0, and

- MSnR1:MSnR0 = 0:1

In Output Compare mode, the PWM can generate timed pulses with programmable position, polarity, duration, and frequency. When the counter matches the value in the CHnV register of an output compare channel, the channel (n) output can be set, cleared, or toggled. When a channel is initially configured to Toggle mode, the previous value of the channel output is held until the first output compare event occurs. The CHnIF bit is set and the channel (n) interrupt is generated if CHnIE = 1 at the channel(n) match (PWM counter = CHnV).

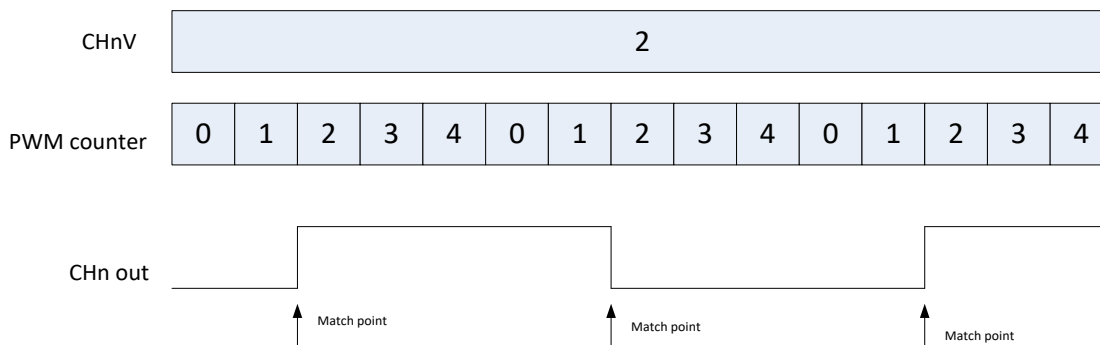


Figure 12-7 Output Compare mode

12.3.7 Edge-Aligned PWM(EPWM) mode

The Edge-Aligned mode is selected when:

- QEIE = 0
- DECAPEN = 0
- COMBINE = 0
- CNTMODE = 0, and
- MSnR1 = 1

The EPWM period is determined by (MCVR - CNTIN + 0x0001) and the pulse width (duty cycle) is determined by (CHnV - CNTIN). The CHnIF bit is set and the channel (n) interrupt is generated if CHnIE = 1 at the channel (n) match (PWM counter = CHnV), that is, at the end of the pulse width. This type of PWM signal is called edge-aligned because the leading edges of all PWM signals are aligned with the beginning of the period, which is the same for all channels within an PWM.

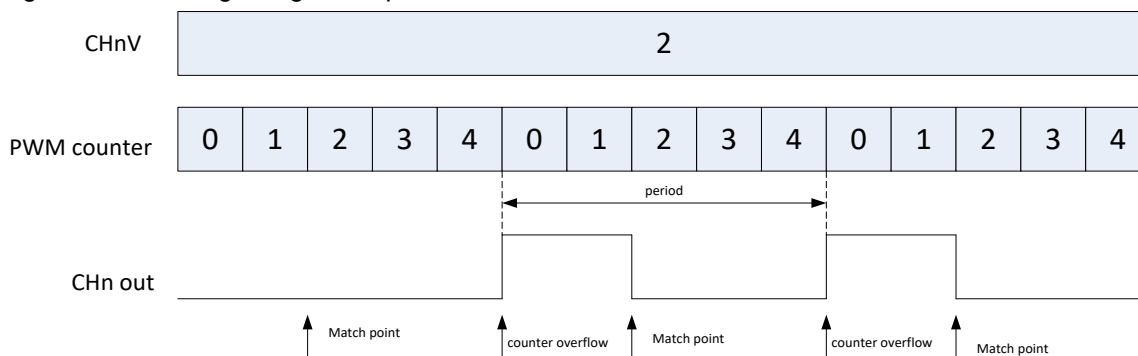


Figure 12-8 Edge-aligned PWM mode

12.3.8 Center-Aligned PWM(CPWM) mode

The Center-Aligned mode is selected when:

- QEIEN = 0
- DECAPEN = 0
- COMBINE = 0, and
- CNTMODE = 1

The CPWM pulse width (duty cycle) is determined by $2 \times (\text{CHnV} - \text{CNTIN})$ and the period is determined by $2 \times (\text{MCVR} - \text{CNTIN})$. See the following figure. MCVR must be kept in the range of 0x0001 to 0x7FFF because values outside this range can produce ambiguous results. In the CPWM mode, the PWM counter counts up until it reaches MCVR and then counts down until it reaches CNTIN. The CHnIF bit is set and channel (n) interrupt is generated (if CHnIE = 1) at the channel (n) match (PWM counter = CHnV) when the PWM counting is down (at the begin of the pulse width) and when the PWM counting is up (at the end of the pulse width). This type of PWM signal is called center-aligned because the pulse width centers for all channels are aligned with the value of CNTIN.

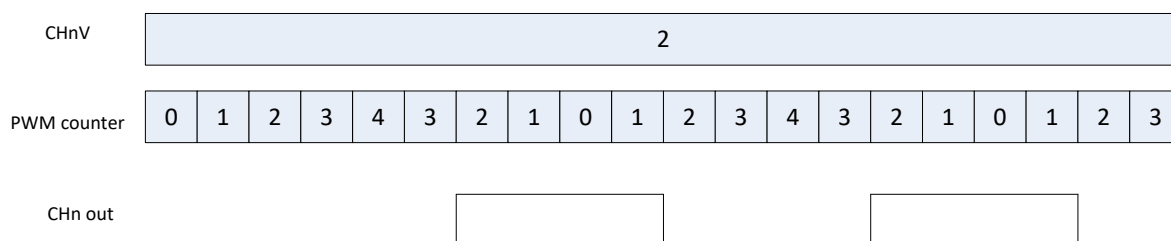


Figure 12-9 Center-aligned PWM mode

12.3.9 Combine mode

The Combine mode is selected when:

- PWMEN2 = 1
- QEIEN = 0
- DECAPEN = 0
- COMBINE = 1, and
- CNTMODE = 0

In Combine mode, an even channel (n) and adjacent odd channel (n+1) are combined to generate a PWM signal in the channel (n) output. In the Combine mode, the PWM period is determined by $(\text{MCVR} - \text{CNTIN} + 0x0001)$ and the PWM pulse width (duty cycle) is determined by $(|\text{CH}(n+1)\text{V} - \text{CH}(n)\text{V}|)$. The CHnIF bit is set and the channel (n) interrupt is generated (if CHnIE = 1) at the channel (n) match (PWM counter = CH(n)V). The CH(n+1)F bit is set and the channel (n+1) interrupt is generated, if CH(n+1)IE = 1, at the channel (n+1) match (PWM counter = CH(n+1)V). If (ELSnR1:ELSnR0 = 1:0), then the channel (n) output is forced low at the beginning of the period (PWM counter = CNTIN) and at the channel (n+1) match (PWM counter = CH(n+1)V). It is forced high at the channel (n) match (PWM counter = CH(n)V). See the following figure. If (ELSnR1:ELSnR0 = X:1), then the channel (n) output is forced high at the beginning of the period (PWM counter = CNTIN) and at the channel (n+1) match (PWM counter = CH(n+1)V). It is forced low at the channel (n) match

(PWM counter = CH(n)V). See the following figure. In Combine mode, the ELS(n+1)R1 and ELS(n+1)R0 bits are not used in the generation of the channels (n) and (n+1) output.

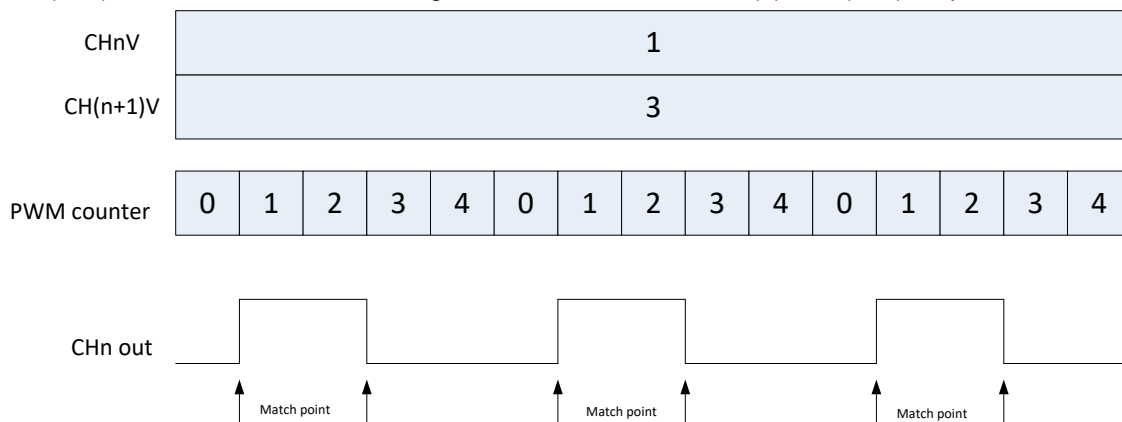


Figure 12-10 Combine mode

12.3.10 Complementary mode

The Complementary mode is selected when:

- PWMEN2 = 1
- QEIEN = 0
- DECAPEN = 0
- COMBINE = 1
- CNTMODE = 0, and
- COMP = 1

In Complementary mode, the channel (n+1) output is the inverse of the channel (n) output. If (PWMEN2 = 1), (QEIEN = 0), (DECAPEN = 0), (COMBINE = 1), (CNTMODE = 0), and (COMP = 0), then the channel (n+1) output is the same as the channel (n) output.

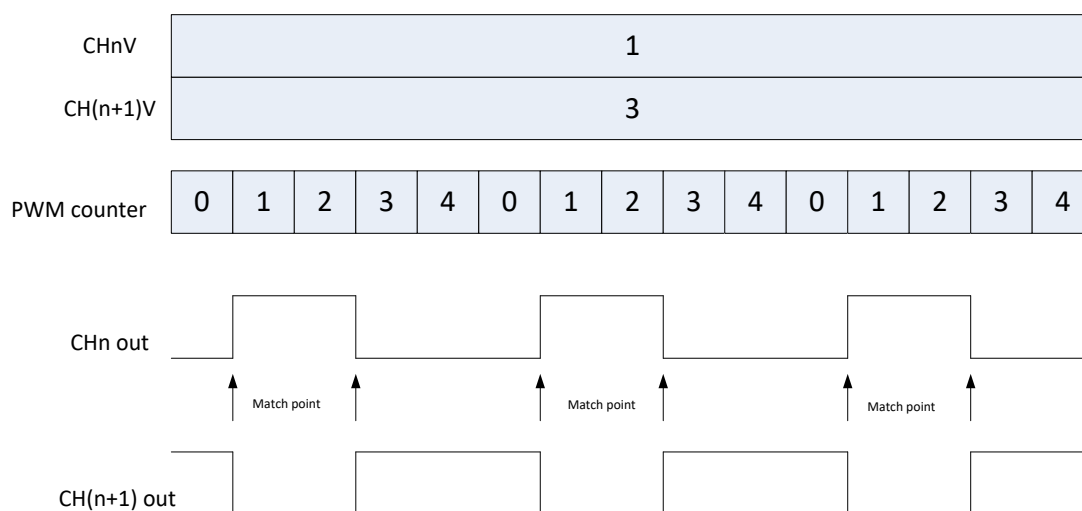


Figure 12-11 Complementary mode

12.3.11 Registers updated from write buffers

12.3.11.1 PWM_CNTIN register update buffer

Table 12-4 PWM_CNTIN register update buffer

Condition	Update opportunity
CLKSRC[1:0] = 0:0	When PWM_CNTIN register is written
CLKSRC[1:0] \neq 0:0 & PWMSYNCEN = 0	At the next system clock after PWM_CNTIN was written
CLKSRC[1:0] \neq 0:0 & PWMSYNCEN = 1	By the PWM_CNTIN register synchronization

12.3.11.2 PWM_CH(n)V register update buffer

Table 12-5 PWM_CH(n)V register update buffer

Condition	Update opportunity
CLKSRC[1:0]=0:0	When PWM_CH(n)V register is written
CLKSRC[1:0] \neq 0:0 & PWMSYNCEN = 0	<ul style="list-style-type: none"> If the selected mode is EPWM, then register is updated after the PWM counter changes from MCVR to CNTIN. If the selected mode is CPWM, then register is updated after the PWM counter changes from MCVR to (MCVR – 0x0001).
CLKSRC[1:0] \neq 0:0 & PWMSYNCEN = 1	By the PWM_CH(n)V register synchronization

12.3.11.3 PWM_MCVR register update buffer

Table 12-6 PWM_MCVR register update buffer

Condition	Update opportunity
CLKSRC[1:0]=0:0	When PWM_MCVR register is written
CLKSRC[1:0] \neq 0:0 & PWMSYNCEN = 0	<ul style="list-style-type: none"> If the selected mode is EPWM, then register is updated after the PWM counter changes from MCVR to CNTIN. If the selected mode is CPWM, then register is updated after the PWM counter changes from MCVR to (MCVR – 0x0001).
CLKSRC[1:0] \neq 0:0 & PWMSYNCEN = 1	By the PWM_MCVR register synchronization

12.3.12 PWM synchronization

The PWM synchronization provides an opportunity to update the MCVR, CNTIN, CHnV, OMCR, INVCr and CHOSWCR registers with their buffered value and force the PWM counter to the CNTIN register value.

Note:

- The PWM synchronization must be used only in Combine mode.
- The legacy PWM synchronization (SYNCMODE = 0) is a subset of the enhanced PWM synchronization (SYNCMODE = 1). Thus, only the enhanced PWM synchronization must be used.

12.3.12.1 Hardware trigger

Three hardware trigger signal inputs of the PWM module are enabled when TRIGN = 1, where n = 0, 1 or 2 corresponding to each one of the input signals, respectively. The hardware trigger input n is synchronized by the system clock. The PWM synchronization with hardware trigger is initiated when a rising edge is detected at the enabled hardware trigger inputs. If (HWTRIGMODESEL = 0) then the TRIGN bit is cleared when 0 is written to it or when the trigger n event is detected. In this case, if two or more hardware triggers are enabled (for example, TRIG0 and TRIG1 = 1) and only trigger 1 event occurs, then only TRIG1 bit is cleared. If a trigger event occurs together with a write setting TRIGN bit, then the synchronization is initiated, but TRIGN bit remains set due to the write operation.

12.3.12.2 Software trigger

A software trigger event occurs when 1 is written to the SYNC[SWSYNC] bit. The SWSYNC bit is cleared when 0 is written to it or when the PWM synchronization, initiated by the software event, is completed.

If another software trigger event occurs (by writing another 1 to the SWSYNC bit) at the same time, the PWM synchronization initiated by the previous software trigger event is ending, a new PWM synchronization is started and the SWSYNC bit remains equal to 1.

If SYNCMODE = 0, then the SWSYNC bit is also cleared by PWM according to PWMSYNC and REINIT bits. In this case, if (PWMSYNC = 1) or (PWMSYNC = 0 and REINIT = 0) then SWSYNC bit is cleared at the next selected loading point after that the software trigger event occurred.

If (PWMSYNC = 0) and (REINIT = 1), then SWSYNC bit is cleared when the software trigger event occurs.

If SYNCMODE = 1, then the SWSYNC bit is also cleared by PWM according to the CNTVSWSYNC bit. If CNTVSWSYNC=0, then SWSYNC bit is cleared at the next selected loading point after that the software trigger event occurred. If CNTVSWSYNC = 1, then SWSYNC bit is cleared when the software trigger event occurs.

12.3.13 Inverting

The Invert functionality swaps the signals between channel(n) and channel(n+1) outputs. The Inverting operation is selected when:

- PWMEN2 = 1
- QEIEN = 0
- DECAPEN = 0
- COMBINE = 1
- CNTMODE = 0, and

- $PAIR(n)INVEN = 1$, $n=0,1,2$ and 3

12.3.14 Channel trigger output

If $CHjTRIG = 1$, where $j = 0, 1, 2, 3, 4$, or 5 , then the PWM generates a trigger when the channel (j) match occurs ($PWM\ counter = CH(j)V$). The channel trigger output provides a trigger signal that is used for on-chip modules.

Functional description: The PWM is able to generate multiple triggers in one PWM period. Because each trigger is generated for a specific channel, several channels are required to implement this functionality.

12.3.15 Initialization trigger

If $INITTRIGEN = 1$, then the PWM generates a trigger when the PWM counter is updated with the $CNTIN$ register value in the following cases.

- The PWM counter is automatically updated with the $CNTIN$ register value by the selected counting mode.
- When there is a write to CNT register.
- When there is the PWM counter synchronization.

12.3.16 Capture Test mode

The Capture Test mode allows to test the $CHnV$ registers, the PWM counter and the interconnection logic between the PWM counter and $CHnV$ registers. In this test mode, all channels must be configured for Input Capture mode and PWM counter must be configured to the Up counting.

When the Capture Test mode is enabled ($CAPTEST = 1$), the PWM counter is frozen and any write to CNT register updates directly the PWM counter. After it was written, all $CHnV$ registers are updated with the written value to CNT register and $CHnIF$ bits are set. Therefore, the PWM counter is updated with its next value according to its configuration. Its next value depends on $CNTIN$, $MCVR$, and the written value to PWM counter. The next reads of $CHnV$ registers return the written value to the PWM counter and the next reads of CNT register return PWM counter next value.

12.3.17 Dual Edge Capture mode

The Dual Edge Capture mode is selected if $PWMEN2 = 1$ and $DECAPEN = 1$. This mode allows to measure a pulse width or period of the signal on the input of channel (n). In the mode, only channel (n) input is used and channel ($n+1$) input is ignored. The channel (n) filter can be active in this mode when n is 0 or 2. The $ELS(n)R1:ELS(n)R0$ bits select the edge that is captured by channel (n), and $ELS(n+1)R1:ELS(n+1)R0$ bits select the edge that is captured by channel ($n+1$).

If both $ELS(n)R1:ELS(n)R0$ and $ELS(n+1)R1:ELS(n+1)R0$ bits select the same edge, then it is the period measurement. If these bits select different edges, then it is a pulse width measurement. If the selected edge by channel (n) bits is detected at channel (n) input, then $CH(n)IF$ bit is set and the channel (n) interrupt is generated (if $CH(n)IE = 1$). The Dual edge capture does not support enabling channel (n) and channel ($n+1$) interrupts simultaneously.

If the selected edge by channel ($n+1$) bits is detected at channel (n) input and ($CH(n)IF = 1$), then $CH(n+1)IF$ bit is set and the channel ($n+1$) interrupt is generated (if $CH(n+1)IE = 1$). The $CH(n)V$ register stores the value of PWM counter when the selected edge by channel(n) is detected at channel (n) input. The $CH(n+1)V$ register stores the value of PWM counter when the selected edge by

channel (n+1) is detected at channel (n) input. In this mode, a coherency mechanism ensures coherent data when the CH(n)V and CH(n+1)V registers are read. The only requirement is that CH(n)V must be read before CH(n+1)V.

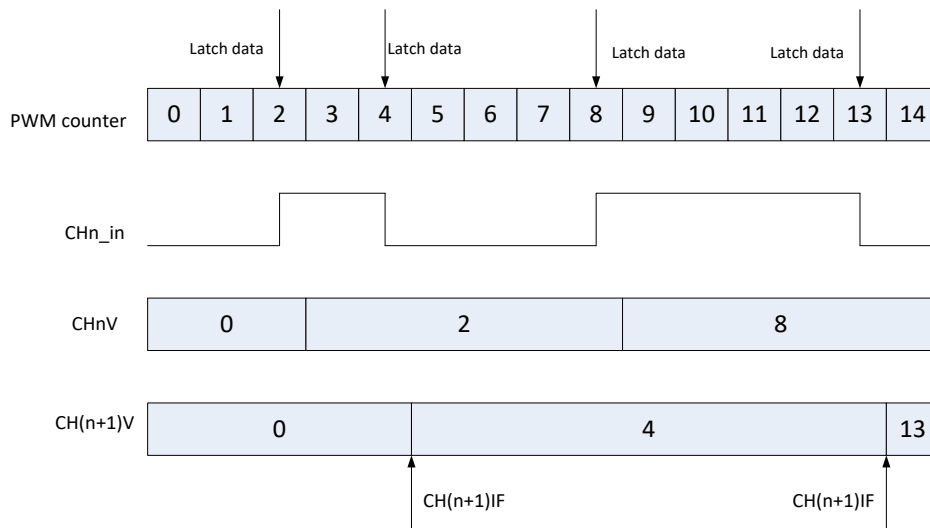


Figure 12-12 Dual edge capture mode

12.3.18 Software output control

The software output control forces the channel output according to software defined values at a specific time in the PWM generation.

The software output control is selected when:

- PWMEN2 = 1
- DECAPEN = 0
- COMBINE = 1
- CNTMODE = 0, and
- CH(n)SWEN = 1, n=0,1,2,3,4 and 5

The CH(n)SWEN bit enables the software output control, and the CH(n)SWCV selects the value that is forced to this channel output. Software output control forces the following values on channels(n) and (n+1) when the COMP bit is zero.

Table 12-7 Software output control when COMP bit is zero

CH(n)SWE N	CH(n+1)SWE N	CH(n)SW CV	CH(n+1)SWC V	Channel(n)Outp ut	Channel(n+1)Outp ut
0	0	X	X	Software control not enable	Software control not enable
1	1	0	0	0	0
1	1	0	1	0	1
1	1	1	0	1	0
1	1	1	1	1	1

Software output control forces the following values on channels(n) and (n+1) when the COMP bit is one:

Table 12-8 Software output control when COMP bit is one

CH(n)SWE N	CH(n+1)SWE N	CH(n)SW CV	CH(n+1)SWC V	Channel(n)Outp ut	Channel(n+1)Outp ut
0	0	X	X	Software control not enable	Software control not enable
1	1	0	0	0	0
1	1	0	1	0	1
1	1	1	0	1	0
1	1	1	1	1	0

12.3.19 Deadtime insertion

The deadtime insertion is enabled when (DTEN = 1) and (DTVAL[5:0] is non-zero). DTSET register defines the deadtime delay that can be used for all PWM channels. The DTPSC[1:0] bits define the prescaler for the system clock and the DTVAL[5:0] bits define the deadtime modulo, that is, the number of the deadtime prescaler clocks. The deadtime delay insertion ensures that no two complementary signals (channels (n) and (n+1)) drive the active state at the same time.

If CH(n)POL = 0, CH(n+1)POL = 0, and the deadtime is enabled, then when the channel (n) match (PWM counter = C(n)V) occurs, the channel (n) output remains at the low value until the end of the deadtime delay when the channel (n) output is set. Similarly, when the channel (n+1) match (PWM counter = C(n+1)V) occurs, the channel (n+1) output remains at the low value until the end of the deadtime delay when the channel (n+1) output is set.

If CH(n)POL = 1, CH(n+1)POL = 1, and the deadtime is enabled, then when the channel (n) match (PWM counter = CH(n)V) occurs, the channel (n) output remains at the high value until the end of the deadtime delay when the channel (n) output is cleared. Similarly, when the channel (n+1) match (PWM counter = CH(n+1)V) occurs, the channel (n+1) output remains at the high value until the end of the deadtime delay when the channel (n+1) output is cleared.

12.3.20 Fault control

There exists 4 Fault input Source, 2 internal and 2 external, ACMP0_OUT and ACMP1_OUT can be chosen for Internal Fault Input, while external Fault1 and Fault2 can be chosen for external Fault Input.

Table 12-9 Fault source and number table

Fault Source	Fault Input Number	Comment
ACMP0_OUT	FAULT0	Internal Fault Input
External Fault1	FAULT1	External Fault Input
External Fault2	FAULT2	External Fault Input
ACMP1_OUT	FAULT3	Internal Fault Input

12.3.20.1 Automatic fault clearing

If the automatic fault clearing is selected (FAULTMODE[1:0]=1:1), then the channels output disabled by fault control is again enabled when the fault input signal returns to zero and a new PWM cycle begins.

12.3.20.2 Manual fault clearing

If the manual fault clearing is selected (FAULTMODE[1:0]=0:1 or 1:0), then the channels output disabled by fault control is again enabled when the FAULTDF bit is cleared and a new PWM cycle begins.

12.3.20.3 Fault inputs polarity control

The FLTNPOL bit selects the fault input n polarity, when n=0,1,2 and 3:

- (1) If FLTNPOL = 0, the fault n input polarity is high, so the logical one at the fault input n indicates a fault.
- (2) If FLTNPOL = 1, the fault n input polarity is low, so the logical zero at the fault input n indicates a fault.

12.3.21 Polarity control

The CHnPOL bit selects the channel(n) output polarity, when n=0,1,2,3,4 and 5:

- (1) If CHnPOL = 0, the channel(n) output polarity is high, so the logical one is the active state and logical zero is the inactive state.
- (2) If CHnPOL = 1, the channel(n) output polarity is low, so the logical zero is the active state and logical one is the inactive state.

12.3.22 Features priority

The following figure shows the priority of the features used at the generation of CH(n) and CH(n+1) outputs signals.

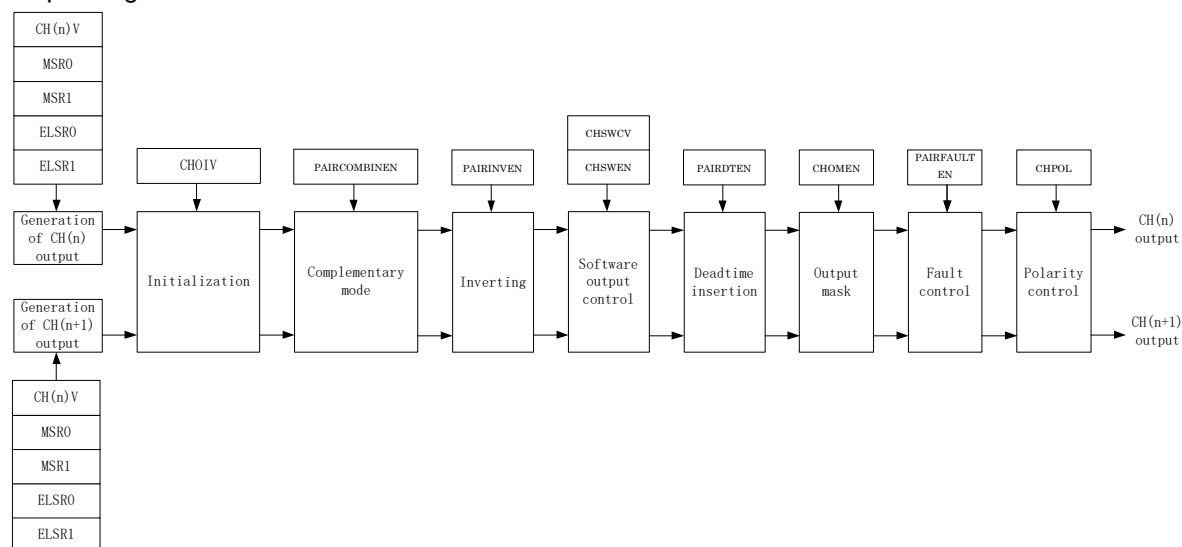


Figure 12-13 Features priority

12.4 PWM interrupts

12.4.1 Count Overflow interrupt

The Count Overflow interrupt is generated when (CNT0IE=1) and (CNT0F=1).

12.4.2 Channel interrupt

The Channel(n) interrupt is generated when (CHnIE=1) and (CHnIF = 1).

12.4.3 Fault interrupt

The Fault interrupt is generated when (FAULTIE =1) and (FAULTDF = 1).

13 Pulse Width Detect Timer(PWDT)

13.1 Introduction

PWDT (Pulse Width Detect Timer) module is used as a tool to measure the pulse width or as a 16-bit timer.

13.2 Features

- 2 selectable clock sources: bus clock and back-up clock.
- 4 selectable pulse input.
- Support 2 functions: pulse width measurement function and timer function.
 - For pulse width measurement function:
 - Programmable start trigger edge.
 - 4 programmable measurement modes.
 - Support 3 hall sensors' signal input measurement.
 - Support 3 inputs from analog comparator.
 - For timer function:
 - modify the timer load value when timer is disabled or on normal operation.
- 16-bit counter used for pulse width measurement or timer function.
- Interrupts:
 - OVF: PWDT counter overflows on pulse width measurement function or timer function.
 - RDYF: PWDT pulse width measurement value update.

13.3 Functional description

The following is the block diagram of the PWDT Module.

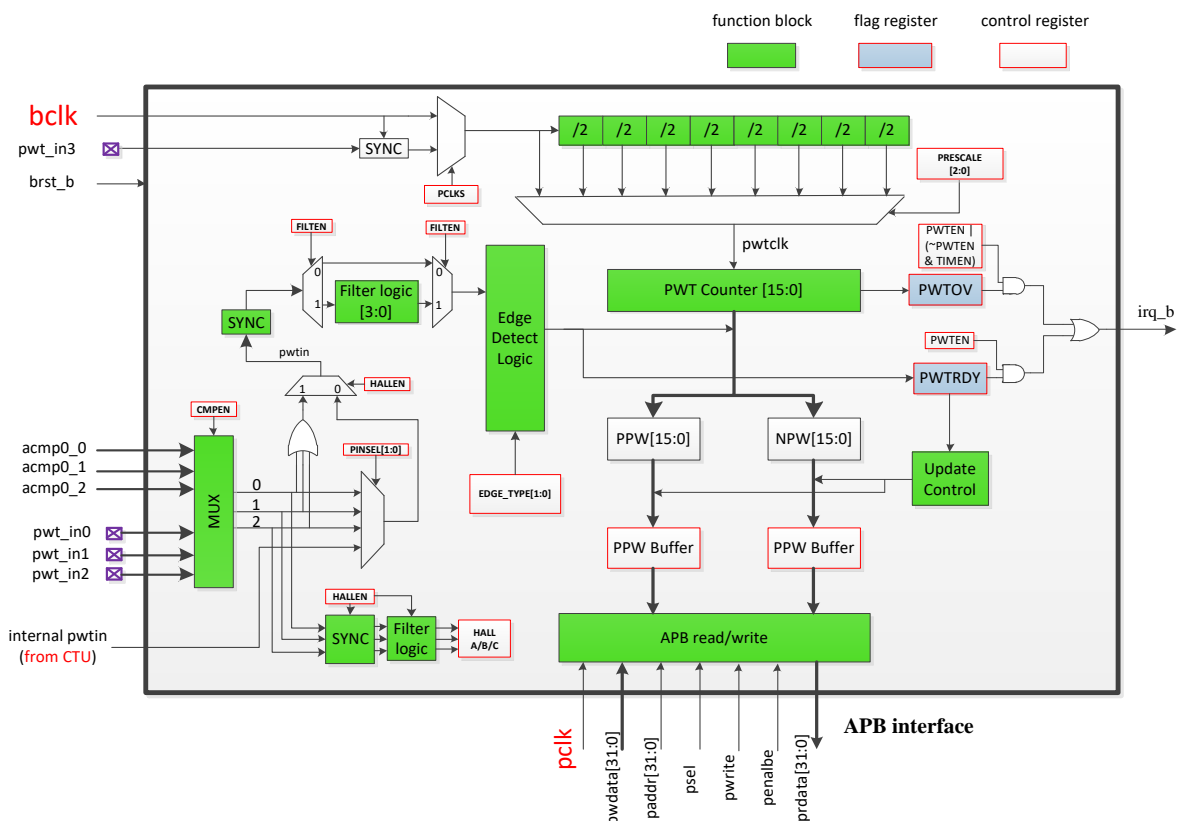


Figure 13-1 PWDT block diagram

13.3.1 Pulse Width Measurement function

To get the pulse width measurement value, the PWDT counter runs based on the **pwtclk** described in block diagram after **PWDTEN=1**. And **pwtclk** is deriving from the bus clock division by the **PSC[2:0]**. So, in order to get more accurate measurement value, user had better take the smaller value of **PSC[2:0]** for narrower pulse input and larger value of **PSC[2:0]** for wider pulse input. The measurable pulse width is listed in [Table 13-1](#).

Table 13-1 Measurable pulse width range

Item	Clock	Time
Measurable pulse width range	4 bclk to 128*65535 bclk	0.08us to 0.168s

Firstly, for pulse width measurement, user must clearly know the application scenarios, which mainly include two conditions: one is just single channel input for basic measurement and the other is 3 channel inputs for hall measurement. For basic measurement, user choose to measure specific channel input by setting **PINSEL[1:0]** and can choose one of the 4 measurement modes by setting **EDGE[1:0]** based on the practical applications.

For hall measurement, the module measure the pulse input deriving from XOR of the 3 channel inputs and user should set the **EDGE[1:0]=2'b01**. Further, the configuration of the internal 3 channel comparator inputs is similar to the hall measurement and just **CMPEN** should be configured to 1'b1 simultaneously.

For basic measurement, 4 measurement modes can be selected based on practical applications and are illustrated as [Figure 13-2](#).

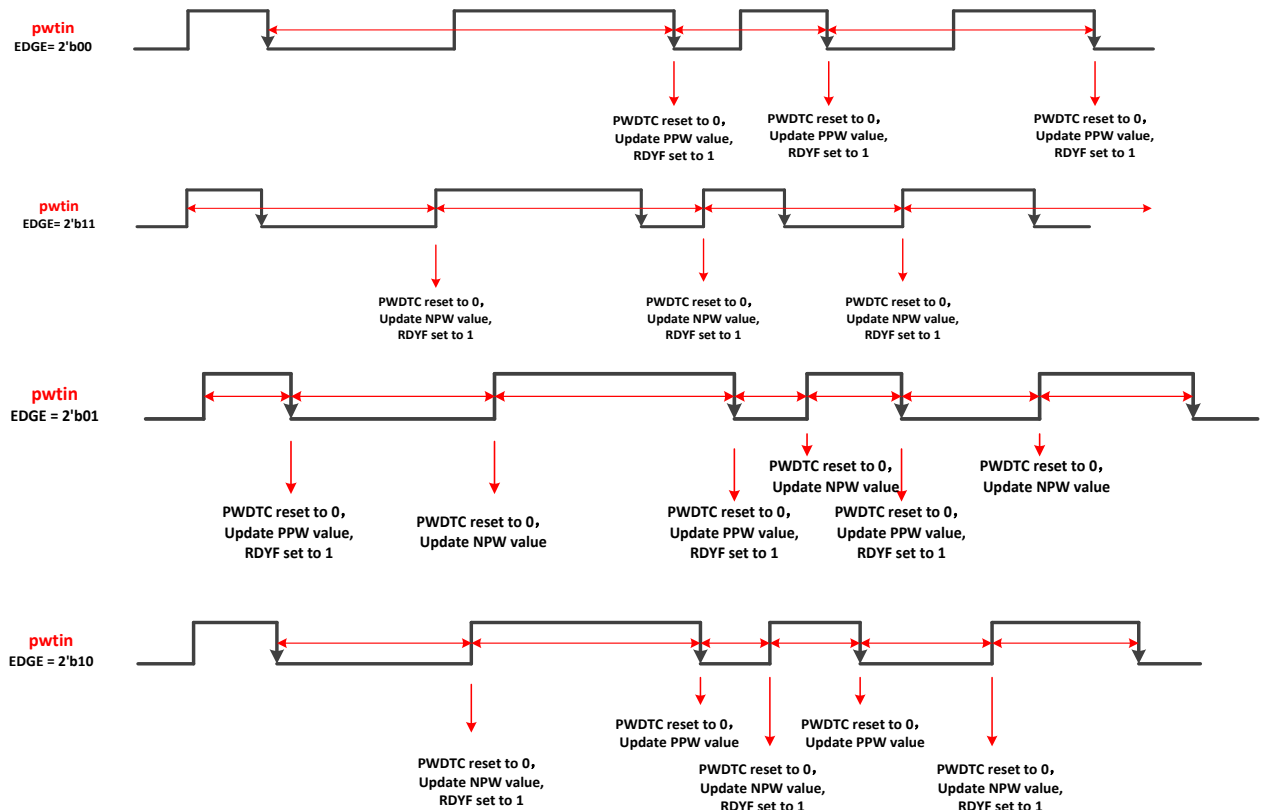


Figure 13-2 Four basic measurement modes

For hall measurement or 3 internal comparator inputs, just one mode can be selected for motor speed calculation or commutation and is illustrated as [Figure 13-3](#).

In the motors, installation of hall devices is used for detecting location of the rotor to commutate properly. Hall devices are usually installed in two ways, as shown in [Figure 13-4](#). One is 120 electrical degree interval and the other is 60 electrical degree interval.

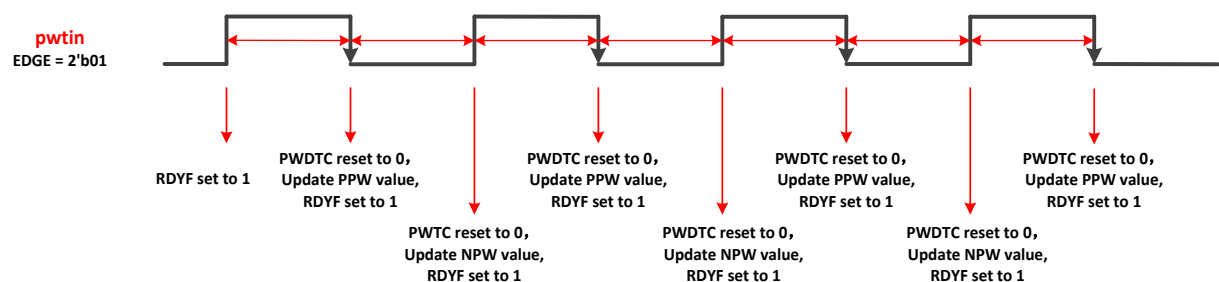


Figure 13-3 Hall measurement modes

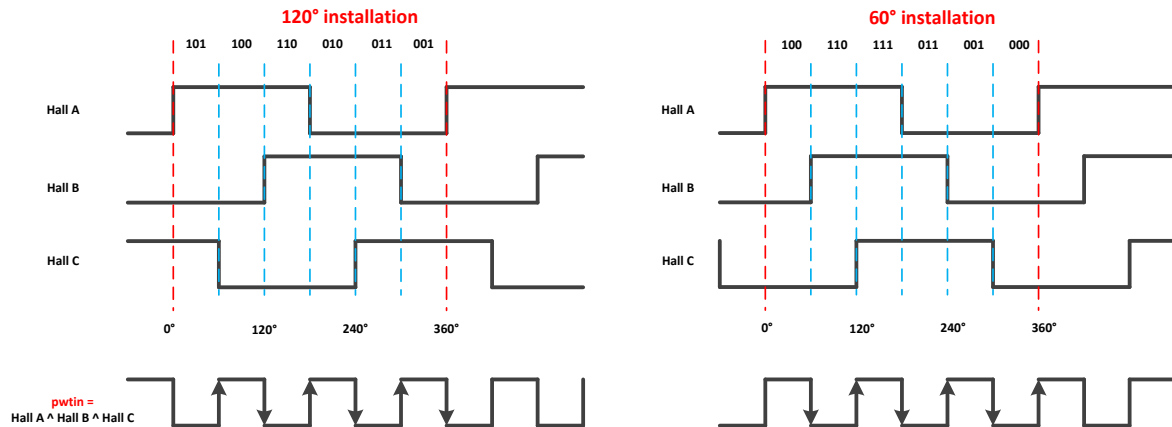


Figure 13-4 Two common installation ways

Secondly, filter in module is designed for filter the noise signal which high or low level is less than specific width described in the paragraph. The `FILT_PSC[7:4]` and `FILTVAL[3:0]` settings determine the maximum and minimum noise pulse width. The [Figure 13-5](#) and [Figure 13-6](#) introduce the noise width settings, judgement and filter. When user configures the `FILTVAL=15` and `FILT_PSC=2`, the filter pulse width is 60 bclk and pulses less than 60 bclk are judged as noise pulse and will be filtered. The filterable pulse width is listed in [Table 13-2](#).

Table 13-2 Filterable pulse width range

Item	Clock	Time
Filterable pulse width range	4 bclk to 16*4096 bclk	0.08us to 1.311ms

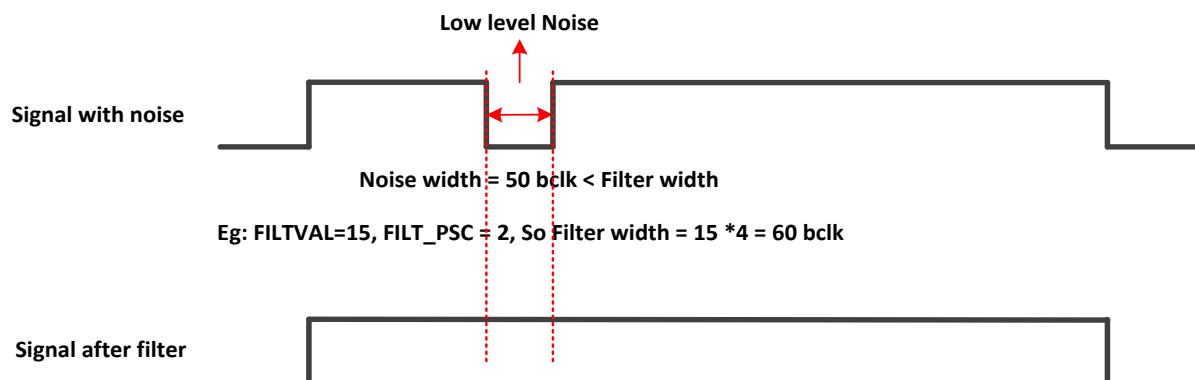


Figure 13-5 Example for low level noise and filter

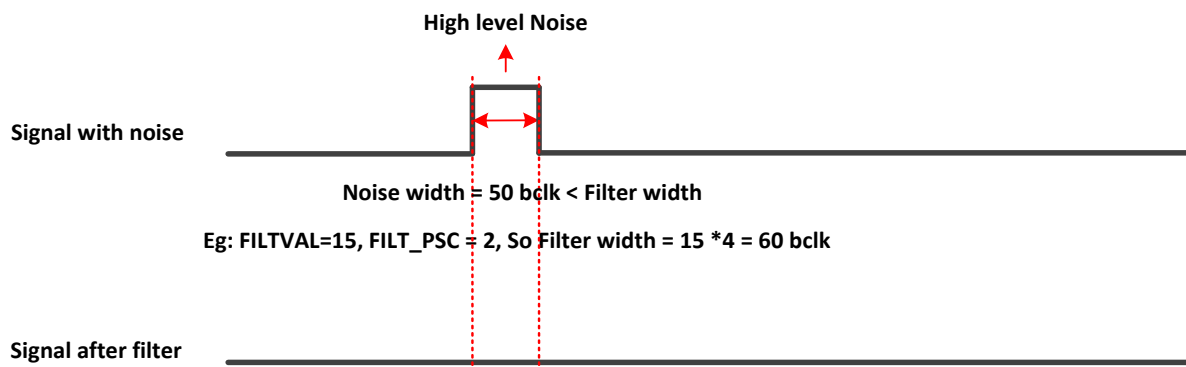


Figure 13-6 Example for high level noise and filter

Thirdly, RDYF and OVF status or interrupt when corresponding enable is 1'b1 are used for pulse width measurement function. The RDYF status occurs under the condition described above. The OVF status occurs when the PWDTC counter overflows.

At last, user should understand the measurement accuracy for pulse width measurement to configure proper value of PSC[2:0] to reach a more accurate measurement value. A basic principle is that more accurate measurement value can be gotten with smaller PSC[2:0]. Obviously, more narrow input pulse, more relative measurement error. The [Figure 13-7](#) describes the error when it operates for pulse width measurement function. In the [Figure 13-7](#), the PWDTC counter and the pwtclk divisor counter reset to 0 simultaneously when the pwtin pulse changes from high to low or from low to high. And exactly here the counting error occurs which is deriving from the last counting value illustrated in [Figure 13-7](#). The practical width value is less than measurement value by less than a pwtclk period.

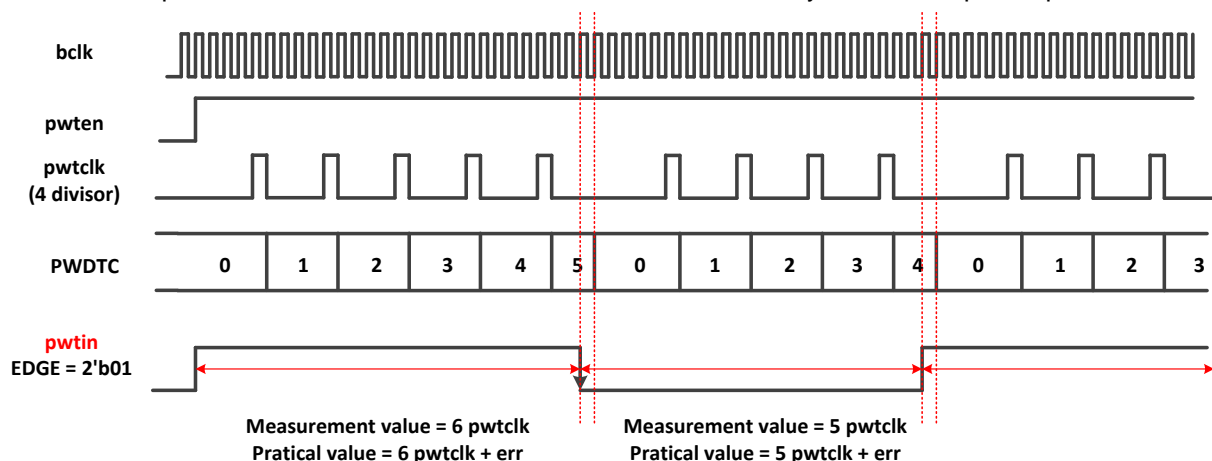


Figure 13-7 PWDTC counter and counting error

13.3.2 Timer function

For timer function, just OVF status is valid and occurs when PWDTC counter overflows. The counter load value TIMCNTVAL[15:0] can be modified all the time, but different time of modifying the value leads to different operation in microcosm in [Figure 13-8](#) and [Figure 13-9](#).

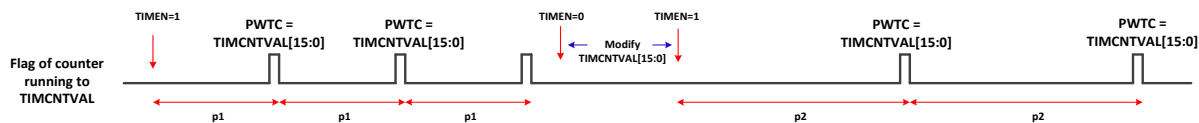


Figure 13-8 Modify the TIMCNTVAL between the TIMEN=0 and TIMEN=1

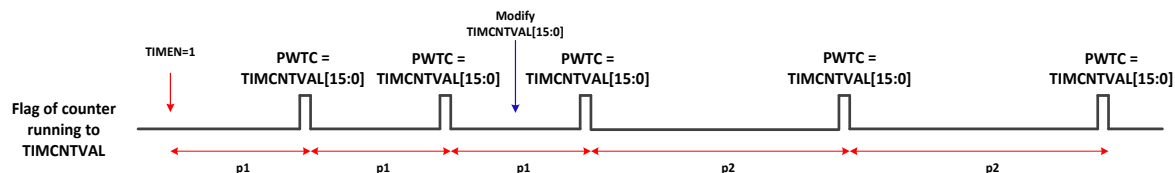


Figure 13-9 Modify the TIMCNTVAL during TIMEN=1

13.3.3 Reset operation

Here just introduce the module software reset operation. User writes 1 to bit SR can trigger a software reset for the module. Once software reset is triggered, the following operations will occur:

- PWDTC reset to 0.
- PWDTC divisor reset to 0.
- Edge detecting logic reset.
- Width latch mechanism reset.
- PPWCV and NPWCV reset to 0.
- RDYF and OVF reset to 0.
- Other control bits don't reset.

Meanwhile, Configuring the PWDTEN to 0 can reach the same goal of reset above.

13.4 Program guide

13.4.1 Pulse Width Measurement function program guide

User must keep in mind that PWDTEN should be configured to 1 after all other controlling bits. Otherwise, abnormal condition may occur. Especially, HALLEN and CMPEN should be configured to 1 for the internal 3 comparator inputs.

13.4.2 Timer function program guide

Timer function is used easily with just configuration of the TIMCNTVAL, PRESCALE and TIMEN and so on. And user should set TIMEN to 1 at last and must not set the PWDTEN to 1 because the pulse width measurement function is prior to timer function.

13.5 Register definition

Table 13-3 PWDT register map and reset values

Base address: 0x40017000

Address	Name	Width	Register function
Base address+0x00000000	PWDT_INIT0	32	General control, status bits and positive pulse width contents
Base address+0x00000004	PWDT_NPWCV	32	Negative pulse width content and 16-bit free running counter
Base address+0x00000008	PWDT_INIT1	32	Hall function control and timer function control

00000000 PWDT_INIT0

PWDT Initialization Register 0

00000000

Bit	31~16	15	14	13~12	11	10	9	8	7	6	5	4	3	2	1	0
Name	PPWCV	PCLKSEL		PINSEL	EDGE		PSC			PWDTE N	IE	PRDY IE	OVI E	SR	RDYF	OV F
Type	R	RW		RW	RW		RW			RW	RW	RW	RW	W	R	R
Reset	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0

Note: PWDTE N (pwm function) enable prior to TIMEN (timer function), so when timer function is going to be enabled, PWDTE N must be disabled.

Bit(s)	Mnemonic	Name	Description
31:16	PPWCV	PPWCV	Positive Pulse Width or Timer Count Value Indicate the negative pulse width value.
15	PCLKSEL	PCLKSEL	PWDT Clock source selection 0: timer clock for PWDT counter. 1: alternative clock for PWDT counter.
13~12	PINSEL	PINSEL	Pin selection Note: Pwdt_in3 is from the CTU module internally. 00/01/10/11: respectively select pwdt_in0/ pwdt_in1/ pwdt_in2/ internal_pwdtin.
11~10	EDGE	EDGE	Selects the input edge trigger type 00: first falling edge start, and all the subsequent falling edge trigger the pulse width to be captured. 01: first rising edge starts, and all the subsequent rising edge and falling edge trigger width to be captured. 10: first falling edge starts, and all the subsequent rising edge and falling edge trigger width to be captured. 11: first rising edge start, and all the subsequent rising edge trigger the pulse width to be captured. Note: details described in the subsequent diagram .
9~7	PSC	PSC	PWDT counter prescale 000 ~ 111: respectively represents 1/2/4/8/.../128.
5	IE	IE	PWDT module interrupt enable 0: disable 1: enable

Bit(s)	Mnemonic	Name	Description
4	PRDYIE	PRDYIE	PWDT pulse width data ready interrupt enable 0: disable 1: enable
3	OVIE	OVIE	PWDT counter overflow interrupt enable 0: disable 1: enable
2	SR	SR	PWDT soft reset <i>Note: read as 0</i> 0: no effect 1: enable
1	RDYF	RDYF	PWDT pulse width valid <i>Note: write 0 to clear</i> 0: PWDT pulse width register is not up to date. 1: PWDT pulse width register has been updated.
0	OVF	OVF	PWDT counter overflow <i>Note: write 0 to clear</i> 0: no overflow 1: runs from 0x 0000 to 0x FFFF.

00000004 PWDT_NPWCV PWDT NPWCV Count Value 00000000

Bit	31~16	15~0
Name	PWDTC	NPWCV
Type	R	R
Reset	0	0

Bit(s)	Mnemonic	Name	Description
31:16	PWDTC	PWDTC	Pulse Width Counter Used to count for pulse width measurement or for timer.
15:0	NPWCV	NPWCV	Negative Pulse Width Count Value Indicate the negative pulse width value.

00000008 PWDT_INIT1 PWDT Initialization Register 1 00000000

Bit	31	30	29	28	27~12	11	10	9	8	7~4	3~0
Name		HALA	HALB	HALLC	TIMCNTVAL	CM PEN	TIMEN	HALL EN	FILT EN	FILT_PSC	FILTVAL
Type		R	R	R	RW	RW	RW	RW	RW	RW	R
Reset		0	0	0	0	0	0	0	0	0	0

Note: To make the filter function work, PWDT_INIT1[FILTVAL] must be set more than 1, Otherwise filter function will not work.

Bit(s)	Mnemonic	Name	Description
30:28	HALLA/B/C	halla/b/c	HALLA/HALLB/HALLC status value If 3 hall sensors installed spacing 60 electrical degree: 100 → 110 → 111 → 011 → 001 → 000 else 3 hall sensors installed spacing 120 electrical degree: 101 → 100 → 110 → 010 → 011 → 001

Bit(s)	Mnemonic	Name	Description
27 ~ 12	TIMCNTVAL	timcntval	Timer counter load value Counter runs from 0x0000 to TIMCNTVAL.
11	CMPEN	cmpen	Switch the source of pwdt_in0 ~ pwdt_in2 <i>Note: When CMPEN=1, pwdt_in0 ~ pwdt_in2 are derived from acmp0_0 ~ acmp0_2 internally. And then acmp0_0 ~ acmp0_2 signals will be measured by the behavior of HALL sensor if HALLEN=1, otherwise just one of them will be measured based on the PINSEL.</i> 1: enable the pwdt_in0 ~ pwdt_in2 derived from acmp0_0 ~ acmp0_2 internally. 0: enable the pwdt_in0 ~ pwdt_in2 derived from pad PWDT_IN0 ~ PWDT_IN2 externally.
10	TIMEN	timen	Enable the timer function when PWDTEN = 0 0: disable. 1: enable timer function.
9	HALLEN	hallen	Enable hall sensor signal detect function when PWDTEN = 1 0: disable the hall sensor signal detect function. 1: enable the hall sensor signal detect function.
8	FILTEN	filten	Enable the PWDT input filter function when PWDTEN =1 0: disable. 1: enable the filter function.
7 ~ 4	FILT_PSC	filt_psc	Prescaler for filter 1~ 12: respectively represent 2/4/8.../4096 divisor. 0, 13 ~ 15: not divide the filter clock.
3 ~ 0	FILTVAL	filtval	Filter value 0 ~ 15 : for filter different width pulse.

14 TIMER

14.1 Introduction

The TIMER module is an array of timers that can be used to raise interrupts and triggers.

14.2 Features

- Ability of timers to generate trigger pulses.
- Ability of timers to generate interrupts.
- Maskable interrupts.
- Independent timeout periods for each timer.
- Maxim timer number is 8.
 - 2 * 32bit timer.
 - 6 * 16bit timer.
- Chain Mode.

14.3 Functional description

The following is the block diagram of the TIMER module.

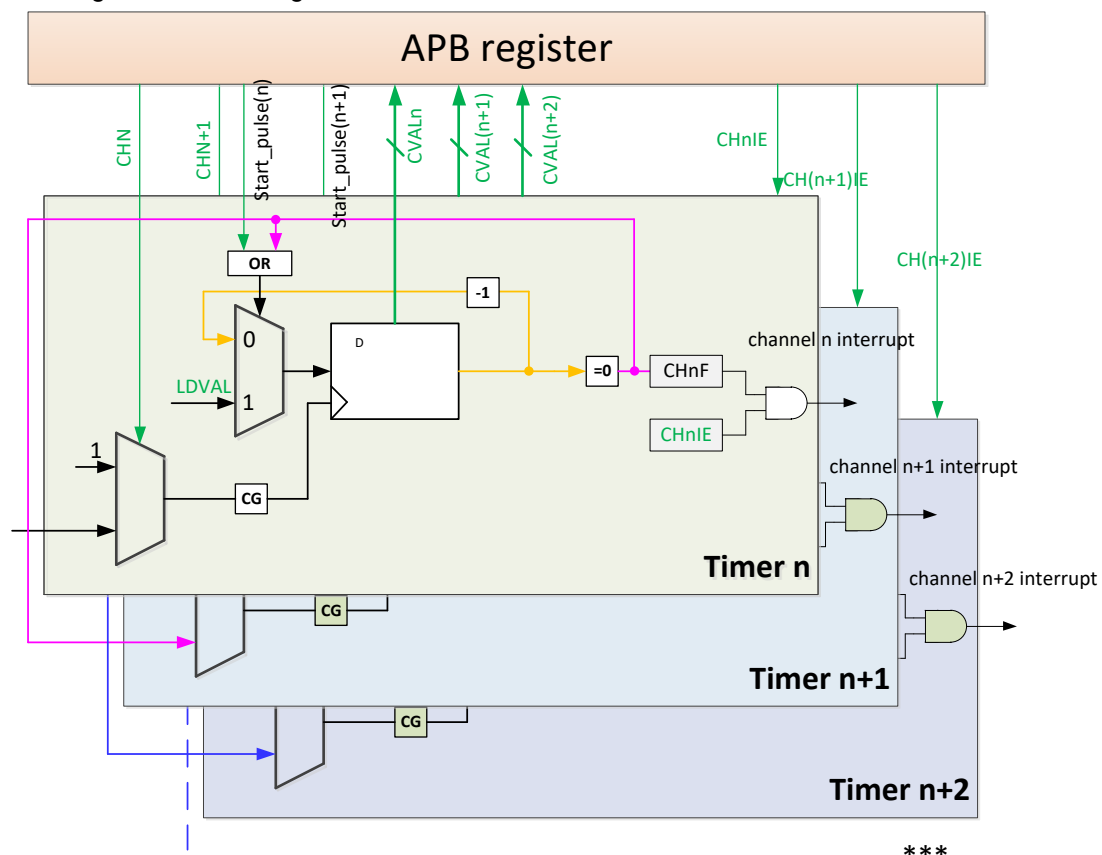


Figure 14-1 TIMER block diagram

14.3.1 General operation

The timers generate triggers at periodic intervals, when enabled. The timers load the start values as specified in their LDVAL registers, count down to 0 and then load the respective start value again. Each time a timer reaches 0, it will generate a trigger pulse and set the interrupt flag.

All interrupts can be enabled or masked by setting INIT[TIE]. A new interrupt can be generated only after the previous one is cleared. If desired, the current counter value of the timer can be read via the CVAL registers. The counter period can be restarted, by first disabling, and then enabling the timer with INIT[TIMEREN].

14.3.2 Chained timers

When a timer has chain mode enabled, it will only count after the previous timer has expired. So if timer n-1 has counted down to 0, counter n will decrement the value by one. This allows chaining some of the timers together to form a longer timer. The first timer (timer 0) cannot be chained to any other timer.

As TIMER block has 8 timers, Chains can more than one. For example, Time 0/1/2 in one chain, and Time 3/4/5/6 in another one, the rest of Timer in third one.

14.4 Memory map and register definition

Table 14-1 TIMER register map

TIMER: 0x40011000

ADDRESS	TITLE	DESCRIPTION
TIMER+ 0x000	MCR	TIMER MCR register
TIMER + 0x100	INITVAL	TIMER INITVAL register OFFSET ADDR = 0x100+16*x(x=0 to 7)
TIMER + 0x104	CVAL	TIMER CVAL register OFFSET ADDR = 0x104+16*x (x=0d to 7d)
TIMER + 0x108	INIT	TIMER INIT register OFFSET ADDR = 0x108+16*x (x=0 to 7)
TIMER + 0x10C	TF	TIMER TF register OFFSET ADDR = 0x10C+16*x(x=0 to 7)

MCR

TIMER MCR register

OFFSET ADDR = 0x00

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

Bit(s)	Name	Description
Module Disable - (TIMER section)		
1	MDIS	Disables the timer module. This field must be enabled before any other setup is done. 0: TIMER module is enabled. 1: TIMER module is disabled.

INITVAL **TIMER INITVAL register** **OFFSET ADDR = 0x100+16*x(x=0 to 7)**

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

Bit(s)	Name	Description
LDVAL: Load Value Register		
31:0	LDVAL	Sets the timer start value. The timer will count down until it reaches 0, then it will generate an interrupt and load this register value again. Writing a new value to this register will not restart the timer. Instead the value will be loaded after the timer expires. To abort the current cycle and start a timer period with the new value, the timer must be disabled and enabled again. Note: when $x > 1$, LDVAL bit width is 16.

CVAL **TIMER CVAL register** **OFFSET ADDR = 0x104+16*x (x=0d to 7d)**

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

Bit(s)	Name	Description
CVALx : Current Timer Value Register		
31:0	CVALx	Current Timer Value Represents the current timer value, if the timer is enabled. Note: when $x > 1$, CVALx bit width is 16.

INIT **TIMER INIT register** **OFFSET ADDR = 0x108+16*x (x=0 to 7)**

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W																
Default	0															
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R														LINKEN	TIE	TIMEREN
W																
Default	0													0	0	0

Bit(s)	Name	Description
2	LINKEN	LINKEN: Chain Mode When activated, Timer n-1 needs to expire before timer n can decrement by 1. Timer 0 cannot be chained. 0: Timer is not chained. 1: Timer is chained to previous timer. For example, for Timer 2, if this field is set, Timer 2 is chained to Timer 1.
1	TIE	TIE: Timer Interrupt Enable When an interrupt is pending, or, TFLGn[TIF] is set, enabling the interrupt will immediately cause an interrupt event. To avoid this, the associated TFLGn[TIF] must be cleared first. 0: Interrupt requests from Timer n are disabled. 1: Interrupt will be requested whenever TIF is set.
0	TIMEREN	TIMEREN: Timer Enable Enables or disables the timer. 0: Timer n is disabled. 1: Timer n is enabled.

TF **TIMER TF register** **OFFSET ADDR = 0x10C+16*x(x=0 to 7)**

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

Bit(s)	Name	Description
0	TIF	TIF: Timer Interrupt Flag Sets to 1 at the end of the timer period. Writing 1 to this flag clears it. Writing 0 has no effect. If enabled, or, when TCTRLx[TIE] = 1, TIF causes an interrupt request. 0: Timeout has not yet occurred.

Bit(s)	Name	Description
		1: Timeout has occurred.

14.5 Interrupts

All the timers support interrupt generation.

Timer interrupts can be enabled by setting INIT[TIE].TF[TIF] flag are set to 1 when a timeout occurs on the associated timer. and are cleared to 0 by writing a 1 to the corresponding TF[TIF].

15 CTU

15.1 Introduction

The CTU module defines module-to-module interconnections for this device.

15.2 Features

- UART TX modulation.
- UART RX capture.
- UART RX filter.
- RTC capture.
- ADC trigger.
- PWM2 synchronization.

15.3 Function description

The following is the block diagram of CTU.

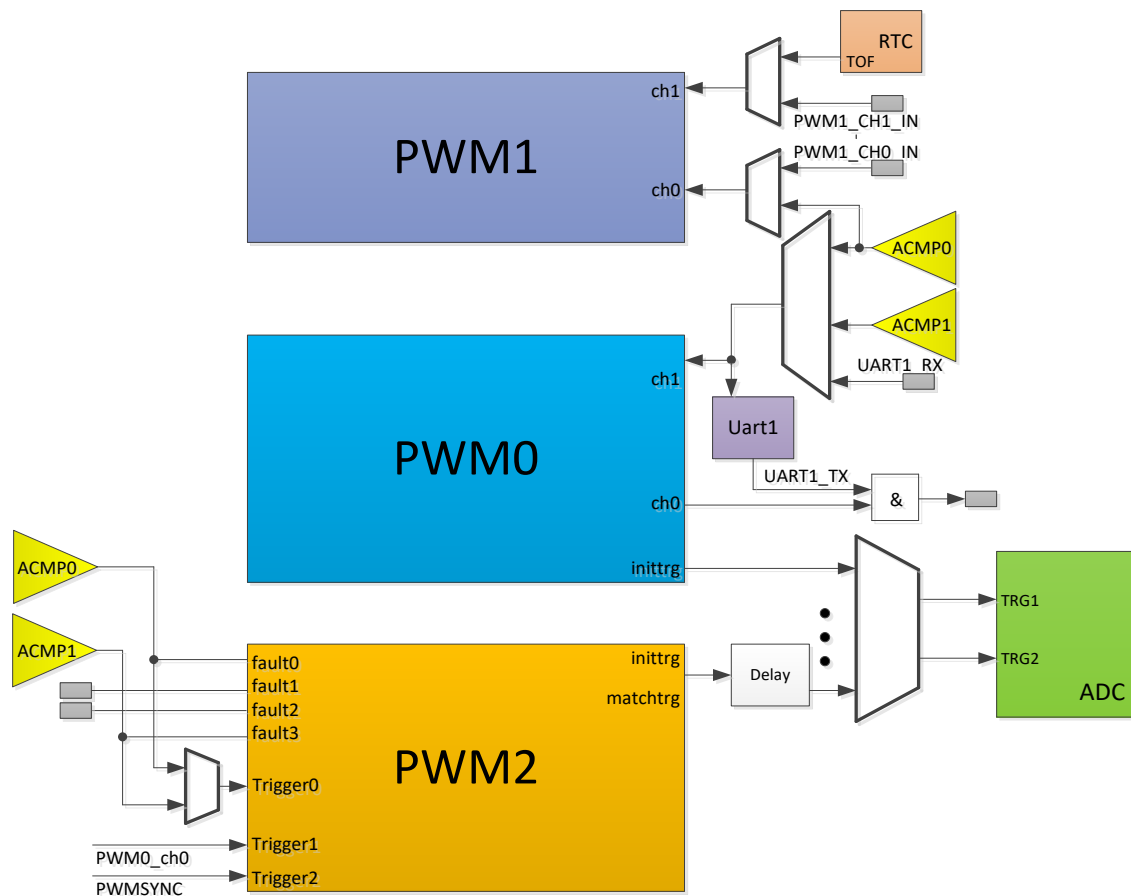


Figure 15-1 CTU block diagram

15.3.1 ACMP output capture

The CONFIG1[ACIC] bit enables the output of ACMP0 to be connected to PWM1_CH0, and the PWM1_CH0 pin is released to give other multiplexing functions.

Setting CONFIG1[RXDFE] to 01b selects the ACMP0 output to be connected to the receiver channel of UART1.

Setting CONFIG1[RXDFE] to 10b selects the ACMP1 output to be connected to the receiver channel of UART1.

The ACMP0 and ACMP1 outputs can be connected to the PWDT input (for BLDC use) or can be used as PWM2 trigger/fault input and ADC hardware trigger.

15.3.2 ADC hardware trigger

ADC module may initiate a conversion via a hardware trigger. The following table shows the available ADC hardware trigger1 sources by setting CTU_CFG1 [ADHWT1]. And be like to ADC hardware trigger1, ADC hardware trigger2 sources by setting CTU_CFG2 [ADHWT2].

Table 15-1 ADC hardware trigger1 source

ADHWT1	ADC hardware trigger1
000	RTC overflow
001	PWM0 init trigger
010	PWM2 init trigger with 8-bit programmable delay
011	PWM2 match trigger with 8-bit programmable delay
100	TIMER ch0 overflow
101	TIMER ch1 overflow
110	ACMP0 out
111	ACMP1 out

When ADC hardware trigger selects the output of PWM2 triggers, an 8-bit delay block will be enabled. This logic delays any trigger from PWM2 with an 8-bit counter whose value is specified by CTU_CFG1 [DELAY]. The reference clock to this module is the bus clock with selectable predivider specified by CTU_CFG1 [BUSREF].

15.4 Register definition

Table 15-2 CTU register map

CTU: 0x40016000

ADDRESS	TITLE	DESCRIPTION
CTU + 0x000	Config1	CTU Config1 register
CTU + 0x004	Config2	CTU Config2 register

CTU Config1 register

OFFSET ADDR = 12'h00

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DELAY								DLYACT	ADHWT1				BUSREF		
W																
RESET	0								0	0				0		
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TX	PW	0	RXDCE	ACIC	RTC	RXDFE	0	0	0	ACTRG	0	0	0	0	0
W	D	MS														
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit(s)	Name	Description
31:24	DELAY	Trigger Delay This write-once bit enables window mode. Specifies the delay from PWM2 initial or match trigger to ADC hardware trigger when 1 is written to ADHWT. The 8-bit modulo value allows the delay from 0 to 255 upon the BUSREF clock settings. This is a one-shot counter that starts ticking when the trigger arrives and stops ticking when the counter value reaches the modulo value that is defined.
23	DLYACT	Trigger Delay Active This read-only field specifies the status if the PWM2 initial or match delay is active. This field is set when an PWM2 trigger arrives and the delay counter is ticking. Otherwise, this field will be clear. 0: The delay is inactive. 1: The delay is active.
22:20	ADHWT1	ADC Regular Group Hardware Trigger Source1 Selects the ADC hardware trigger source. All trigger sources start ADC conversion on rising-edge. 000: RTC overflow as the ADC hardware trigger 001: PWM0 as the ADC hardware trigger 010: PWM2 init trigger with 8-bit programmable counter delay 011: PWM2 match trigger with 8-bit programmable counter delay 100: TIMER channel0 overflow as the ADC hardware trigger 101: TIMER channel1 overflow as the ADC hardware trigger 110: ACMP0 out as the ADC hardware trigger. 111: ACMP1 out as the ADC hardware trigger
18:16	CLK	BUS Clock Output select Enables bus clock output via an optional prescaler. 000: Bus 001: Bus divided by 2 010: Bus divided by 4 011: Bus divided by 8 100: Bus divided by 16 101: Bus divided by 32

Bit(s)	Name	Description
		110: Bus divided by 64 111: Bus divided by 128
15	TXDME	UART1_TX Modulation Select Enables the UART1_TX output modulated by PWM0 channel 0. 0: UART1_TX output is connected to pinout directly. 1: UART1_TX output is modulated by PWM0 channel 0 before mapped to pinout.
14	PWMSYNC	PWM Synchronization Select Generates a PWM synchronization trigger to the PWM module if 1 is written to this field. Note that when set PWMSYNC = 1 to generate a trigger behavior, the PWMSYNC bit should be write to 0 manually. 0: No synchronization triggered. 1: Generates a PWM synchronization trigger to the PWM modules.
12	RXDCE	UART1_RX Capture Select Enables the UART1_RX to be captured by PWM0 channel 1. 0: UART1_RX input signal is connected to the UART0 module only. 1: UART1_RX input signal is connected to the UART0 module and PWM0 channel 1.
11	ACIC	Analog Comparator to Input Capture Enable Connects the output of ACMP0 to PWM1 input channel 0. 0: ACMP0 output is not connected to PWM1 input channel 0. 1: ACMP0 output is connected to PWM1 input channel 0.
10	RTCC	Real-Time Counter Capture Allows the Real-time Counter (RTC) overflow to be captured by PWM1 channel 1. 0: RTC overflow is not connected to PWM1 input channel 1. 1: RTC overflow is connected to PWM1 input channel 1.
9:8	RXDFE	UART1 RxD Filter Select Enables the UART1 RxD input to be filtered by ACMP. When this function is enabled, any signal tagged with ACMP inputs can be regarded UART1. 00: RXD input signal is connected to UART1 module directly. 01: RXD input signal is filtered by ACMP0, then injected to UART1. 10: RXD input signal is filtered by ACMP1, then injected to UART1. 11: Reserved.
5	ACTRG	ACMP Trigger PWM2 selection Selects the two ACMP outputs as the trigger0 input of PWM2. 0: ACMP0 out 1: ACMP1 out

CTU Config2 register
OFFSET ADDR = 12'h04

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
RESET	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0							ADHWT2			UARTPWDTS		ACPWDTS			
W																
RESET	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Bit(s)	Name	Description
8:6	ADHWT2	ADC Injection Group Hardware Trigger Source2
		Selects the ADC hardware trigger source. All trigger sources start ADC conversion on rising-edge.
		000: RTC overflow as the ADC hardware trigger
		001: PWM0 as the ADC hardware trigger
		010: PWM2 init trigger with 8-bit programmable counter delay
		011: PWM2 match trigger with 8-bit programmable counter delay
		100: TIMER channel0 overflow as the ADC hardware trigger
		101: TIMER channel1 overflow as the ADC hardware trigger
5:4	UARTPWDTS	110: ACMP0 out as the ADC hardware trigger.
		111: ACMP1 out as the ADC hardware trigger
		PWDT UART RX select
		This field selects PWDTIN3 input signal
		00: UART1 RX is connected to PWDTIN3(Internal PWDT Channel).
3	ACPWDTS	01: UART2 RX is connected to PWDTIN3(Internal PWDT Channel).
		10: UART3 RX is connected to PWDTIN3(Internal PWDT Channel).
		11: Reserved.
		PWDT ACMP_OUT select
		Note that if this function wants to be used, Please Set UARTPWDTS = 11 firstly
		This field selects PWDTIN3 input signal.
		0: ACMP1_OUT is connected to PWDTIN3(Internal PWDT Channel).
		1: ACMP0_OUT is connected to PWDTIN3(Internal PWDT Channel).

16 CRC

16.1 Introduction

The cyclic redundancy check (CRC) module generates 16-/32-bit CRC code for error detection. The CRC module provides a programmable polynomial, WAS, and other parameters required to implement a 16-bit or 32-bit CRC standard. The 16-/32-bit code is calculated for 32 bits of data at a time.

16.2 Features

- Hardware CRC generator circuit using a 16-bit or 32-bit programmable shift register.
- Programmable initial seed value and polynomial.
- Option to transpose input data or output data (the CRC result) bitwise or byte-wise.
- This option is required for certain CRC standards. A byte-wise transpose operation is not possible when accessing the CRC data register via 8-bit accesses. In this case, the user's software must perform the byte-wise transpose function.
- Option for inversion of final CRC result.
- 32-bit CPU register programming interface.

16.3 Block diagram

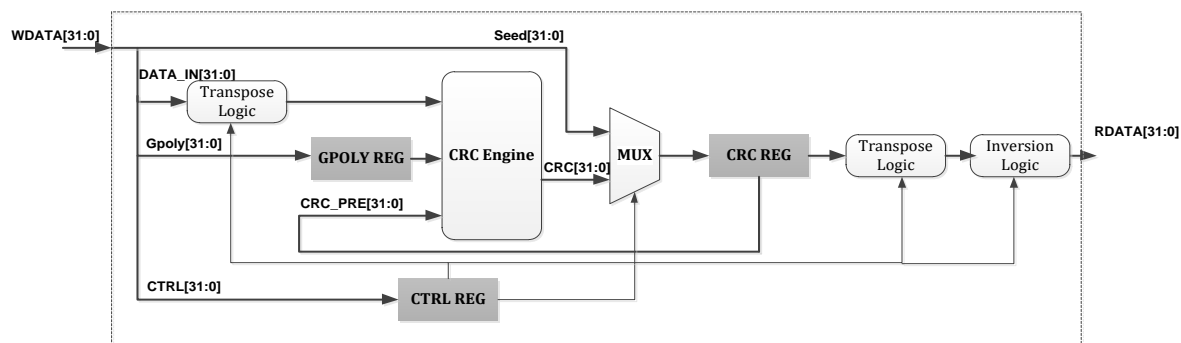


Figure 16-1 CRC block diagram

16.4 Memory map and register descriptions

Table 16-1 CRC register map

CRC: 0x20081000

ADDRESS	TITLE	DESCRIPTION
CRC + 0x000	CRC_DATA	CRC Data register
CRC + 0x004	CRC_DATA	CRC Data register
CRC + 0x008	CRC_CTRL	CRC control register

0x000	CRC DATA				CRC Data register								00			
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	Byte3								Byte2							
Type	WR								WR							
Reset	0x00								0x00							
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Byte1								Byte0							
Type	WR								WR							
Reset	0x00								0x00							

Bit(s)	Mnemonic	Name	Description
[31:24]	Byte3	Byte3	CRC DATA Byte3 In 16-bit CRC mode (CTRL[TCRC] is 0), this field is not used for programming a seed value. In 32-bit CRC mode (CTRL[TCRC] is 1), values written to this field are part of the seed value when CTRL[WAS] is 1. When CTRL[WAS] is 0, data written to this field is used for CRC checksum generation in both 16-bit and 32-bit CRC modes.
[23:16]	Byte2	Byte2	CRC DATA Byte2 In 16-bit CRC mode (CTRL[TCRC] is 0), this field is not used for programming a seed value. In 32-bit CRC mode (CTRL[TCRC] is 1), values written to this field are part of the seed value when CTRL[WAS] is 1. When CTRL[WAS] is 0, data written to this field is used for CRC checksum generation in both 16-bit and 32-bit CRC modes.
[15:8]	Byte1	Byte1	CRC DATA Byte1 When CTRL[WAS] is 1, values written to this field are part of the seed value. When CTRL[WAS] is 0, data written to this field is used for CRC checksum generation.
[7:0]	Byte0	Byte0	CRC DATA Byte0 When CTRL[WAS] is 1, values written to this field are part of the seed value. When CTRL[WAS] is 0, data written to this field is used for CRC checksum generation.

0x004	CRC DATA				CRC Data register								00			
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	High															
Type	WR															
Reset	0x00															
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	Low															
Type	WR															
Reset	0x00															

Bit(s)	Mnemonic	Name	Description
31:16	High	High Half word	High Polynomial Half-word Writable and readable in 32-bit CRC mode (CTRL[TCRC] is 1). This field is not writable in 16-bit CRC mode (CTRL[TCRC] is 0).

Bit(s)	Mnemonic	Name	Description
15:0	Low	Low Half word	Low Polynomial Half-word Writable and readable in both 32-bit and 16-bit CRC modes

0x008	CRC_CTRL				CRC control register												00
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Name																	
Type																	
Reset																	
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Name									TOTW		TOTR			FXOR	WAS	TCRC	
Type									WR		WR			WR	WR	WR	
Reset									0		0			0	0	0	

Bit(s)	Mnemonic	Name	Description
7:6	TOTW	Type Of Transpose For Write	Type Of Transpose For Writes Defines the transpose configuration of the data written to the CRC data register. See the description of the transpose feature for the available transpose options. 00: No transposition. 01: Bits in bytes are transposed; bytes are not transposed. 10: Both bits in bytes and bytes are transposed. 11: Only bytes are transposed; no bits in a byte are transposed.
5:4	TOTR	Type Of Transpose For Read	Type Of Transpose For Read Identifies the transpose configuration of the value read from the CRC Data register. See the description of the transpose feature for the available transpose options. 00: No transposition. 01: Bits in bytes are transposed; bytes are not transposed. 10: Both bits in bytes and bytes are transposed. 11: Only bytes are transposed; no bits in a byte are transposed.
3	RSV	Reserved	
2	FXOR	Complement Read Of CRC Data Register	Some CRC protocols require the final checksum to be XORed with 0xFFFFFFFF or 0xFFFF. Asserting this bit enables on the fly complementing of read data. 0: No XOR on reading. 1: Invert or complement the read value of the CRC Data register.
1	WAS	Write as seed	Write CRC Data Register As Seed When asserted, a value written to the CRC data register is considered a seed value. When deasserted, a value written to the CRC data register is taken as data for CRC computation. 0: Writes to the CRC data register are data values. 1: Writes to the CRC data register are seed values.
0	TCRC	Type of CRC	0: 16-bit CRC protocol.

Bit(s)	Mnemonic	Name	Description
1: 32-bit CRC protocol.			

16.5 Functional description

16.5.1 CRC initialization/reinitialization

To enable the CRC calculation, the user must program CRC_CTRL[WAS], CRC_GPOLY, necessary parameters for transposition and CRC result inversion in the applicable registers. Asserting CRC_CTRL[WAS] enables the programming of the seed value into the CRC_DATA register.

After a completed CRC calculation, reasserting CRC_CTRL[WAS] and programming a seed, whether the value is new or a previously used seed value, reinitialize the CRC module for a new CRC computation. All other parameters must be set before programming the seed value and subsequent data values.

16.5.2 CRC calculations

In 16-bit and 32-bit CRC modes, data values can be programmed 8 bits, 16 bits, or 32 bits at a time, provided all bytes are contiguous. Noncontiguous bytes can lead to an incorrect CRC computation.

16.5.3 Transpose feature

By default, the transpose feature is not enabled. However, some CRC standards require the input data and/or the final checksum to be transposed. The user software has the option to configure each transpose operation separately, as desired by the CRC standard. The data is transposed on the fly while being read or written.

Some protocols use little endian format for the data stream to calculate a CRC. In this case, the transpose feature usefully flips the bits. This transpose option is one of the types supported by the CRC module.

16.5.4 Types of transpose

The CRC module provides several types of transpose functions to flip the bits and/or bytes, for both writing input data and reading the CRC result, separately using the CTRL[TOTW] or CTRL[TOTR] fields, according to the CRC calculation being used.

The following types of transpose functions are available for writing to and reading from the CRC data register:

1. CTRL[TOTW] or CTRL[TOTR] is 00.

No transposition occurs.

2. CTRL[TOTW] or CTRL[TOTR] is 01.

Bits in a byte are transposed, while bytes are not transposed. reg[31:0] becomes {reg[24:31], reg[16:23], reg[8:15], reg[0:7]}.

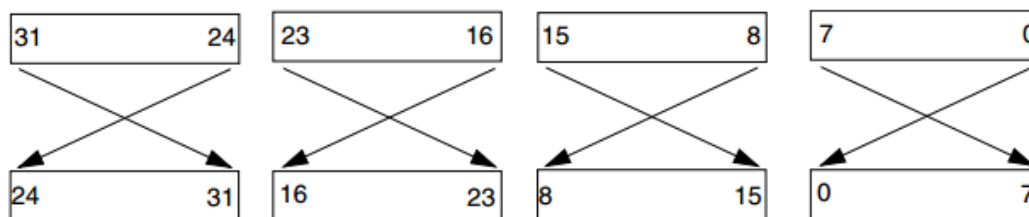


Figure 16-2 CTRL[TOTW] / CTRL[TOTR] is 01

3. CTRL[TOTW] or CTRL[TOTR] is 10.

Both bits in bytes and bytes are transposed.

reg[31:0] becomes = {reg[0:7], reg[8:15], reg[16:23], reg[24:31]}.

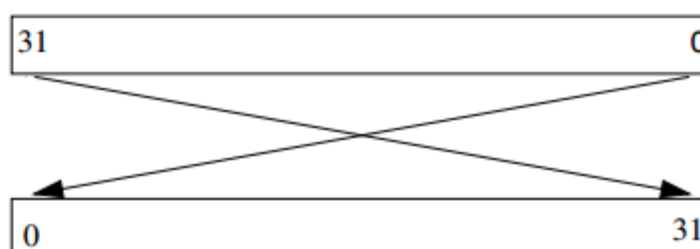


Figure 16-3 CTRL[TOTW] / CTRL[TOTR] is 10

4. CTRL[TOTW] or CTRL[TOTR] is 11.

Bytes are transposed, but bits are not transposed.

reg[31:0] becomes {reg[7:0], reg[15:8], reg[23:16], reg[31:24]}.

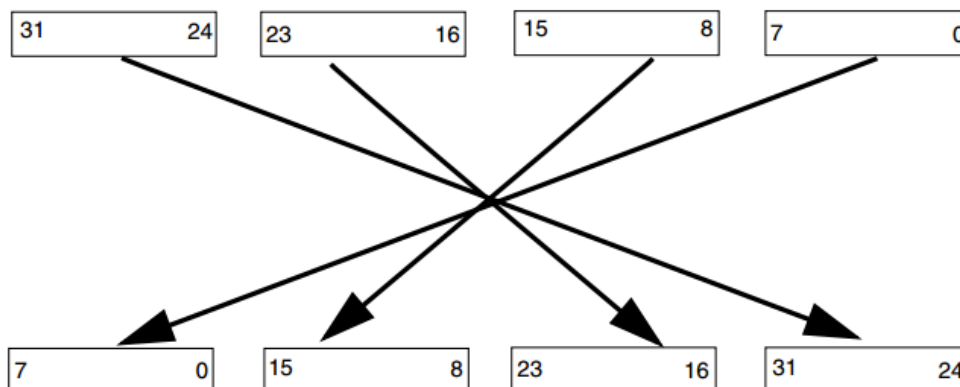


Figure 16-4 CTRL[TOTW] / CTRL[TOTR] is 11

NOTE:

For 8-bit and 16-bit write accesses to the CRC data register, the data is transposed with zeros on the unused byte or bytes (taking 32 bits as a whole), but the CRC is calculated on the valid byte(s) only. When reading the CRC data register for a 16-bit CRC result and using transpose options 10 and 11, the resulting value after transposition resides in the CRC[*HU*: *HL*] fields. The user software must account for this situation when reading the 16-bit CRC result, so reading 32 bits is preferred.

16.5.5 CRC result complement

When CTRL[FXOR] is set, the checksum is complemented. The CRC result complement function outputs the complement of the checksum value stored in the CRC data register every time the CRC data register is read. When CTRL[FXOR] is cleared, reading the CRC data register accesses the raw checksum value.

16.5.6 CRC data register (CRC_DATA)

The CRC Data register contains the value of the seed, data, and checksum. When CTRL[WAS] is set, any write to the data register is regarded as the seed value. When CTRL[WAS] is cleared, any write to the data register is regarded as data for general CRC computation.

In 16-bit CRC mode, the HU and HL fields are not used for programming the seed value, and reads of these fields return an indeterminate value. In 32-bit CRC mode, all fields are used for programming the seed value.

When programming data values, the values can be written 8 bits, 16 bits, or 32 bits at a time, provided all bytes are contiguous; with MSB of data value written first.

After all data values are written, the CRC result can be read from this data register. In 16-bit CRC mode, the CRC result is available in the LU and LL fields. In 32-bit CRC mode, all fields contain the result. Reads of this register at any time return the intermediate CRC value, provided the CRC module is configured.

16.5.7 CRC Polynomial register (CRC_GPOLY)

This register contains the value of the polynomial for the CRC calculation. The HIGH field contains the upper 16 bits of the CRC polynomial, which are used only in 32-bit CRC mode. Writes to the HIGH field are ignored in 16-bit CRC mode. The LOW field contains the lower 16 bits of the CRC polynomial, which are used in both 16- and 32-bit CRC modes.

16.5.8 CRC Control register (CRC_CTRL)

This register controls the configuration and working of the CRC module. Appropriate bits must be set before starting a new CRC calculation. A new CRC calculation is initialized by asserting CTRL[WAS] and then writing the seed into the CRC data register.

16.6 Program guide

16.6.1 16-bit CRC

To compute a 16-bit CRC:

1. Clear CRC_CTRL[TCRC] to enable 16-bit CRC mode.
2. Program the transpose and complement options in the CTRL register as required for the CRC calculation. See [16.5.3 Transpose feature](#) and [16.5.5 CRC result complement](#) for details.
3. Write a 16-bit polynomial to the CRC_GPOLY[LOW] field. The CRC_GPOLY[HIGH] field is not usable in 16-bit CRC mode.
4. Set CRC_CTRL[WAS] to program the seed value.
5. Write a 16-bit seed to CRC_DATA[15:0]. CRC_DATA[31:16] are not used.
6. Clear CRC_CTRL[WAS] to start writing data values.

7. Write data values into CRC_DATA[31:0]. A CRC is computed on every data value write, and the intermediate CRC result is stored back into CRC_DATA[15:0].
8. When all values have been written, read the final CRC result from CRC_DATA[15:0].

16.6.2 32-bit CRC

To compute a 32-bit CRC:

1. Set CRC_CTRL[TCRC] to enable 32-bit CRC mode.
2. Program the transpose and complement options in the CTRL register as required for the CRC calculation. See [16.5.3 Transpose feature](#) and [16.5.5 CRC result complement](#) for details.
3. Write a 32-bit polynomial to CRC_GPOLY[HIGH:LOW].
4. Set CRC_CTRL[WAS] to program the seed value.
5. Write a 32-bit seed to CRC_DATA[31:0].
6. Clear CRC_CTRL[WAS] to start writing data values.
7. Write data values into CRC_DATA[31:0]. A CRC is computed on every data value write, and the intermediate CRC result is stored back into CRC_DATA[31:0].
8. When all values have been written, read the final CRC result from CRC_DATA[31:0]. The CRC is calculated bitwise, and two clocks are required to complete one CRC calculation.

17 GPIO

17.1 Introduction

The general-purpose input and output (GPIO) module is accessible via APB BUS and also communicates to the processor core via AHB BUS for maximum performance. The GPIO registers support APB 32-bit accesses, and AHB byte accesses.

The GPIO data direction and output data registers control the direction and output data of each pin when the pin is configured for the GPIO function. The GPIO input data register displays the logic value on each pin when the pin is configured for any digital function, provided the corresponding Port Control and Interrupt module for that pin is enabled. Efficient bit manipulation of the general-purpose outputs is supported through the addition of set, clear, and toggle write-only registers for each port output data register.

Also, The MCU I/O pins are connected to onboard peripherals/modules through a multiplexer that allows only one peripheral's alternate function (AF) connected to an I/O pin at a time. In this way, there can be no conflict between peripherals sharing the same I/O pin. Each I/O pin has a multiplexer with alternate function that can be configured through the GPIOx_Pmux registers.

17.2 Features

GPIO pins support the following modes:

- Up to 68 I/Os under control.
- Output states: push-pull or open drain (be related to I2C).
- Output data from output data register (GPIOx_ODR) or peripheral (alternate function output).
- Driving capability selection for each I/O.
- Input states: floating, pull-up/down, analog (be related to ADC PAD).
- Input data to input data register (GPIOx_IDR) or peripheral (alternate function input).
- Bit set and reset register (GPIOx_BSRR) for bitwise write access to GPIOx_ODR.
- Fast toggle capable of changing every two clock cycles.
- Highly flexible pin multiplexing allows the use of I/O pins as GPIOs or as one of several peripheral functions.
- Configurable rising or falling edge interrupt.
- Low power mode wakes up interrupt.

17.3 Block diagram

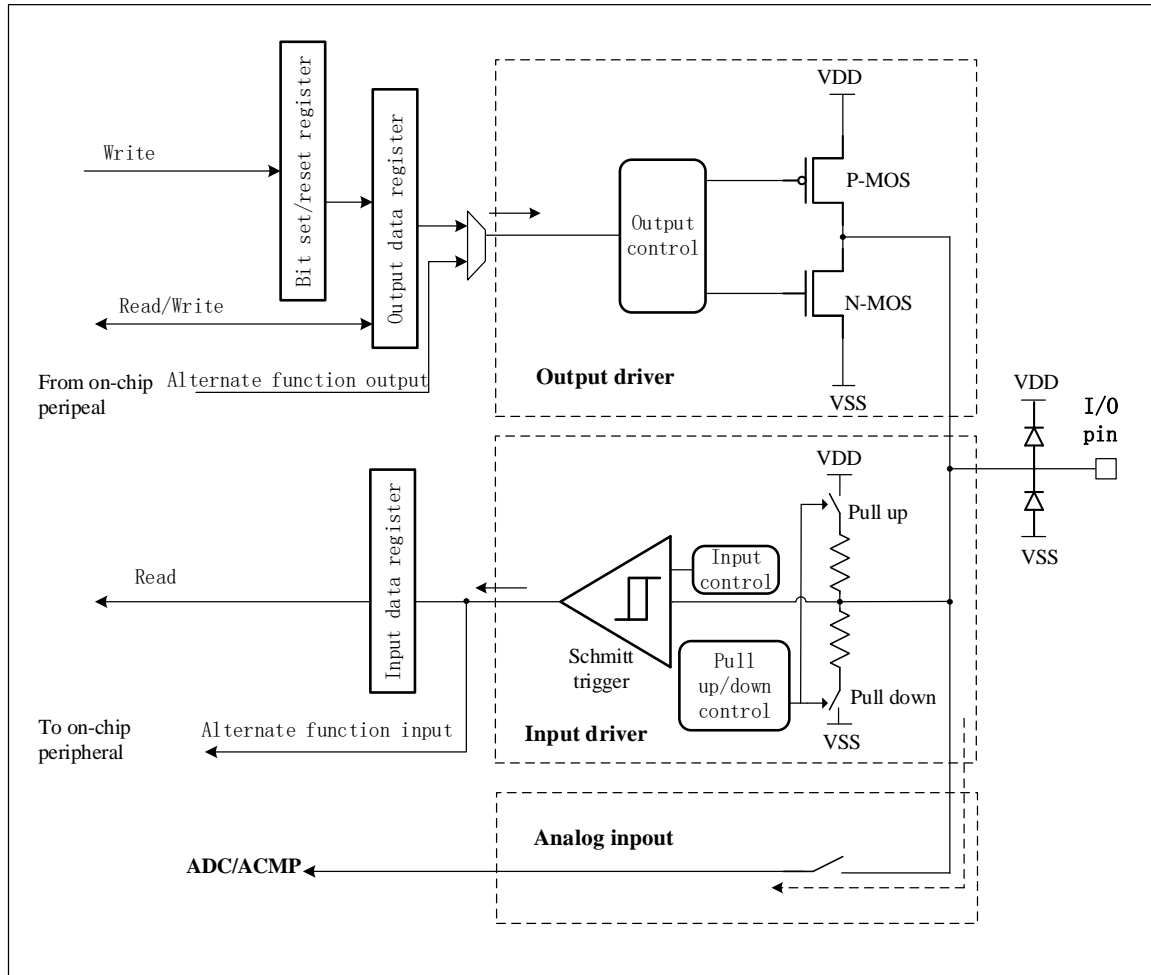


Figure 17-1 GPIO block diagram

17.4 Mode of operation

17.4.1 External interrupt

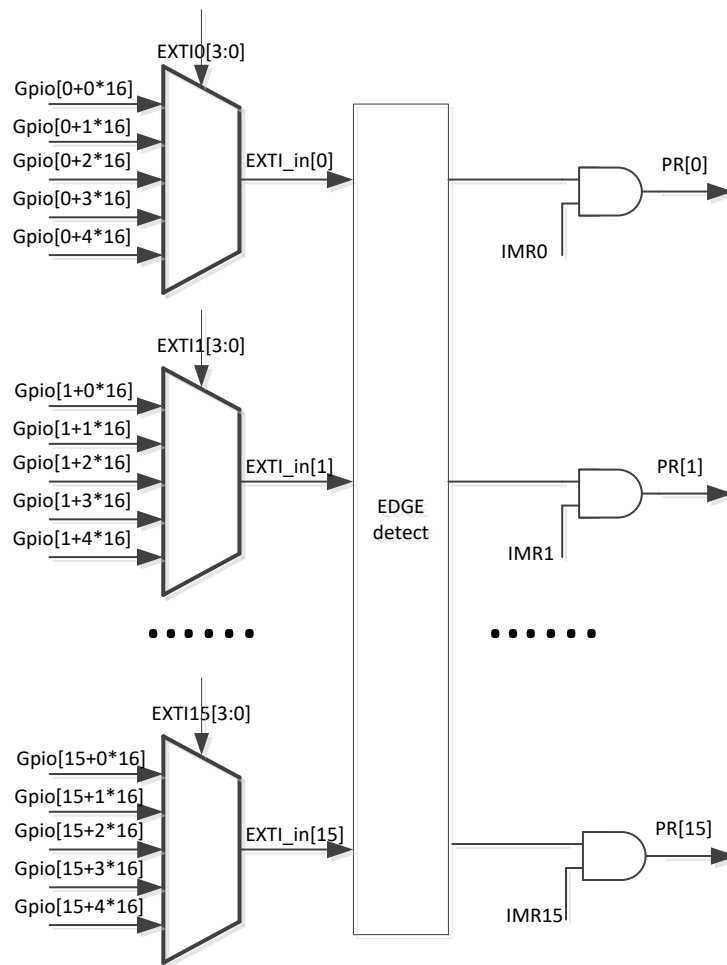


Figure 17-2 GPIO external interrupt

17.4.2 Multi-Function

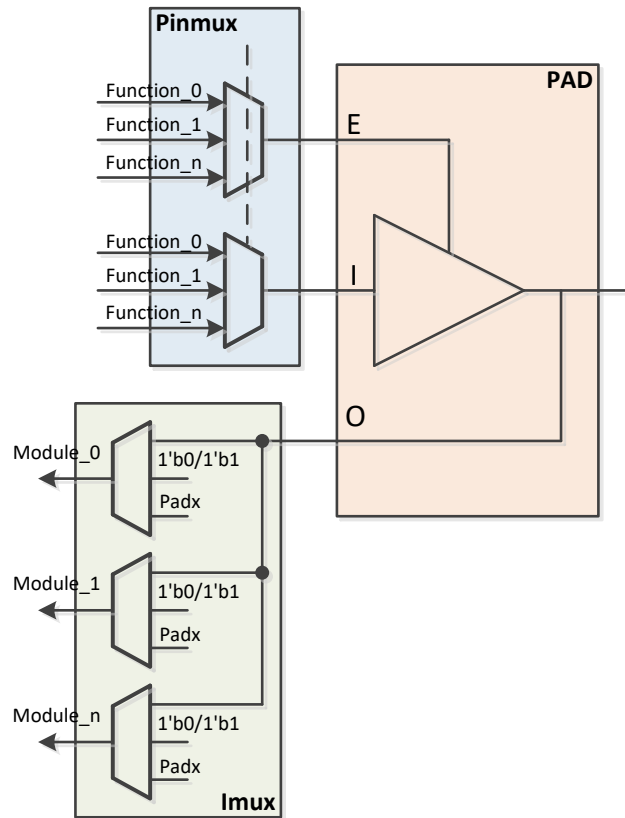


Figure 17-3 GPIO multi-function

Each IO have multi-functions, if we want to begin a peripheral communication, we should configure GPIO multi-function first. The corresponding multi-function of each GPIO is as below.

Table 17-1 GPIO multi-function

80 LQFP	64 LQFP	PIN Name	function 0 (after reset)	function 1	function 2	function 3	PINMUX	GPIO (NUM)
1	1	PA0	PA0	SPI1_NSS	TRACED3	FLASH_NSS	PMUX0[2:0]	0
2	2	PA1	PA1	SPI1_SCK	TRACECLK	FLASH_SCK	PMUX0[5:3]	1
3	3	PA2	PA2	SPI1_MISO	TRACED0	FLASH_DQ1	PMUX0[8:6]	2
4	4	PA3	PA3	SPI1_MOSI	TRACED1	FLASH_DQ0	PMUX0[11:9]	3
5	5	PA4	PA4	UART4_TX	CAN1_RX	FLASH_DQ3	PMUX0[14:12]	4
6	6	PA5	PA5	UART4_RX	CAN1_TX	FLASH_DQ2	PMUX0[17:15]	5
7		PD3	PD3	PWM1_CH0	HWLIN_TX		PMUX5[5:3]	51
8		PD4	PD4	PWM1_CH1	HWLIN_RX		PMUX5[8:6]	52
9		PD5	PD5	FTM_EXT	CAN1_STDBY		PMUX5[11:9]	53
10	7	OSC_OUT	OSC_OUT					
11	8	OSC_IN	OSC_IN					

80 LQF P	64 LQF P	PIN Name	function 0 (after reset)	function 1	function 2	function 3	PINMUX	GPIO (NUM)
12	9	AVSS50	AVSS50					
13	10	AVDD50	AVDD50					
14	11	AVDD50_I O	AVDD50_IO					
15	12	AVSS50_I O	AVSS50_IO					
16	13	PA6	PA6	ADC_IN10	UART1_CTS	UART4_TX	PMUX0[20:18]	6
17		PD6	PD6	ADC_IN12	UART1_RTS	UART4_RX	PMUX5[14:12]	54
18	14	PA7	PA7	ADC_IN0 (ACMP_IN0)	UART1_TX	UART5_TX	PMUX0[23:21]	7
19	15	PA8	PA8	ADC_IN1 (ACMP_IN1)	UART1_RX	UART5_RX	PMUX0[26:24]	8
20	16	PA9	PA9	ADC_IN2 (ACMP_IN2)	SPI2_NSS	HWLIN_TX	PMUX0[29:27]	9
21	17	PA10	PA10	ADC_IN3 (ACMP_IN3)	SPI2_SCK	HWLIN_RX	PMUX1[2:0]	10
22	18	PA11	PA11	ADC_IN4 (ACMP_IN4)	SPI2_MISO	UART3_RX	PMUX1[5:3]	11
23	19	PA12	PA12	ADC_IN5	SPI2_MOSI	UART3_TX	PMUX1[8:6]	12
24	20	PA13	PA13	ADC_IN6	I2C1_SCL	UART6_RX	PMUX1[11:9]	13
25	21	PA14	PA14	ADC_IN7	I2C1_SDA	UART6_TX	PMUX1[14:12]	14
26	22	PA15	PA15	ADC_IN8	UART2_RTS		PMUX1[17:15]	15
27	23	PB0	PB0	ADC_IN9	UART2_TX	CAN2_STDBY	PMUX1[20:18]	16
28	24	PB1	PB1	ADC_IN11	UART2_RX		PMUX1[23:21]	17
29		PD7	PD7	ADC_IN13	PWDT_EXT	PWM3_CH0	PMUX5[17:15]	55
30		PD8	PD8	ADC_IN14	CAN2_TX	PWM3_CH1	PMUX5[20:18]	56
31		PD9	PD9	ADC_IN15	CAN2_RX	PWDT_IN0	PMUX5[23:21]	57
32	25	PB2	PB2	NMI_B	UART3_TX	CAN1_STDBY	PMUX1[26:24]	18
33	26	NRST	NRST					
34	27	PB3	PB3	PWDT_IN1	UART3_RX		PMUX1[29:27]	19
35	28	PB4	PB4	PWDT_IN2	TRACED2		PMUX2[2:0]	20
36	29	PB5	PB5	UART1_TX	PWDT_IN1		PMUX2[5:3]	21
37	30	PB6	PB6	UART1_RX	PWDT_IN2	PWM3_CH0	PMUX2[8:6]	22
38	31	PB7	PB7	UART1_RTS	PWM_FAULT 1		PMUX2[11:9]	23
39		PD10	PD10	PWM_FAULT 1	I2C2_SCL		PMUX5[26:24]	58
40	32	PB8	PB8	PWDT_IN0	I2C2_SDA		PMUX2[14:12]	24
41	33	PB9	PB9	PWM0_CH0	PWM2_CH0	CAN1_RX	PMUX2[17:15]	25
42	34	DVSS50_ 1	DVSS50_1					
43	35	DVDD50_ 1	DVDD50_1					
44	36	VPP	VPP					
45	37	PB10	PB10	PWM0_CH1	PWM2_CH1	CAN1_TX	PMUX2[20:18]	26
46	38	PB11	PB11	SPI2_NSS	PWM2_CH2		PMUX2[23:21]	27

80 LQF P	64 LQF P	PIN Name	function 0 (after reset)	function 1	function 2	function 3	PINMUX	GPIO (NUM)
47	39	PB12	PB12	SPI2_SCK	PWM2_CH3		PMUX2[26:24]	28
48	40	PB13	PB13	SPI2_MISO	PWM2_CH4	UART5_TX	PMUX2[29:27]	29
49	41	PB14	PB14	SPI2_MOSI	PWM2_CH5	UART5_RX	PMUX3[2:0]	30
50		PD11	PD11	UART5_TX	UART3_RTS		PMUX5[29:27]	59
51		PD12	PD12	UART5_RX	UART4_RTS		PMUX6[2:0]	60
52		PD13	PD13	PWM2_CH4	UART5_RTS		PMUX6[5:3]	61
53		PD14	PD14	PWM2_CH5	UART6_RTS		PMUX6[8:6]	62
54	42	PB15	PB15	PWM2_CH0	PWM0_EXT		PMUX3[5:3]	31
55	43	PC0	PC0	PWM2_CH1	PWM1_EXT		PMUX3[8:6]	32
56	44	PC1	PC1	PWM2_CH2	UART5_RTS		PMUX3[11:9]	33
57	45	PC2	PC2	PWM2_CH3	UART6_RTS		PMUX3[14:12]	34
58	46	PC3	PC3	UARTTX_SFLASH	PWM2_CH4	PWM2_CH0	PMUX3[17:15]	35
59	47	PC4	PC4	UARTTX_SFLASH	PWM2_CH5	PWM2_CH1	PMUX3[20:18]	36
60	48	PC5	PC5	I2C2_SCL	UART1_CTS		PMUX3[23:21]	37
61	49	PC6	PC6	I2C2_SDA	UART1_CTS		PMUX3[26:24]	38
62	50	PC7	PC7	JTCK_SWCLK	UART3_RTS		PMUX3[29:27]	39
63	51	PC8	PC8	JTDO_TRACESWO	UART2_RTS		PMUX4[2:0]	40
64	52	PC9	PC9	JTMS_SWDIO	UART4_RTS	PWM_FAULT1	PMUX4[5:3]	41
65	53	PC10	PC10	CAN2_TX	UART6_TX	PWM_FAULT2	PMUX4[8:6]	42
66	54	PC11	PC11	CAN2_RX	UART6_RX	PWDT_IN0	PMUX4[11:9]	43
67		PD15	PD15	JTDI	CAN1_STDBY	PWDT_IN1	PMUX6[11:9]	63
68	55	BOOT	BOOT					
69	56	PC12	PC12	I2C1_SCL	UART5_TX		PMUX4[14:12]	44
70	57	PC13	PC13	I2C1_SDA	UART5_RX		PMUX4[17:15]	45
71	58	DVSS50_2	DVSS50_2					
72	59	DVDD50_2	DVDD50_2					
73		PE0	PE0	NJTRST	CAN2_STDBY	PWDT_IN2	PMUX6[14:12]	64
74	60	PC14	PC14	CAN1_RX	UART4_RX	PWDT_IN3	PMUX4[20:18]	46
75	61	PC15	PC15	CAN1_TX	UART4_TX		PMUX4[23:21]	47
76		PE1	PE1	HWLIN_TX	PWM1_CH0		PMUX6[17:15]	65
77		PE2	PE2	HWLIN_RX	PWM1_CH1		PMUX6[20:18]	66
78	62	PD0	PD0	UART1_CTS	PWM_FAULT2		PMUX4[26:24]	48
79	63	PD1	PD1	UART2_TX	PWM0_CH0	I2C2_SCL	PMUX4[29:27]	49
80	64	PD2	PD2	UART2_RX	PWM0_CH1	I2C2_SDA	PMUX5[2:0]	50

Note:

- 1、 All the pins are default as gpio on the first time power on except some dedicated pins.
- 2、 GPIO Control NUM is the only way to change gpio status.like set input/output,pull up/down.it's differ from PIN NUM(Importance).
- 3、 Eg: if we want to config PIN1 as SPI1_NSS, we should set PMUX0[2:0]=1.
- 4、 VPP pin is used for test and it must be floating.

17.5 Application note

17.5.1 External interrupt

The external interrupt/event controller consists of up to 16 edge detectors for generating event/interrupt requests. Each input line can be independently configured to select the type (event or interrupt) and the corresponding trigger event (rising edge, falling edge or both). Each line can also be masked independently. A pending register maintains the status line of the interrupt requests.

17.5.2 Multi-Function

To optimize the functionality in a small package, the pins provide multiple available functions through signal multiplexing. the PINMUX register controls the signals on the external pins. For details, please refer to [GPIOx_PINMUX](#) and section [17.4.2](#).

17.5.3 GPIO function

During and just after reset, the GPIO functions are active and the I/O ports are configured in input mode, except RST and Arm debug interface. User can program PINMUX Register to change I/O ports from other function to GPIO function.

When the pin is configured as output, the value written to the output data register (GPIOx_ODR) is output on the I/O pin. It is possible to use the output driver in push-pull mode or open-drain mode (only the N-MOS is activated when 0 is output). The input data register (GPIOx_IDR) captures the data present on the I/O pin at every AHB clock cycle.

All GPIO pins have weak internal pull-up and pull-down resistors, which can be activated or not depending on the value in the GPIOx_PU and GPIOx_PD register.

17.5.4 Programming guide

Firstly, after reset, all I/O ports are in GPIO input mode, except RST and Arm debug interface.

Secondly, software can program PINMUX to map/remap I/O function.

The software can program external interrupts for I/O as GPIO functions. When MCU in low power mode, external interrupt uses internal LPOSC with 32 kHz.

17.6 Memory map and register descriptions

Table 17-2 GPIO register map

GPIO: 0x40001000

ADDRESS	TITLE	DESCRIPTION
GPIO + OFFSET	GPIOx_CR	Port configuration
GPIO + OFFSET	GPIOx_IDR	Port input data
GPIO + OFFSET	GPIOx_ODR	Port output data
GPIO + OFFSET	GPIOx_BSRR	Port set/reset
GPIO + OFFSET	GPIOx_BRR	Port reset
GPIO + OFFSET	GPIOx_PD	Pull-down enable
GPIO + OFFSET	GPIOx_PU	Pull-up enable
GPIO + OFFSET	GPIOx_E4_E2	Driving Capability selection
GPIO + OFFSET	GPIOx_PINMUX	Multi-function selection
GPIO + OFFSET	GPIOx_PR	External interrupt flag Pending
GPIO + OFFSET	GPIOx_IMR	Interrupt mask on line
GPIO + OFFSET	GPIOx_RTSR	Rising edge trigger event configuration
GPIO + OFFSET	GPIOx_FTSR	Falling edge trigger event configuration
GPIO + OFFSET	GPIOx_EXTICR	external interrupt

GPIOx_CR register(x=0...4) OFFSET ADDR = (12'h00, 12'h30, 12'h60, 12'h90, 12'hc0)

BI T	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
d e f a u l t	0															
BI T	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	mod e(16 *x+1 5)	mod e(16 *x+1 4)	mod e(16 *x+1 3)	mod e(16 *x+1 2)	mod e(16 *x+1 1)	mod e(16 *x+1 0)	mod e(16 *x+9)	mod e(16 *x+8)	mod e(16 *x+7)	mod e(16 *x+6)	mod e(16 *x+5)	mod e(16 *x+4)	mod e(16 *x+3)	mod e(16 *x+2)	mod e(16 *x+1)	mod e(16 *x+0)

Bit(s)	Name	Description
[31:16]	Reserved	No use
[15:0]	Mode	Mode(y): Port y configuration bits These bits are written by software to configure the I/O direction mode. 0: Input (reset state) 1: output mode

GPIOx_IDR register(x=0...4)

OFFSET ADDR = (12'h04, 12'h34, 12'h64, 12'h94, 12'hc4)

BI T	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
de fa ul t	0															
BI T	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IDR(16*x+15)	IDR(16*x+14)	IDR(16*x+13)	IDR(16*x+12)	IDR(16*x+11)	IDR(16*x+10)	IDR(16*x+9)	IDR(16*x+8)	IDR(16*x+7)	IDR(16*x+6)	IDR(16*x+5)	IDR(16*x+4)	IDR(16*x+3)	IDR(16*x+2)	IDR(16*x+1)	IDR(16*x+0)
W																
de fa ul t	0															

Bit(s)	Name	Description
[15:0]	IDR	IDR(y): Port y input data These bits are read-only. They contain the input value of the corresponding I/O port.

GPIOx_ODR register(x=0...4) OFFSET ADDR = (12'h08, 12'h38, 12'h68, 12'h98, 12'hc8)

BI T	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
d e f a u l t	0															
BI T	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	OD R(16*x+15)	OD R(16*x+14)	OD R(16*x+13)	OD R(16*x+12)	OD R(16*x+11)	OD R(16*x+10)	OD R(16*x+9)	OD R(16*x+8)	OD R(16*x+7)	OD R(16*x+6)	OD R(16*x+5)	OD R(16*x+4)	OD R(16*x+3)	OD R(16*x+2)	OD R(16*x+1)	OD R(16*x+0)
W																
d e f a u l t	0															

Bit(s)	Name	Description
[15:0]	ODR	ODR(y): Port(y) output data These bits can be read and written by software.
		Note: For PIN set/reset, the ODR bits can be individually set and reset by writing to the GPIOx_BSRR register (x = A,B,C,D) and GPIOx_BRR register (x = A,B,C,D).

GPIOx_BSRR register(x=0...4) OFFSET ADDR = (12'h0c, 12'h3c, 12'h6c, 12'h9c, 12'hcc)

BI T	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	BR(16*x+15)	BR(16*x+14)	BR(16*x+13)	BR(16*x+12)	BR(16*x+11)	BR(16*x+10)	BR(16*x+9)	BR(16*x+8)	BR(16*x+7)	BR(16*x+6)	BR(16*x+5)	BR(16*x+4)	BR(16*x+3)	BR(16*x+2)	BR(16*x+1)	BR(16*x+0)
W																
de fa ult	0															
BI T	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BS(16*x+15)	BS(16*x+14)	BS(16*x+13)	BS(16*x+12)	BS(16*x+11)	BS(16*x+10)	BS(16*x+9)	BS(16*x+8)	BS(16*x+7)	BS(16*x+6)	BS(16*x+5)	BS(16*x+4)	BS(16*x+3)	BS(16*x+2)	BS(16*x+1)	BS(16*x+0)
W																
de fa ult	0															

Bit(s)	Name	Description
[31:16]	BR	BR(y): Port(y) Reset bit y These bits are write-only. A read to these bits returns the value 0x0000 0: No action on the corresponding ODRy bit 1: Reset the corresponding ODRy bit
[15:0]	BS	BS(y): Port(y) set bit y These bits are write-only and can be accessed in word, half-word or byte mode. A read to these bits returns the value 0x0000. 0: No action on the corresponding ODRy bit 1: Sets the corresponding ODRy bit Note : if BS and BR both configured, BR has higher priority.

GPIOx_BRR register(x=0...4) OFFSET ADDR = (12'h10, 12'h40, 12'h70, 12'ha0, 12'hd0)

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
default	0															
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0															
W	BR(16*x+15)	BR(16*x+14)	BR(16*x+13)	BR(16*x+12)	BR(16*x+11)	BR(16*x+10)	BR(16*x+9)	BR(16*x+8)	BR(16*x+7)	BR(16*x+6)	BR(16*x+5)	BR(16*x+4)	BR(16*x+3)	BR(16*x+2)	BR(16*x+1)	BR(16*x+0)
default	0															

Bit(s)	Name	Description
[15:0]	BR	BR(y): Port(y) Reset bit y These bits are write-only. A read to these bits returns the value 0x0000 0: No action on the corresponding ODRy bit 1: Reset the corresponding ODRy bit

GPIOx_PD register(x=0...4) OFFSET ADDR = (12'h18, 12'h48, 12'h78, 12'ha8, 12'hd8)

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
default	0															
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PD(16*x+15)	PD(16*x+14)	PD(16*x+13)	PD(16*x+12)	PD(16*x+11)	PD(16*x+10)	PD(16*x+9)	PD(16*x+8)	PD(16*x+7)	PD(16*x+6)	PD(16*x+5)	PD(16*x+4)	PD(16*x+3)	PD(16*x+2)	PD(16*x+1)	PD(16*x+0)
W																
default																

Bit(s)	Name	Description
[0]	PD	PD (y): Pull-down enable
[1]		0: disable pull-down
[2]		1: enable pull-down (typical value: 75 KΩ)

Bit(s)	Name	Description
.....		These bits can be read and written by software. Note: Pull-up and Pull-down are not supported enable at the same time

GPIOx_PU register(x=0...4) OFFSET ADDR = (12'h1c, 12'h4c, 12'h7c, 12'hac, 12'hdc)

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
default	0															
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PU(16*x+15)	PU(16*x+14)	PU(16*x+13)	PU(16*x+12)	PU(16*x+11)	PU(16*x+10)	PU(16*x+9)	PU(16*x+8)	PU(16*x+7)	PU(16*x+6)	PU(16*x+5)	PU(16*x+4)	PU(16*x+3)	PU(16*x+2)	PU(16*x+1)	PU(16*x+0)
W																
default	0															

Bit(s)	Name	Description
[0]	PU	PU (y): Pull-up enable
[1]		0: disable pull-up
[2]		1: enable pull-up (typical value: 75 KΩ)
.....		These bits can be read and written by software. Note: Pull-up and Pull-down are not supported enable at the same time

GPIOx_E4_E2 register(x=0...4) OFFSET ADDR = (12'h20, 12'h50, 12'h80, 12'hb0, 12'he0)

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	E4_E2		E4_E2		E4_E2		E4_E2		E4_E2		E4_E2		E4_E2		E4_E2	
W	(16*x+15)		(16*x+14)		(16*x+13)		(16*x+12)		(16*x+11)		(16*x+10)		(16*x+9)		(16*x+8)	
default	0															
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	E4_E2		E4_E2		E4_E2		E4_E2		E4_E2		E4_E2		E4_E2		E4_E2	
W	(16*x+7)		(16*x+6)		(16*x+5)		(16*x+4)		(16*x+3)		(16*x+2)		(16*x+1)		(16x+0)	
default	0															

Bit(s)	Name	Description
[1:0]	E4_E2	E4_E2 (y): Driving Capability selection
[3:2]		These bits can be read and written by software.
[5:4]		00: 4mA
.....		01: 8mA

Bit(s)	Name	Description
		10: 12mA
		11: 16mA

GPIOx_PINMUX register(x=0...6)

OFFSET ADDR = (12'h140, 12'h144, 12'h148, 12'h14C, 12'h150, 12'h154, 12'h158)

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

Bit(s)	Name	Description
		PINMUXx (y): Multi-function selection
		These bits are written by software to configure alternate function I/Os
		000: function 0
[2:0]	PINMUXx	001: function 1
[5:3]		010: function 2
[8:6]		011: function 3
.....		100: function 4
		101: function 5
		110: function 6
		111: function 7

GPIOx_PR register(x=0)

OFFSET ADDR = (12'h160)

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
default	0															
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PR(15)	PR(14)	PR(13)	PR(12)	PR(11)	PR(10)	PR(9)	PR(8)	PR(7)	PR(6)	PR(5)	PR(4)	PR(3)	PR(2)	PR(1)	PR(0)
W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit(s)	Name	Description
[0]	PR	PR (y): External interrupt flag Pending bit
[1]		0: No trigger request occurred
[2]		1: The selected trigger request occurred
.....		This bit is set when the selected edge event arrives on the external interrupt line. This bit is cleared by writing a '1' to the bit.

GPIOx_IMR register(x=0)

OFFSET ADDR = (12'h164)

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

Bit(s)	Name	Description
[0]	IMR	IMR (y): Interrupt mask on line y
[1]		0: Interrupt request from Line y is masked
[2]		1: Interrupt request from Line y is not masked
.....		

GPIOx_RTSR register(x=0)

OFFSET ADDR = (12'h168)

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

Bit(s)	Name	Description
[0]	RTSR	RTSR (y): Rising edge trigger event configuration bit of line y
[1]		0: Rising edge trigger disabled (for Event and Interrupt) for input line y
[2]		1: Rising edge trigger enabled (for Event and Interrupt) for input line y
.....		

GPIOx_FTSR register(x=0)
OFFSET ADDR = (12'h16C)

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

Bit(s)	Name	Description
[0]	FTSR	FTSR (y): Falling edge trigger event configuration bit of line y
[1]		0: Falling edge trigger disabled (for Event and Interrupt) for input line y
[2]		1: Falling edge trigger enabled (for Event and Interrupt) for input line y
.....		

GPIOx_EXTICR register(x=0...3)
OFFSET ADDR = (12'h170, 12'h174, 12'h178, 12'h17C)

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
default																
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	EXTI(4*x+3)				EXTI(4*x+2)				EXTI(4*x+1)				EXTI(4*x+0)			
W																
default	0				0				0				0			

Bit(s)	Name	Description
EXTI (y): EXTI y configuration		
These bits are written by software to select the source input for the EXTI(y)		
[3:0]	EXTI	external interrupt.
[7:4]		0000: PA[x] pin
[11:8]		0001: PB[x] pin
.....		0010: PC[x] pin
		0011: PD[x] pin
		0100: PE[x] pin

18 I2C

18.1 Introduction

I2C(Inter-IC) is a two-wire serial interface. The two signals are SCL and SDA. SCL is a clock signal that is driven by the master. SDA is bi-directional data signal that can be driven by either the master or the slave. This generic interface supports the master and slave role both, and conforms to the I2C specification.

18.2 Features

- Support master and slave mode operation.
- Support I2C standard mode 100 kHz and fast mode 400 kHz.
- Master switch to slave mode automatically while arbitration lost.
- Master programmable transmit bit rate.
- Slave address identification interrupt.
- Slave 10-bit address extension.
- Slave support low power mode wakeup.
- Slave support monitor function.
- Programmable input glitch filter.
- Support SCL stretch.
- Software-controlled acknowledge bit.
- Interrupt-driven byte-by-byte data transfer.
- Bus START/STOP detection.
- Support TX and RX DMA operation.

18.3 Block diagram

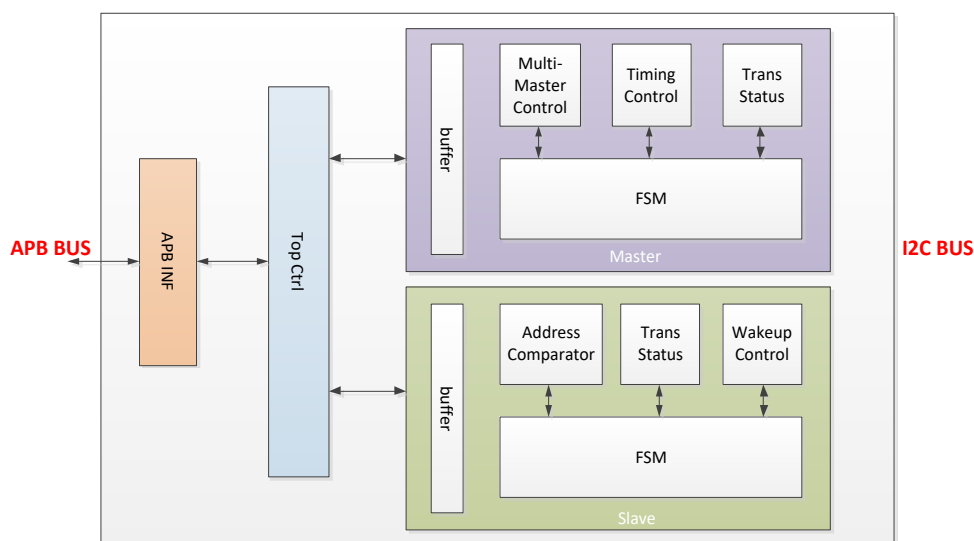


Figure 18-1 I2C block diagram

18.4 Mode of operation

I2C module supports wakeup function which can wake up MCU from STOP mode. When I2C is configured as slave mode, user can enable wakeup function by set WUEN bit of CONTROL_1 register.

When the MCU is entered STOP mode, I2C slave is waiting for an address match. If a master sends a match-address to I2C slave, then MCU will be waked up from STOP mode after receiving all match address.

18.5 Memory map and register descriptions

Table 18-1 I2C register map

I2C1 Base address: 0x4000e000

I2C2 Base address: 0x4000f000

Address	Name	Width	Description
I2Cx +0x000	ADDR_1	8	Address register 1
I2Cx +0x004	ADDR_2	8	Address register 2
I2Cx +0x008	SAMPLE_CNT	8	Baud rate config register 1
I2Cx +0x00c	STEP_CNT	8	Baud rate config register 2
I2Cx +0x010	CONTROL_1	8	Control register 1
I2Cx +0x014	CONTROL_2	8	Control register 2
I2Cx +0x018	CONTROL_3	8	Control register 3
I2Cx +0x01c	CONTROL_4	8	Control register 4
I2Cx +0x020	STATUS_1	8	Status register 1
I2Cx +0x024	STATUS_2	8	Status register 2
I2Cx +0x028	DGL_CFG	8	Deglintch config register
I2Cx +0x02c	DATA	9	Data port register
I2Cx +0x030	START_STOP	8	Master START STOP signal control register

I2Cx+0x000 ADDR1 Address Register 1

Bit	31:8	7	6	5	4	3	2	1	0
Name		AD[7:1]							
Type		R/W							
Reset		1	1	1	1	1	1	1	

Bit	Mnemonic	Name	Description
31:8			
7:1	AD[7:1]	Slave Address	Specifies the address when I2Cx is slave mode for the 7-bit address scheme and the lower seven bits in the 10-bit address scheme.
0			

I2Cx+0x004 ADDR_2 Address Register 2

Bit	31:8	7	6	5	4	3	2	1	0
Name							AD[10:8]		
Type							R/W		
Reset							1	1	1

Bit	Mnemonic	Name	Description
31:8			
7:3			
2:0	AD[10:8]	Slave Address Extend	Specifies the address when I2Cx is slave mode Contains the upper three bits of the address in the 10-bit address scheme. This field is valid only while the ADEXT bit is set.

I2Cx+0x008 SAMPLE_CNT Baud Rate Config Register 1

Bit	31:8	7	6	5	4	3	2	1	0
Name		SAMPLE_CNT_DIV							
Type		R/W							
Reset		0	0	0	0	0	1	0	0

Bit	Mnemonic	Name	Description
31:8			
7:0	SAMPLE_CNT_DIV	SAMPLE_CNT_DIV	This adjusts the width of each sample. sample width = (sample_cnt_div + 1) * T _{blk}

I2Cx+0x00c STEP_CNT Baud Rate Config Register 2

Bit	31:8	7	6	5	4	3	2	1	0
Name		STEP_CNT_DIV							
Type		R/W							
Reset		0	0	0	0	0	1	0	0

Bit	Mnemonic	Name	Description
31:8			
7:0	STEP_CNT_DIV	STEP_CNT_DIV	Specifies the number of samples per half pulse width, i.e. each high or low pulse half_baudrate width = (step_cnt_div + 1) * sample width Note: minimum of STEP_CNT_DIV is 3 propose!

I2Cx+0x010 CONTROL_1 Control Register 1

Bit	31:8	7	6	5	4	3	2	1	0
Name		I2CEN	I2CIE	MSTR	TX	TACK	WUEN		

Type		R/W	R/W	R/W	R/W	R/W	R/W		
Reset		0	0	0	0	0	0		

Bit	Mnemonic	Name	Description
31:8			
7	I2CEN	I2C_EN	I2C module enable 1: enable 0: disable
6	I2CIE	I2C_IE	I2C interrupt enable 1: enable the IIC module interrupt 0: disable
5	MSTR	MSTR	I2C operation mode select 1: master 0: slave <i>Note: if ARB_LOST is set, this bit auto clear.</i>
4	TX	TX	Transmission direction select 1: transmit(TX) 0: receive(RX) Selects the direction of master. In master mode this bit must be set according to the type of transmission required. Therefore, for master address cycles, this bit is always set.
3	TACK	TACK	Acknowledge control Specifies the value driven onto the SDA during data acknowledge cycles for both master and slave receivers. 1: NACK will sent to the bus on the following(next) receiving byte 0: ACK will sent to the bus on the following(next) receiving byte <i>Note: slave address byte ACK bit is hardware response auto, software not care this field. ACK is send if slave address match. Otherwise, NACK is sent.</i>
2	WUEN	WAKEUP_EN	Wakeup function enable The I2C slave mode can wake the MCU from low power mode which no peripheral bus running when slave address matching occurs. 1: enable the wakeup function in low power mode. 0: disable the wakeup function. <i>Note: when a 7bit address, 2 byte of 10bit address(if ADEXT=1), or general call address(GCAEN=1) match occurs when I2C module is in slave mode, the I2C module will generates a request to wake up MCU from low power mode.</i>
1			
0			

I2Cx+0x014 CONTROL_2 Control Register 2

Bit	31:8	7	6	5	4	3	2	1	0
Name		GCAEN	ADEXT		SYNCEN	ARBEN			STREN
Type		R/W	R/W		R/W	R/W			R/W
Reset		0	0		0	0			0

Bit	Mnemonic	Name	Description
31:8			
7	GCAEN	GCA_EN	Slave General Call Enable 1: enable 0: disable
6	ADEXT	ADEXT	Slave Address Extension 1: 10-bit address scheme 0: 7-bit address scheme
5			
4	SYNCEN	SYNC_EN	Master SCL Sync Enable enable the SCL synchronization function of master 1: enable 0: disable
3	ARBEN	ARB_EN	Master Arbitration Enable enable the master's arbitration function. 1: enable 0: disable Note: if I2C used in multi-master system, multi-master function must set both SYNC_EN and ARB_EN. Note: if I2C used in single master system, and slave have SCL stretch function, then can set SYNC_EN only.
2			
1			
0	STREN	STR_EN	Slave SCL stretch enable enable this bit, slave hardware will stretch SCL low after 9 th SCL falling per byte. 1: enable 0: disable Note: after SAMF=1, if SRW=1, RXFF=1 will cause SCL stretch; otherwise SRW=0, TXEF=1 will cause SCL stretch; so when enable STR_EN and slave is transmitter(TX), after 9 th SCL falling per byte (include address byte) slave will stretch SCL, until write DATA register; similarly, when slave is receiver(RX), after 9 th SCL falling per byte(without address byte), slave will stretch SCL, until read DATA register. Note: In slave mode, when enable GCA_EN or MNTEN, slave can not stretch SCL.

I2Cx+0x018 CONTROL_3 Control Register 3

Bit	31:8	7	6	5	4	3	2	1	0
Name		RXOF IE	TXUF IE	RXF IE	TXEM IE			NACK IE	MNT EN
Type		R/W	R/W	R/W	R/W			R/W	R/W
Reset		0	0	0	0			0	0

Bit	Mnemonic	Name	Description
31:8			
7	RXOFIE	SLA_RXOF_IE	Slave RX Buffer Overflow Error Interrupt Enable 1: enable 0: disable
6	TXUFIE	SLA_TXUF_IE	Slave TX Buffer Underflow Error Interrupt Enable 1: enable 0: disable
5	RXFIE	SLA_RXF_IE	Slave RX Buffer Full Interrupt Enable 1: enable 0: disable
4	TXEIE	SLA_TXE_IE	Slave TX Buffer Empty Interrupt Enable 1: enable 0: disable
3			
2			
1	NACKIE	NACK_IE	NACK Get Interrupt Enable 1: enable 0: disable Note: do not enable this bit when I2C is slave mode.
0	MNTEN	SLA_MNT_EN	Slave Monitor Function Enable 1: enable 0: disable

I2Cx+0x01c CONTROL_4 Control Register 4

Bit	31:8	7	6	5	4	3	2	1	0
Name								DMA RXEN	DMA TXEN
Type								R/W	R/W
Reset								0	0

Bit	Mnemonic	Name	Description
31:8			
7			
6:2			
1	DMARXEN	DMA_RX_EN	DMA RX Enable 1: enable 0: disable
0	DMATXEN	DMA_TX_EN	DMA TX Enable

Bit	Mnemonic	Name	Description
			1: enable 0: disable

I2Cx+0x020 STATUS_1 Status Register 1

Bit	31:8	7	6	5	4	3	2	1	0
Name		BND	SAMF	BUSY	ARB LOST	READY	SRW		RACK
Type		R/W	R/W	R	R/W	R	R		R/W
Reset		0	0	0	0	1	0		0

Bit	Mnemonic	Name	Description
31:8			
7	BND	Byte End	Byte End Flag 1: one byte transfer finish(include ACK bit, total 9 SCL) 0: transfer in progress, one byte transfer not finish After reset, BND is '0'. BND is set only during data transmission period which between START and STOP signal on the bus. BND will set after per 9 th SCL falling edge. When I2C is RX, read DATA register will clear this bit. Otherwise, when I2C is TX, write DATA register will clear this bit. Note: when I2C is slave and MNTEN=1, when TCF is '1', then read DATA register will clear this bit. Note: write '1' can clear this bit too.
6	SAMF	SAMF	Slave Address Match Flag 1: address match 0: address not match Note: this bit set by one of the following conditions. <ul style="list-style-type: none"> a. 7bit address, address match b. 10bit address, both 1st byte and 2nd byte match. And set after 2nd byte match. c. general call match Note: write '1' clear this bit.
5	BUSY	BUSY	Bus Busy Indicates the status of the bus, and valid for both slave and master mode. This bit is set when a START signal is detected and cleared when a STOP signal is detected on the bus by hardware. 1: bus is busy. 0: bus is idle.
4	ARBLOST	ARBLOST	Arbitration Lost Flag 1: arbitration lost 0: not lost Note: write '1' clear. Note: when arbitration lost, I2C master will switch to slave mode, and MSTR is cleared by hardware.
3	READY	READY	Internal Hardware Core Is Ready For New Command

			<p>This bit indicates the internal hardware states, and only valid for master mode.</p> <p>1: internal hardware is ready for software's new command 0: internal hardware is not ready</p> <p>Note: this bit is valid for master, and maybe used when master generate a START/STOP signal. When I2C module is master mode, write START_STOP[0] will generate a START signal on the bus, then DGL_CFG[4] is set, and this case must wait READY bit is 1, software can write DATA register to send address byte.</p> <p>Note: similar with START signal, write START_STOP[1] will generate a STOP signal on the bus, then DGL_CFG[6] is set. This case must wait READY bit is 1, software can write START_STOP[0], initial a next transmission.</p>
2	SRW	SRW	<p>Slave Read/Write Direction</p> <p>1: slave is transmitter(TX), master read from slave 0: slave is receiver(RX), master write to slave</p>
1			
0	RACK	RACK	<p>Acknowledge Received</p> <p>This field valid for transmitter(TX)</p> <p>1: No acknowledge signal detected 0: Acknowledge signal was received after one byte of data</p> <p>Note: if NACKIE=1, RACK=1 will set interrupt, write 1 clear.</p>

I2C+0x024 STATUS_2 Status Register 2

Bit	31:8	7	6	5	4	3	2	1	0
Name		IDLE				RXOF	TXUF	RXFF	TXEF
Type		R				R/W	R/W	R/W	R/W
Reset		1				0	0	0	1

Bit	Mnemonic	Name	Description
31:8			
7	IDLE	IDLE	<p>I2C core hardware state</p> <p>1: idle 0: not idle</p>
3	RXOF	RXOF	<p>Slave RX Buffer Overflow Flag</p> <p>1: RX buffer overflow 0: Not overflow</p> <p>Note: when RX buffer overflow, new received data not store to RX buffer. Note: write '1' clear.</p>
2	TXUF	TXUF	<p>Slave TX Buffer Underflow Flag</p> <p>1: TX buffer underflow 0: No underflow</p> <p>Note: when TX buffer underflow, send the last DATA again. Note: write '1' clear.</p>
1	RXFF	RXFF	<p>Slave RX Buffer Full Flag</p> <p>1: RX buffer full</p>

			0: Not full Note: read <i>DATA</i> register will clear this bit.
0	TXEF	TXEF	Slave TX Buffer Empty Flag 1: TX buffer empty 0: Not empty Note: write <i>DATA</i> register will clear this bit.

I2C+0x028 DGL_CFG Deglitch Config Register

Bit	31:8	7	6	5	4	3	2	1	0
Name		DGLEN	STOPF	SSIE	STARTF	DGL_CNT			
Type		R/W	R	R/W	R	R/W			
Reset		0	0	0	0	0	0	0	0

Bit	Mnemonic	Name	Description
31:8			
7	DGLEN	DGL_EN	Deglitch Filter Enable 1: enable 0: disable
6	STOPF	STOP FLAG	Bus STOP Flag hardware set this bit when the STOP signal is detected on the I2C bus. 1: STOP detected on I2C bus 0: No STOP detected on I2C bus Note: write "1" clear this bit.
5	SSIE	SSIE	Bus STOP or START Interrupt Enable 1: enable STRAT or STOP detection interrupt 0: disable Note: clear bus <i>START</i> or <i>STOP</i> interrupt: In the interrupt service routine, write '1' clear <i>STARTF</i> or <i>STOPF</i> flag , then interrupt clear auto.
4	STARTF	STARTF	Bus START Flag 1: START detected on I2C bus 0: No START detected Note: write '1' clear.
3:0	DGL_CNT	DGL_CNT	Deglitch Counter control the width of the glitch, in terms of I2C module clock cycles, that the filter must absorb. For any glitch whose size is less or equal to this width setting, the filter does not allow the glitch to pass. 0h: No filter/bypass 1-Fh: Filter glitches up to width of 1-15 clock cycles.

I2C+0x02c DATA Data Port Register

Bit	31:9	8	7	6	5	4	3	2	1	0
Name		MAK	DATA							
Type		R	R/W							
Reset		1	1	1	1	1	1	1	1	1


Bit	Mnemonic	Name	Description
31:9			
8	MAK	Monitor ACK	Slave Monitor Function ACK Bit For slave monitor, this field is the ACK bit from I2C bus. Note: MAK='1', NACK MAK='0', ACK Note: for monitor, DATA[7:0] is the data transfer on I2C bus, and DATA[8] is the ACK bit.
7:0	DATA	DATA	Data For master transmit(TX) mode, when data is written to this register, a data transfer is initiated. The most significant bit is sent first. For master receive(RX) mode, reading this register initiates a receiving sequence of the next byte. Note: when making the transition out of master receive mode, switch the I2C mode before reading the DATA register to prevent an inadvertent initiation of a master receive data transfer. Note: read this register, will return RX buffer data. master mode, RX buffer default 0xff; slave mode RX buffer default 0x1ff.

I2C+0x030 START_STOP START_STOP Register

Bit	31:8	7	6	5	4	3	2	1	0
Name								STOP	START
Type								R/W	R/W
Reset								0	0

Bit	Mnemonic	Name	Description
31:8			
1	STOP	STOP	Master STOP Trig write "1", master will send START signal. read this bit always return "0"
0	START	START	Master START Trig write "1", master will send START(RESTART) signal. read this bit always return "0"

Note: reserved bit must keep default value, not change it.

Note:  not used. read always return 0.

18.6 Functional description

18.6.1 Data flow & Algorithm

For transmitter, data is written to an 8-bit TX buffer, then load to TX shift refer to baud rate control logic. The output data is most significant bit first. The transmitter sample the acknowledge bit every 9th SCL per byte.

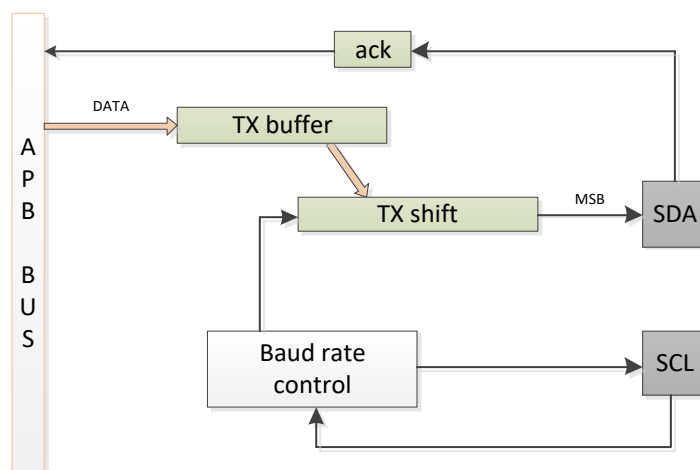


Figure 18-2 Data flow of transmitter

For receiver, RX shift sample and shift in the SDA line input data. Data received store into an 8-bit RX buffer after every 8th SCL per byte with the most significant bit first. The acknowledge bit is put on the SDA line at 9th SCL pulse.

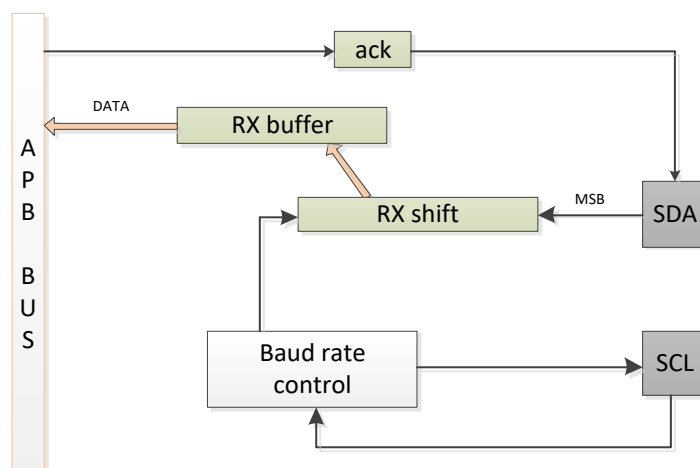


Figure 18-3 Data flow of Receiver

18.6.2 Input & Output timing

All transactions begin with a START (S) and terminated by a STOP (P). A high to low transition on the SDA line while SCL is high defines a START condition. A low to high transition on the SDA line while SCL is high defines a STOP condition (see [Figure 18-4](#)).

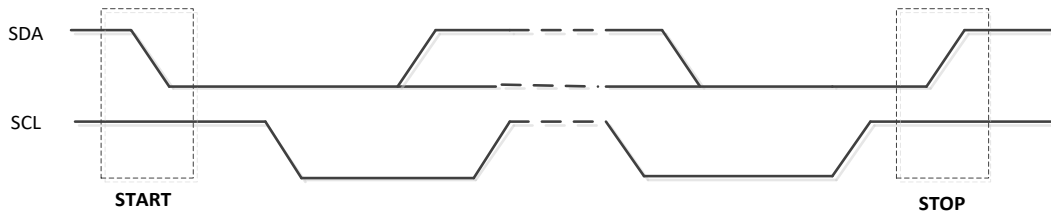


Figure 18-4 START and STOP conditions

Every byte put on the SDA line must be eight bits long. The number of bytes that can be transmitted per transfer is unrestricted. Each byte must be followed by an Acknowledge bit. Data is transferred with the Most Significant Bit (MSB) first (see Figure 18-5).

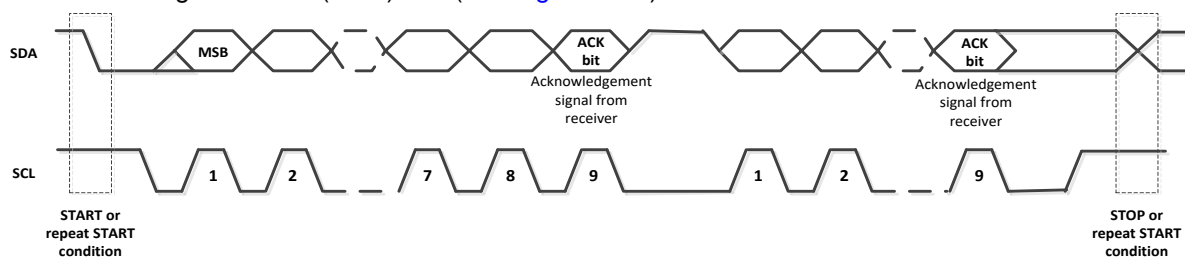


Figure 18-5 Data transmission format

18.6.3 Master SCL output timing set

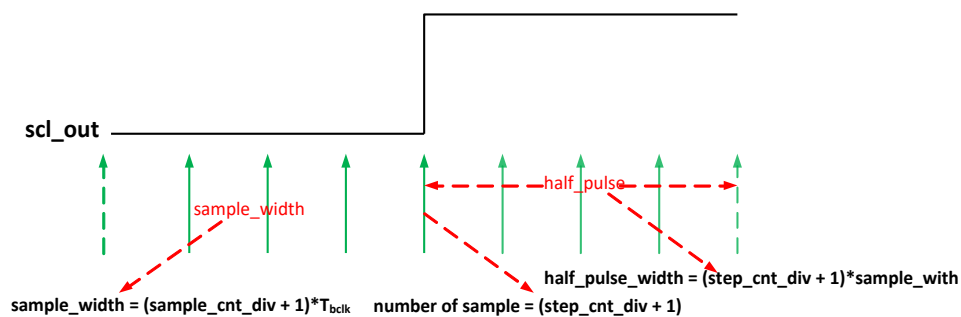


Figure 18-6 Baud rate generation

Baud rate: $f_{SCL} = f_{bclk} / (((SAMPLE_CNT_DIV + 1) * (STEP_CNT_DIV + 1)) * 2)$

f_{bclk} is the APB bus clock frequency(50 MHz).

18.6.4 Data transmission

This I2C module is a byte-by-byte completely serial interface, and need CPU's control every byte. Write *START_STOP[0]* '1' trigger a START signal onto the I2C bus, and initial a transmission. The transmission ends up with a STOP signal on the I2C bus. STOP signal is triggered by the *START_STOP[1]* written '1'. The *BND* flag will be set after every data byte finish. Software write or read data refer to the transmission direction after *BND=1*.

Write *BND* '1' can clear *BND* too. Software can write *BND* '1' instead of writing dummy data to *DATA* when the last byte finishes within master write slave mode.

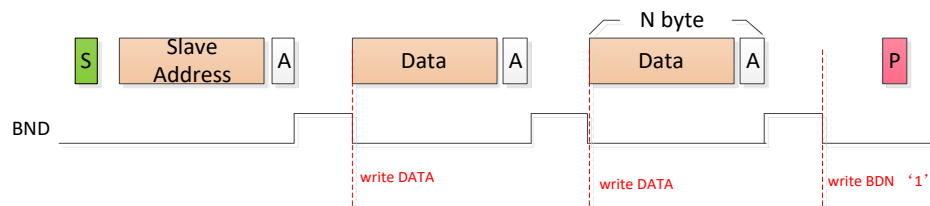


Figure 18-7 BND sequence of master write slave mode

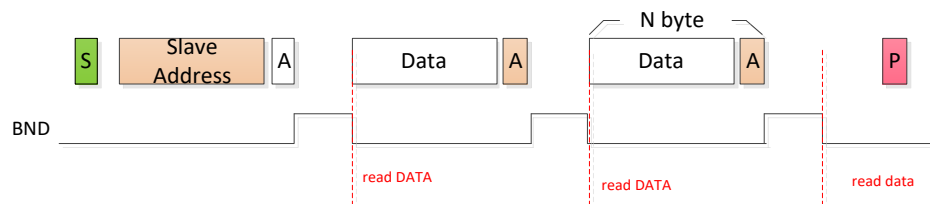


Figure 18-8 BND sequence of master read slave mode

18.6.5 DMA operation

This I2C module supports byte-by-byte DMA TX/RX requests transmission. DMA requests are generated only for data transfer. DMA valid for data transfer phase only, and START/STOP/RESTART signal triggered by software still. The DMA operation is different between transmitter and receiver. EOT_1 signal is set after the last but one byte data finish for receiver.

18.6.5.1 Master transmitter

For master transmitter, after START signal, the first byte is the slave address which defined in I2C protocol. This address byte is written by software. The DMA TX request (DMA_TX_REQ) will be set after 1st byte acknowledge bit (9th SCL). Notice that DMA_TX_REQ be set need an ACK condition (must a slave give back a low on 9th SCL). If NACKIE=1, I2C core will generate CPU interrupt request when a NACK occurs.

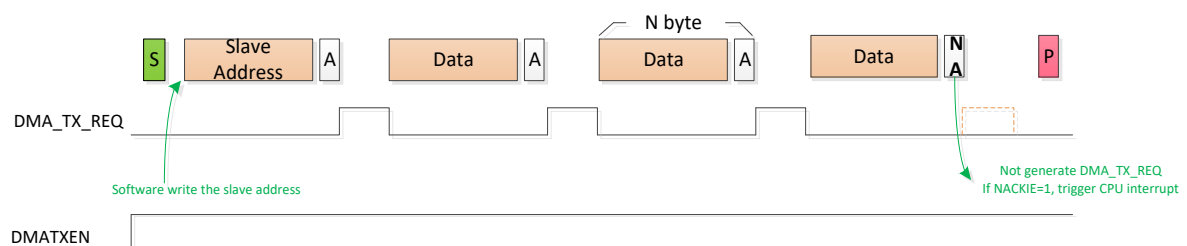


Figure 18-9 Master transmitter case 1

If the slave receiver ACK all the data byte and the DMA transmit finishes, I2C module generates the DMA TX request still because of I2C module can't recognize the transaction length. So software needs to write BND "1" clear this bit filed after DMA transmission finishes.

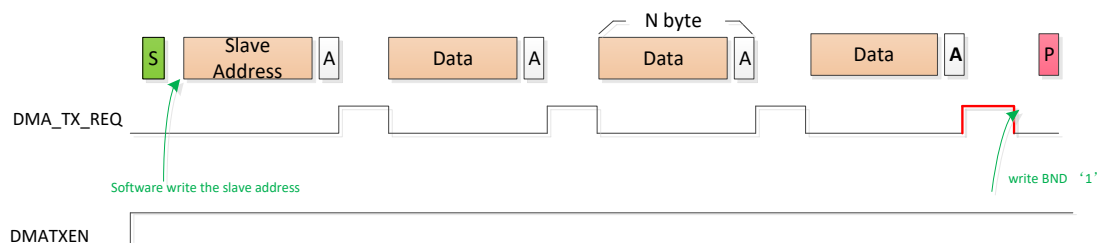


Figure 18-10 Master transmitter case 2

18.6.5.2 Master receiver

For the master receiver, START condition and slave address byte are triggered by software. The direction switch and dummy read DATA register action are necessary too. The first DMA RX request is generated by hardware automatically after 1st data byte finishes.

When the number of bytes to be received is equal to or greater than two, the DMA controller sends a hardware signal *eto_1*=1, corresponding to the last but one data byte (number_of_bytes - 1). I2C automatically sends a NACK after the next byte following *eto_1*=1. Software can generate a STOP condition in the DMA transfer complete interrupt routine if enabled.

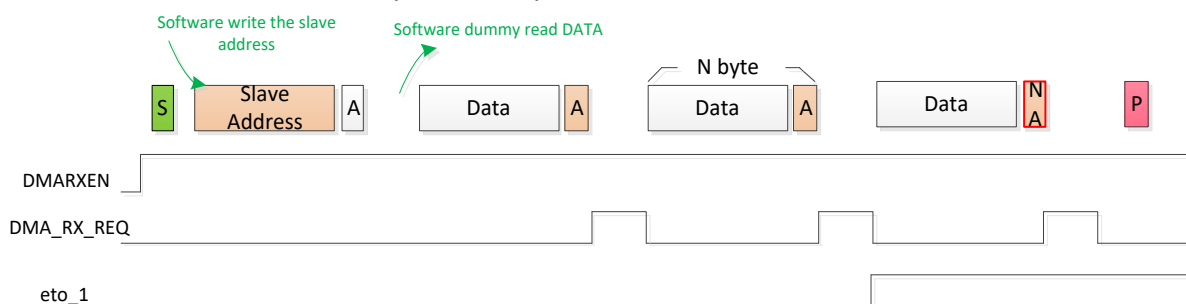


Figure 18-11 Master receiver

18.6.5.3 Slave transmitter

The slave is a transmitter if slave address matches and the LSB is “1” of address byte. I2C hardware generates the 1st DMA TX request after slave address byte acknowledge bit. DMA TX request will not be generated if master gives a NACK. I2C slave will generate CPU interrupt request if enable the *NACKIE* and *IICIE*.

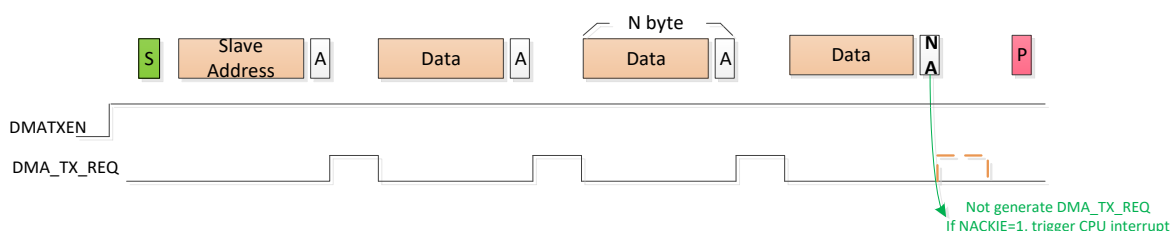


Figure 18-12 Slave transmitter

18.6.5.4 Slave receiver

The first DMA RX request is generated after 1st data byte finish for slave receiver. Similar to the master receiver, eto_1 precedes the last byte of data and replies with a NACK on the next data byte.

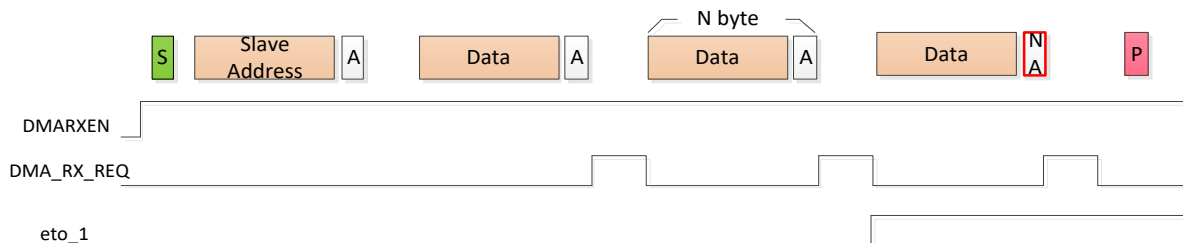


Figure 18-13 Slave receiver

18.6.6 Slave low power wakeup

I2C slave can turn to low power mode. Low power mode I2C slave can generate wake up signal wake up MCU if address matches. I2C slave cannot transmit or receive any data until STOP during wakeup sequence. So slave only ACK address byte if address matches, and all following data transmission is NACK.

I2C slave turn to low power mode must meet the following conditions:

- I2C slave is IDLE.
- WUEN bit is '1'.
- Software WFI instruction.



Figure 18-14 Wakeup sequence

18.6.7 Interrupt

I2C has a total of 10 interrupts.

Table 18-2 Interrupt summary

Interrupt	Flag	Local enable	Global enable
One Byte Transfer End	BND		IICIE
Slave Address Match	SAMF		IICIE
Arbitration Lost	ARBLOST		IICIE
Bus START Detected	START	SSIE	IICIE
Bus STOP Detected	STOP	SSIE	IICIE
RX Buffer Overflow	RXOF	RXOFIE	IICIE
TX Buffer Underflow	TXUF	TXUFIE	IICIE
RX Buffer Full	RXFF	RXFIE	IICIE
TX Buffer Empty	TXEF	TXEIE	IICIE
Ack Get	RACK	NACKIE	IICIE

DMARXEN=1 or DMATXEN=1, BDN, RXFF, and TXEF will not generate interrupt even enable corresponding enable bit.

18.7 Program guide

18.7.1 Description

The transfer of data would proceed as follows:

1. Suppose microcontroller A wants to send information to microcontroller B:
 - Microcontroller A(master), addresses microcontroller B(slave).
 - Microcontroller A(master-transmitter), sends data to microcontroller B(slave-receiver).
 - Microcontroller A terminates the transfer.
2. If microcontroller A wants to receive information from microcontroller B:
 - Microcontroller A(master), addresses microcontroller B(slave).
 - Microcontroller A(master-receiver) receives data from microcontroller B(slave-transmitter).
 - Microcontroller A terminates the transfer.

XXX Master send to slave

XXX Slave send to master

18.7.2 Master transmitter

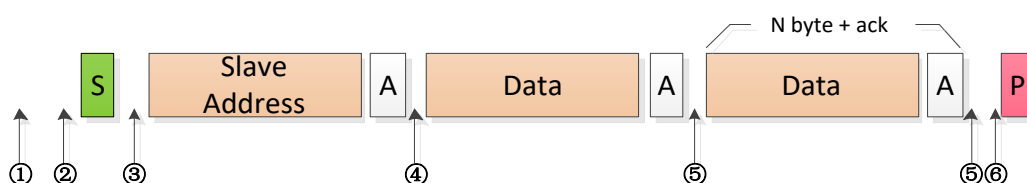


Figure 18-15 Master write operation sequence

- ① Set the timing(*SAMPLE_CNT*,*STEP_CNT*) and other function control.
register(*CONTROL_1*,*CONTROL_2*,*CONTROL_3*,*CONTROL_4*,) first, *CONTROL_1*[4] must be "1", and then enable I2C.
- ② Write *START_STOP*[0] "1", trigger START.
- ③ *STATUS_1*[5]="1", *STATUS_1*[3]="1", then write data(slave address) to *DATA*, *DATA*[0] is the direction bit, this is "0" .
- ④ *STATUS_1*[7]="1", *STATUS_1*[0]="1", write data to *DATA*.
- ⑤ Repeat step ④ if multi bytes transmission.

- ⑥ Write `START_STOP[1]` “1”.

18.7.3 Master receiver

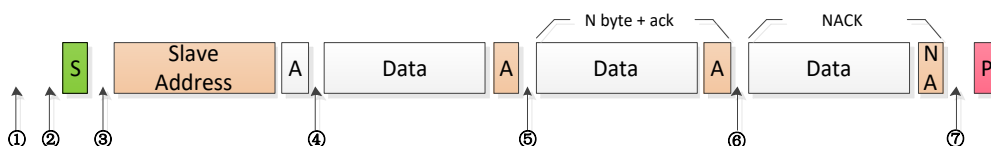


Figure 18-16 Master receiver operation sequence

- ① Set the timing(`SAMPLE_CNT`,`STEP_CNT`) and other function control register(`CONTROL_1`,`CONTROL_2`,`CONTROL_3`,`CONTROL_4`) first, `CONTROL_1[4]` must be “1”, and then enable I2C .
- ② Write `START_STOP[0]` “1”, trigger START.
- ③ `STATUS_1[5]` = “1”, `STATUS_1[3]` = “1”, then write data(slave address) to `DATA`, `DATA[0]` is the direction bit, this is “1” .
- ④ `STATUS_1[7]` = “1”, `STATUS_1[0]` = “1”, write `CONTROL_1[4]` “0”, then dummy read `DATA`, trigger data transmission
- ⑤ `STATUS_1[7]` = “1”, read `DATA`. if multi-byte transmission, repeat step ⑤.
- ⑥ `STATUS_1[7]` = “1”, write `CONTROL_1[3]` “1”, then read `DATA`.
- ⑦ `STATUS_1[7]` = “1”, write `CONTROL_1[4]` “1”, then read `DATA`(the last one byte), and then write `START_STOP[1]`, trigger STOP.

18.7.4 Master combined format

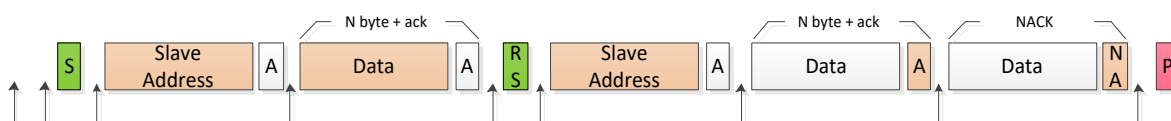


Figure 18-17 Combined format option sequence

Combined format can be master write slave first and read slave then, or master read slave first and then write slave.

All program configuration and sequence master write slave refer to [18.7.2 Master transmitter](#), and master read slave refer to [18.7.3 Master receiver](#). The only difference is the ‘RS’(RESTART). I2C module has not RS trig register, and this is implemented by `START_STOP[0]`. Write `START_STOP[0]` “1” will trigger a START signal if I2C bus is idle, and a RESTART signal will be triggered if I2C not idle.

18.7.5 Slave

Slave address match will generate a *SAMF* flag, and switch to receiver or transmitter controlled by address byte LMSB hardware auto. *SRW* indicates the direction of data transmission next. Software write data to *DATA* register after *SAMF* and *SRW*=1(master read). Every byte(9 SCL) finish, *BND* set "1". Software write data to *DATA* register if *SRW*=1, and read data from *DATA* register if *SRW*=0. *ACK* bit is sent by slave when master write. *CONTROL_1[3] TACK* bit control the next byte *ACK* bit.

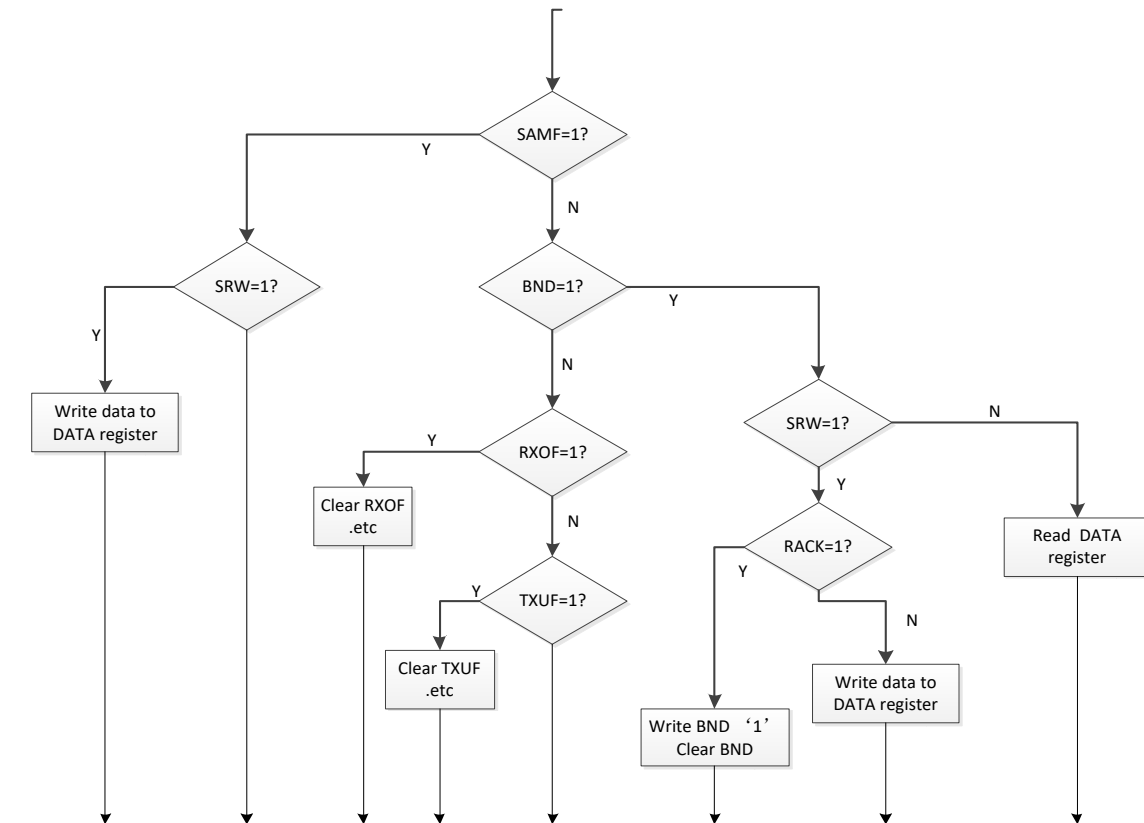


Figure 18-18 Typical I2C slave interrupt routine

19 SPI

19.1 Introduction

The SPI is a bit-serial, synchronous serial, and full duplex protocol. The SPI interface was developed by Motorola and has become a de facto standard. This SPI module is a 4-wire interface that includes master and slave both.

Figure 19-1 gives an example of the connection between SPI master and SPI slave.

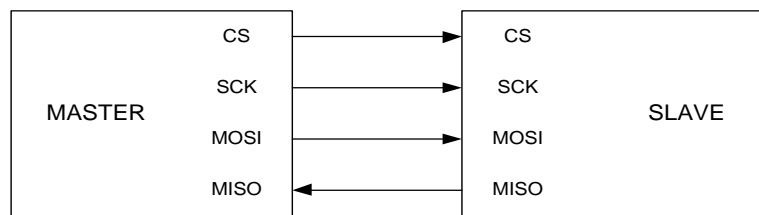


Figure 19-1 SPI system connection

19.2 Feature list

- Master mode or slave mode operation.
 - As master, the highest baud rate is $f_{bus}/2$ Hz (f_{bus} is APB bus clock)
 - As slave, the highest baud rate is 18 MHz
- Full-duplex.
- Master programmable transmit bit rate.
- Serial clock phase and polarity options.
- Configurable continuous or discontinuous CS (slave select) output.
- Mode error flag with CPU interrupt capability.
- Selectable MSB-first or LSB-first shifting.
- Configurable CS setup time, hold time and idle time.
- Configurable SCK high and low period.
- 4-16 bit transfer frame format selection.
- DMA mode.
- TX buffer underflow and RX buffer overflow flag with interrupt capability.
- Slave support low power mode wake up function.

19.3 Block diagram

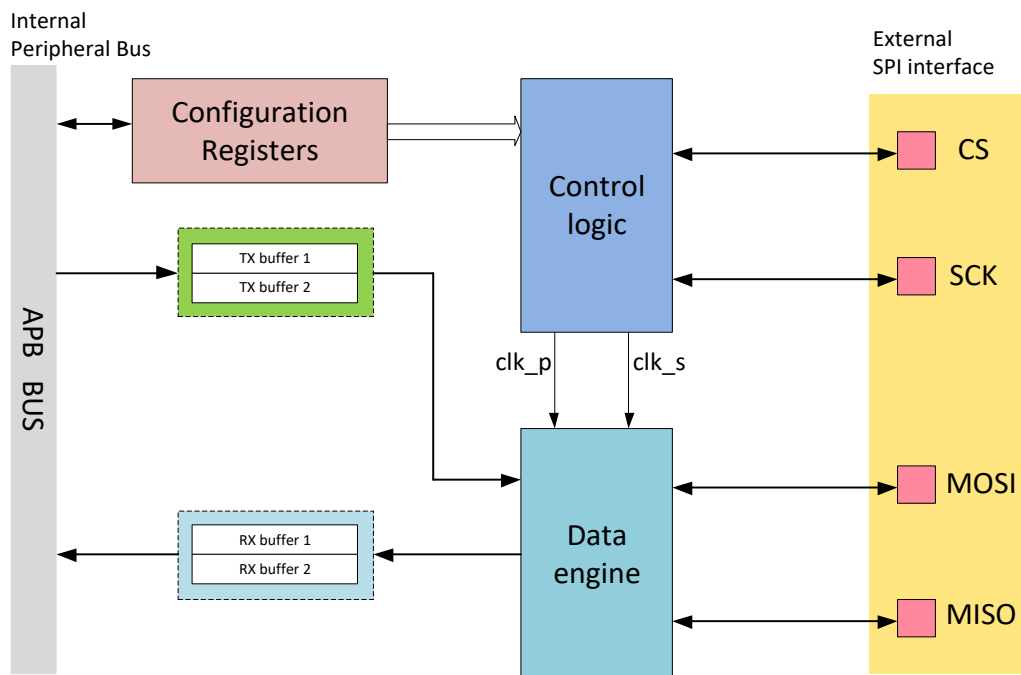


Figure 19-2 SPI block diagram

19.4 Data flow & Algorithm

For master mode, data is written to a 16-bit TX buffer, then loaded to shift register refer to baud rate control logic. The output data is most significant first or least significant first controlled by *TMSBF*. After the numbers SCK periods specified by *FRMSIZE*, shift register shift in single-byte data from MISO pin. Data received is stored into a 16-bit RX buffer.

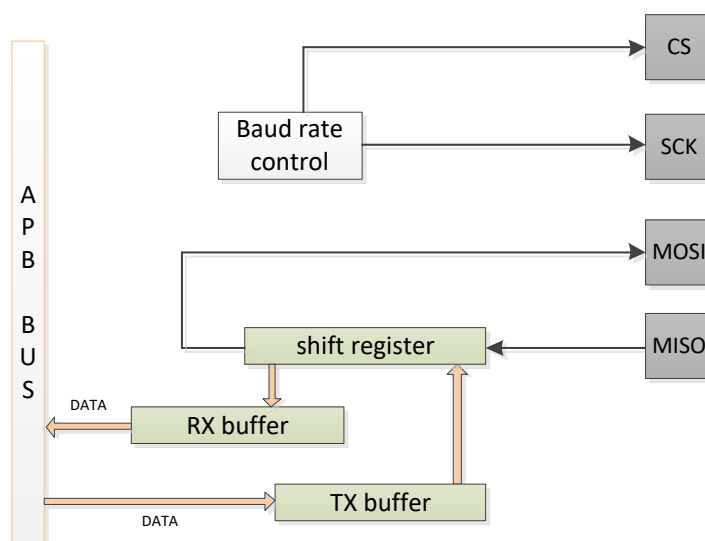


Figure 19-3 Data flow of master

For slave mode, data flow is similar to master mode. But the CS pin is the slave select input, and SCK is the SPI clock input from the master. Before a data transmission occurs, the CS pin of the slave SPI must be low. MOSI is the slave data input pin, and MISO is the data output pin.

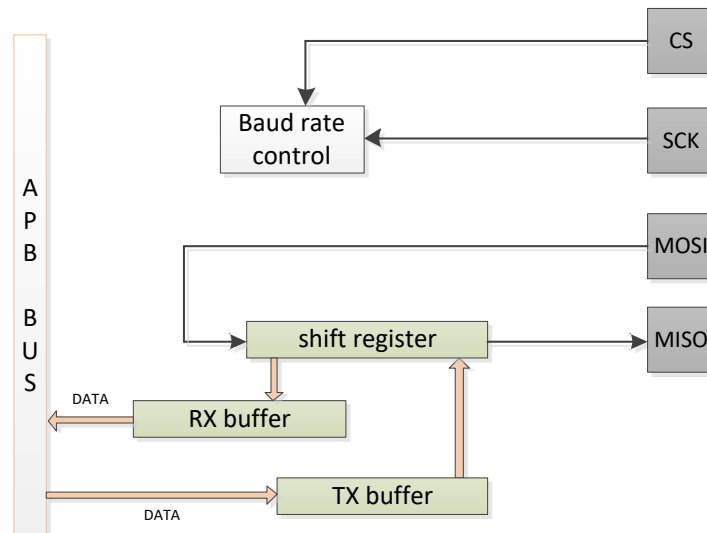


Figure 19-4 Data flow of slave

19.5 Input & Output timing

19.5.1 CPHA = 0 transfer format

The first edge on the SCK line is used to clock the first data bit of the slave into the master and the first data bit of the master into the slave. In some peripherals, the first bit of the slave's data is available at the slave's data out pin as soon as the slave is selected. In this format, the first SCK edge is issued a half cycle after CS has become low.

A half SCK cycle later, the second edge appears on the SCK line. When this second edge occurs, the value previously latched from the serial data input pin is shifted into the shift register.

After this second edge, the next bit of the SPI master data is transmitted out of the serial data output pin of the master to the serial input pin on the slave. This process continues for a total 16 edges on the SCK line, with data being latched on odd numbered edges and shifted on even numbered edges.

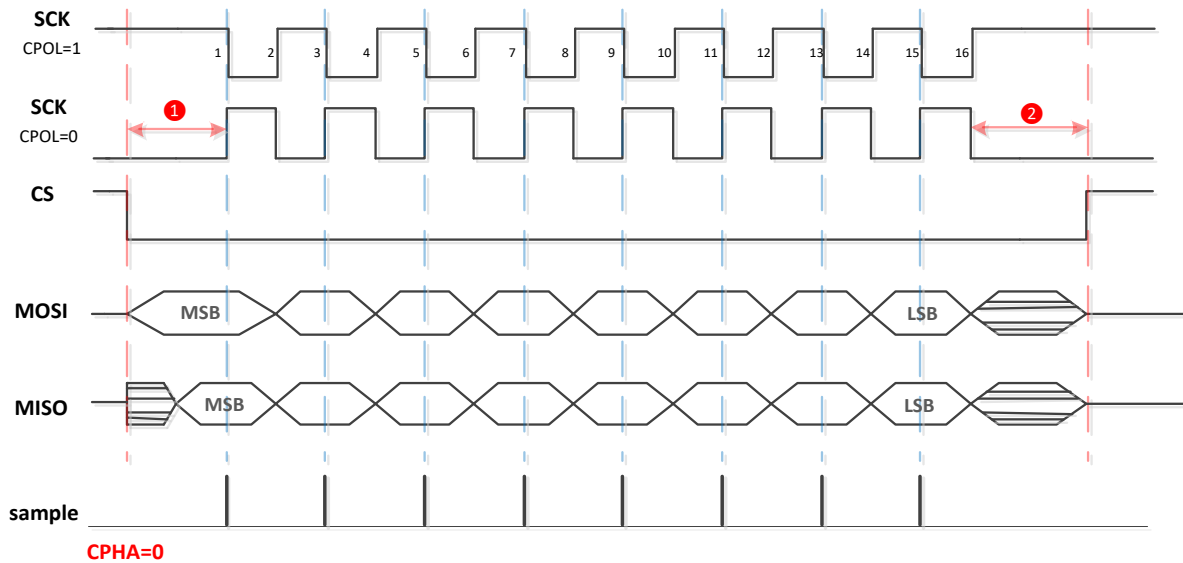


Figure 19-5 CPHA=0 Transmission Format

19.5.2 CPHA = 1 transfer format

Some peripherals require the first SCK edge before the first data bit becomes available at the data output pin, the second edge clocks data into the system.

The first edge of SCK occurs immediately after the half SCK clock cycle synchronization delay. This first edge commands the slave to transfer its first data bit to the serial data input pin of the master.

A half SCK cycle later, the second edge appears on the SCK pin. This is the latching edge for both the master and slave.

When the third edge occurs, the value previously latched from the serial data input pin is shifted into the shift register. After this edge, the next bit of the master data is coupled out of the serial data output pin of the master to the serial input pin on the slave.

This process continues for a total 16 edges on the SCK line with data being latched on even numbered edges and shifting taking place on odd numbered edges.

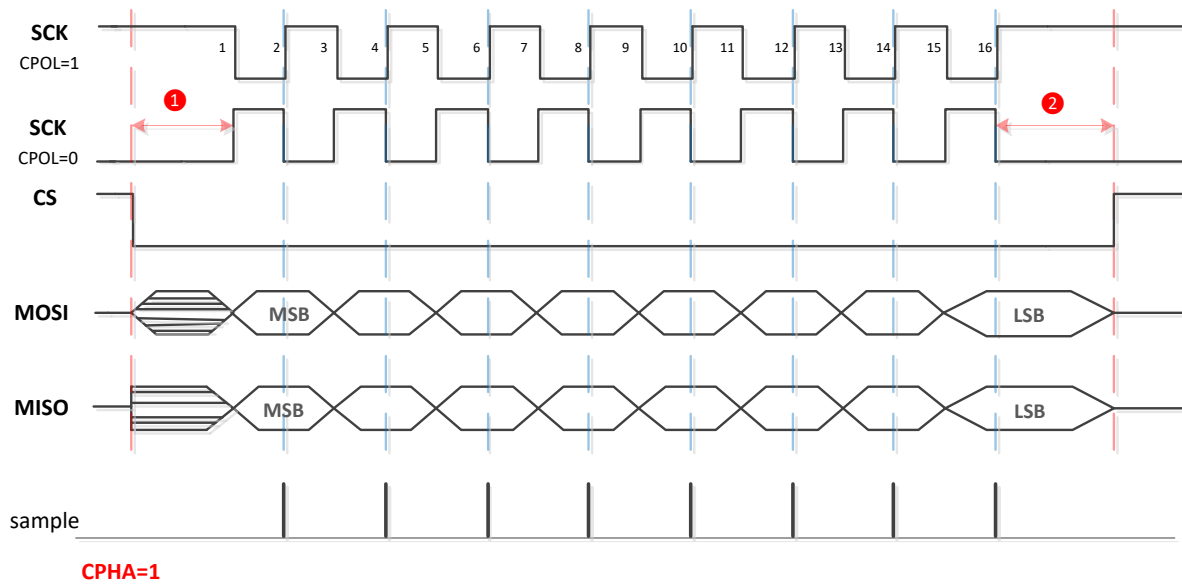


Figure 19-6 CPHA=1 Transmission Format

19.6 Master SCK output timing set

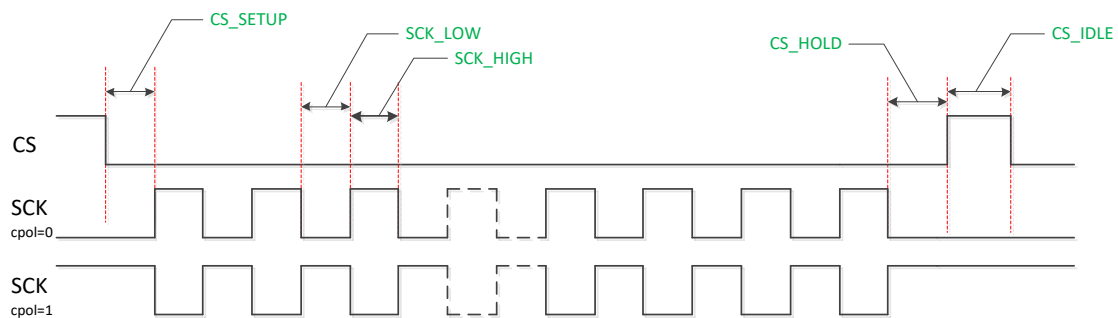


Figure 19-7 Baud rate generation

Baud rate: $f_{SCL} = f_{clk} / (SCK_LOW + 1 + SCK_HIGH + 1)$, f_{clk} is the APB bus clock frequency.

19.7 Master mode fault detect

MODEF is set if the CS pin has been driven to low before SPI master initial a transmission. Then master mode fault detect function is valid only when $MSTR=1$, $MODFEN=1$, $CSOE=1$.

The SCK is different from normal case if enable master mode fault detect function with $CPOL=0$. The SCK is high when IDLE state, and change to low then only if no mode fault error occurs. SCK keep high after transmission finishes.

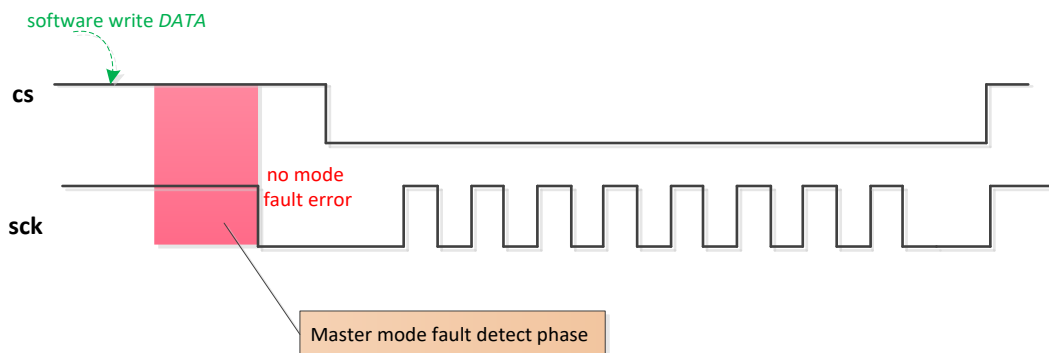


Figure 19-8 SCK output timing with mode fault detect enable

This mode fault detect function may can't detect the mode fault by master 1 with some special cases. If master 2 drive CS low during (1) period in Figure 19-9, master 1 will not set *MODEF*. Only master 2 drive CS low before (1) period that master 1 can set *MODEF*.

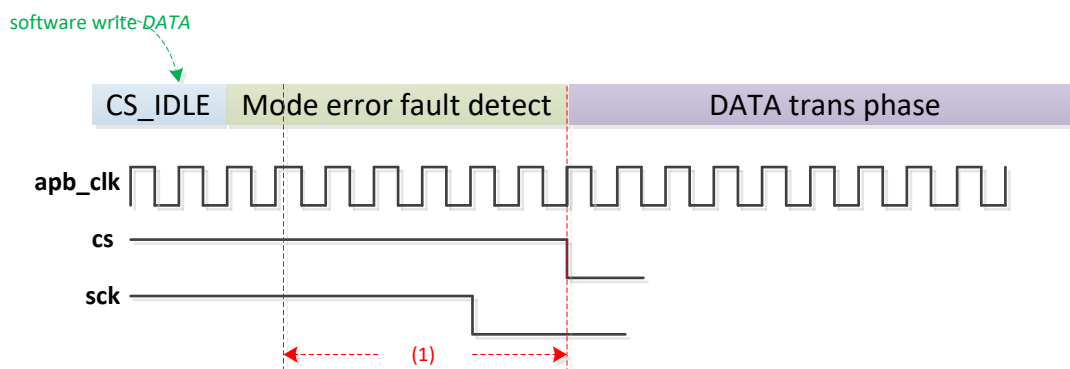


Figure 19-9 Limitation of mode fault detect

19.8 Slave low power wakeup

SPI slave in stop mode can generate an asynchronous interrupt to wake the CPU from the low power mode when receives a transmission.

To ensure SPI module low power wakeup function correct, system must follow some regulations. Before CPU enters low power mode, system must confirm that SPI module is in IDLE state. Software can check STATUS[8] IDLEF bit state. For master mode, tx buffers are empty, rx buffers are empty, and internal hardware is idle, IDLEF can be '1'. For slave mode, tx buffers are empty, rx buffers are empty, and CS is deassert(CS is high), IDLEF can be '1'. SPI module can't ensure data valid or wakeup function correct if CPU enter low power mode when SPI module is busy.

The slave generates asynchronous wakeup interrupt only when all of the following conditions apply:

- SPI module is slave mode.
- SPI slave is IDLE states.
- WUEN bit is '1'.
- The single-byte wakeup sequence transmission end.

CS high to low initial the wakeup phase, and slave generates the asynchronous wakeup request after the numbers SCK cycle that *FRMSIZE* specified. SPI slave can receive the data of wakeup phase byte. The RXFF flag will be set after chip wakeup finish (clock recover), and read data register will return the data master sent in wakeup phase byte. Only one byte can be transmitted during wakeup

phase. During the wakeup phase, a continuous transmission from a master would destroy the data received, and may lead to RXFF flag can't be set rightly.

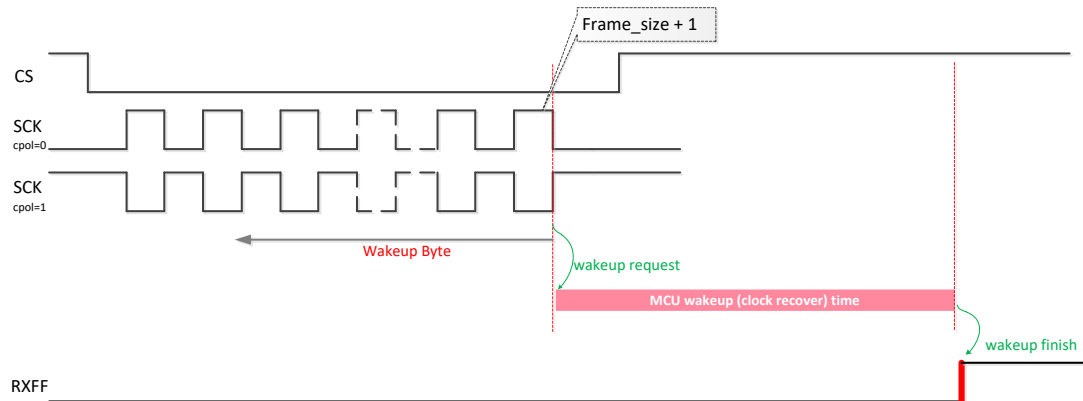


Figure 19-10 Wakeup sequence

19.9 Interrupt

The SPI has a total of five interrupts.

Table 19-1 Interrupt summary

Flag	Local enable
TXEF	TXEIE
RXFF	RXFIE
TXUF	TXUIE
RXOF	RXOIE
MODFF	MODFIE

19.10 Master CS continuous mode

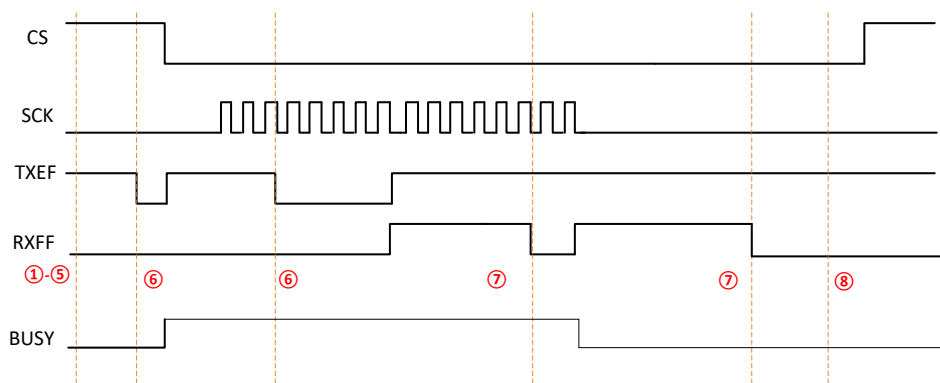


Figure 19-11 CS continuous mode

CS continuous output

1. Configure CFG0: *CS_SETUP*, *CS_HOLD*, *SCK_LOW*, *SCK_HIGH*.
2. Configure CFG1: *CS_IDLE*

3. Configure *FRMSIZE*, *CPHA*, *CPOL.RMSBF*, *TMSBF* etc.
4. Configure *CSOE*, *CONT_CS*, *MSTR*
5. *SPIEN*=1
6. *TXEF*=1, write data to *DATA*
7. *RXFF*=1, read data from *DATA*
8. Write *CSRLS* "1", release CS, then hardware goes to idle.

19.11 Master CS discontinuous output

1. Configure CFG0: *CS_SETUP*, *CS_HOLD*, *SCK_LOW*, *SCK_HIGH*.
2. Configure CFG1: *CS_IDLE*.
3. Configure *FRMSIZE*, *CPHA*, *CPOL.RMSBF*, *TMSBF* etc.
4. Configure *CSOE*, *CONT_CS*, *MSTR*.
5. *SPIEN*=1
6. *TXEF*=1, write data to *DATA*.
7. *RXFF*=1, read data from *DATA*.

CS change to low or high by hardware when CS discontinuous mode. So software does not care the *CSRLS*.

19.12 Slave mode

1. Configure *FRMSIZE*, *CPHA*, *CPOL.RMSBF*, *TMSBF* etc.
2. Configure *MSTR*.
3. *SPIEN*=1.
4. *TXEF*=1, write data to *DATA*.
5. *RXFF*=1, read data from *DATA*.

19.13 DMA mode

TXEF=1 will generate the DMA TX request.

RXFF=1 will generate the DMA RX request.

1. Initialize DMA.
2. Initialize SPI module, and enable *DMATXEN*, *DMARXEN*.
3. Wait DMA finish.
4. Other options similar to CS continuous or discontinuous mode.

19.14 Register definition

Table 19-2 SPI register map

SPI1 Base address: 0x4000c000

SPI2 Base address: 0x4000d000

ADDRESS	TITLE	DESCRIPTION
SPIx+0x000	CFG0	SPI CONFIGURATION 0 REGISTER
SPIx+0x004	CFG1	SPI CONFIGURATION 1 REGISTER
SPIx+0x008	CMD	SPI COMMAND REGISTER
SPIx+0x00c	STATUS	SPI STATUS REGISTER
SPIx+0x010	DATA	SPI DATA REGISTER
SPIx+0x014	CFG2	SPI CONFIGURATION 2 REGISTER

SPIx+0x000 CFG0 SPI Configuration Register 0

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name	CS_SETUP								CS_HOLD							
Type	R/W								R/W							
Reset	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	1
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	SCK_LOW								SCK_HIGH							
Type	R/W								R/W							
Reset	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	1

Bit	Name	Description
31:24	CS_SETUP	CS Setup Time The chip select setup time = (CS_SETUP_COUNT+1)*CLK_PERIOD, where CLK_PERIOD is the cycle time of the clock the SPI engine adopts.
23:16	CS_HOLD	CS Hold Time The chip select hold time = (CS_HOLD_COUNT+1)*CLK_PERIOD.
15:8	SCK_LOW	SCK low Time The SCK clock low time = (SCK_LOW_COUNT+1)*CLK_PERIOD.
7:0	SCK_HIGH	SCK high Time The SCK clock high time = (SCK_HIGH_COUNT+1)*CLK_PERIOD.

SPIx+0x004 CFG1 SPI configuration Register 1

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name		WKUEN		CONT_CS		MODFEN	CSOE		FRMSIZE				RMSBF	TMSBF	CPHA	CPOL
Type		R/W		R/W		R/W	R/W		R/W				R/W	R/W	R/W	R/W
Reset		0		0		0	1		0	1	1	1	1	1	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	DMARXEN	DMATXEN	MODFIE	MSTR	RXOE	TXUIE	RXFIE	TXEIE	CS_IDLE							
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

Bit	Name	Description
31		
30	WKUEN	Slave Wakeup Function enable 1: slave wake up function enable 0: slave wake up function disable
29		
28	CONT_CS	CS Continuous Output Enable 1: CS output continuous 0: CS output non-continuous
27		
26	MODFEN	Master Mode Fault Detect Enable 1: enable the multi-master detect function 0: disable the multi-master detect function
25	CSEO	CS Hardware Output Enable 1: enable the CS hardware output 0: disable the CS hardware output
24		
23:20	FRMSIZE	Frame Size 1111: 16bit 1110: 15bit ... 0011: 4bit 0010: 4bit 0001: 4bit 0000: 4bit
19	RMSBF	RX MSB First 1: the first bit of shifter shift in is the MSB of the input data 0: the first bit of shifter shift in is the LSB of the input data
18	MSBF	TX MSB First 1: TX MSB first(MSB first shift out) 0: TX LSB first(LSB first shift out)
17	CPHA	Clock Phase 1: the first SCK transition edge is the data capture edge 0: the first SCK transition edge is the data shift out edge
16	CPOL	Clock Polarity 1: SCK is 1 when idle 0: SCK is 0 when idle
15	DMARXEN	DMA RX Request Enable 1: enable the DMA RX request 0: disable
14	DMATXEN	DMA TX channel enable

Bit	Name	Description
		1: enable the DMA RX request 0:disable
13	MODFIE	Mode Fault Interrupt Enable 1: enable 0: disable
12	MSTR	Master or Slave Mode Selection 1: master mode 0: slave mode
11	RXOIE	RX Buffer Overflow Interrupt Enable 1: enable 0: disable
10	TXUIE	TX Buffer Underflow Interrupt Enable 1: enable 0: disable
9	RXFIE	RX Buffer Full Interrupt Enable 1: enable 0: disable Note: RX buffers Not Empty will generate an interrupt.
8	TXEIE	TX Buffer Empty Interrupt Enable 1: enable 0: disable Note: TX buffers Not Full will generate an interrupt.
7:0	CS_IDLE	CS Idle Time CS IDLE time = (CS_IDLEA_COUNT+1)*CLK_PERIOD

SPIx+0x008 CMD SPI Command Register

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name										RO TRIG	CS RLS	SW RST				SPI EN
Type										R/W	R/W	R/W				R/W
Reset										0	0	0				0

Bit	Name	Description
31:6		
6	ROTRIG	Master RX only mode trig Note: write this bit '1' will trigger a sequence of read, while ROEN=1 in CFG2. Read this bit will return '0' always.
5	CSRLS	CS Release 1: release CS 0: no effect Software write this bit '1', then CS will go to high. Read this bit always return "0". This bit valid for CS continuous output (CONT_CS=1, CSOE=1).
4	SWRST	Software reset 1: reset 0: no effect Note: this reset only reset master engine/buffer/flag logic, slave buffer/flag logic. CFG0/CFG1/CFG2/CMD control bits will not reset.
0	SPIEN	SPI Enable 1: enable 0: disable

SPIx+0x00C STATUS SPI Status Register

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name								IDLEF	MEBY			MODEF	RX OF	TX UF	RX FF	TX EF
Type								R/W	R/W			R/W	R/W	R/W	R/W	R/W
Reset								1	0			0	0	0	0	1

Bit	Name	Description
31:8		
8	IDLEF	SPI IDLE Flag 1: SPI module hardware IDLE 0: SPI module hardware not IDLE Note: for master, TX buffer empty, RX buffer empty, and internal hardware idle, this bit can be '1'. For slave, TX buffer empty, RX buffer empty, and CS deassert(not be selected) , this bit can be '1'.
7	MEBY	SPI Master Engine Busy Flag 1: SPI master hardware's one byte transmission not finish 0: SPI master hardware's one byte transmission finish
4	MODEF	Mode Fault Error Flag 1: master mode error detected 0: no multi-master error detected When SPI configured as a master, it will detect the CS line state before drive CS low. If CS have been low, indicating another master have initial a transmission. CS pin act as a multi-master error detect input only when CSOE=1,MODEEN=1.
3	RXOF	RX Buffer Overflow Flag 1: RX buffer overflow 0: no overflow write "1" clear this bit. Note: if overflow, then only 1 byte valid data in DATA register can read out.
2	TXUF	TX Buffer Underflow Flag 1: TX buffer underflow 0: not underflow Note: write "1" clear this bit.
1	RXFF	RX Buffer Full Flag 1: full 0: not full Note: as long as rx buffers have valid data received(1 byte or 2 bytes), this bit will be '1'. So this bit '1' does not indicate that 2 bytes data received in rx buffers. Read DATA register will clear this bit hardware auto. Note: this bit named RX Buffers Not Empty is more reasonable maybe.
0	TXFF	TX Buffer Not Full Flag 1: empty 0: not empty Note: as long as tx buffers not full(2 bytes data), this bit will be '1'. Write DATA register will clear this bit hardware auto. Note: this bit named TX buffers Not Full is more reasonable maybe.

SPIx+0x010 DATA SPI DATA Register

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	DATA															
Type	R/W															
Reset	0															
Bit	Name		Description													
31:16																
15:0		DATA	SPI Data Port Register read: read will return DATA received write: write the DATA to be send													

SPIx+0x014 CFG2 SPI CFG2 Register

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Name																
Type																
Reset																
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name													ROEN	TOEN	MNDC	RSV
Type													R/W	R/W	R/W	R/W
Reset													0	0	0	0

Bit	Name		Description
31:4			
3	ROEN		RX Only Mode Enable 1: enable 0: disable Note: RX only mode, TXEF will keep 0. Not trigger RXEF IRQ even TXEIE=1.
2	TOEN		TX Only Mode Enable 1: enable 0: disable Note: TX only mode, RXFF not set after one byte transfer finished. Not trigger RXFF IRQ even RXFIE=1.
1	MNOV		Master No Overflow Mode 1: enable 0: disable If rxbuff is full, then write to txbuff will not trigger new send.
0	RSV		Reserved Note: not change this bit.

20 DMA

20.1 Introduction

The Direct Memory Access controller (DMA) is used to speed up memory transmission between peripherals-to-memory or memory-to-memory. Without CPU operations, DMA can move data fast and improve microcontroller's performance.

There are 12 dedicated channels for different kinds peripherals, such as UART, I2C, SPI, ADC, etc. It has a round-robin arbiter for handling priorities between each channel.

20.2 Features

- 12 independently configurable channels (requests).
- Each of the 12 channels is connected to dedicated hardware DMA requests, software trigger is also supported on each channel (memory to memory). This configuration is done by software.
- Priorities between requests from channels of one DMA are software programmable (4 levels consisting of very high, high, medium, low) or hardware in case of equality (channel 0 has the highest priority and channel 15 has the lowest priority, etc.)
- Independent source and destination transfer size (byte, half word, word). Source/destination addresses must be aligned on the data size.
- Support for circular buffer management.
- 3 event flags (DMA Half Transfer, DMA Transfer complete and DMA Transfer Error) logically ORed together in a single interrupt request for each channel.
- Memory-to-memory transfer.
- Peripheral-to-memory and memory-to-peripheral transfers.
- Access to SRAM, APB peripherals as source and destination.
- Programmable number of data to be transferred: up to 32767.

20.3 Block diagram

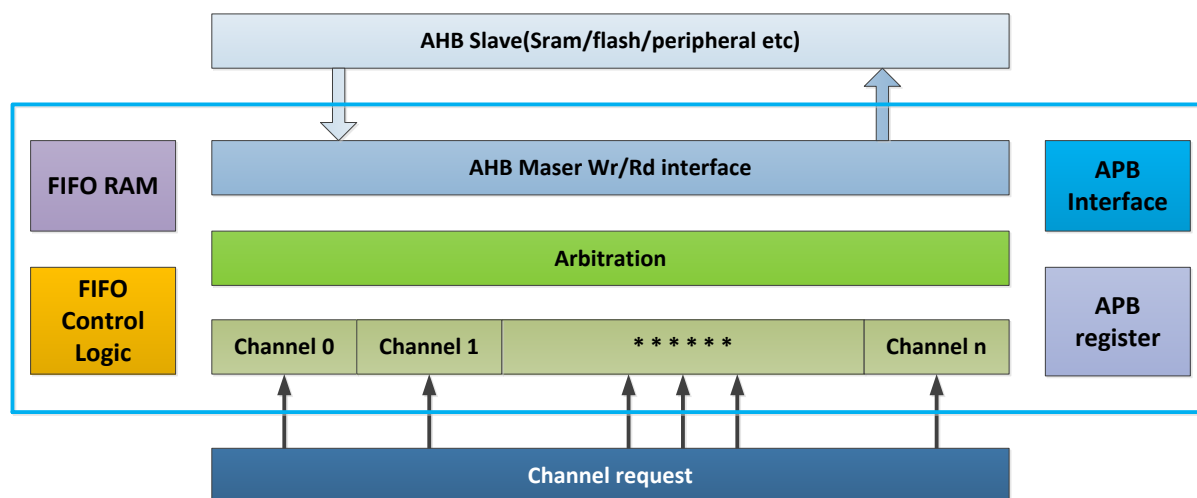


Figure 20-1 DMA block diagram

20.4 Mode of operations

The DMA module does not support in either Stop or Standby mode. Before entering either Stop or Standby mode, all DMA channels must be disabled, that is, the `CHAN_ENABLE` bit of the `CHANNELx_CHAN_ENABLE` register configured to 0 for all DMA channels.

20.5 Function description

20.5.1 DMA request mapping

Table 20-1 DMA request mapping

Peripherals	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5	Channel 6
ADC	ADC					
SPI		SPI1_RX	SPI1_TX	SPI2_RX	SPI12_TX	
UART	UART6_RX	UART3_TX	UART3_RX	UART1_TX	UART1_RX	UART2_TX
I2C						
Peripherals	Channel 7	Channel 8	Channel 9	Channel 10	Channel 11	Channel 12
ADC						
SPI						
UART	UART2_RX	UART4_TX	UART4_RX	UART5_TX	UART5_RX	UART6_TX
I2C			I2C2_TX	I2C2_RX	I2C1_TX	I2C1_RX

20.5.2 DMA arbiter

The arbiter manages the channel requests based on their priority and launches the peripheral/memory access sequences.

The priorities are managed in two stages:

- Software: each channel priority can be configured in the `CHANNELx_CONFIG` register. There are four levels:
 - Very high priority.

- High priority.
 - Medium priority.
 - Low priority.
- When software priorities are equal, priority is determined based on the hardware channel number. For example, channel 0 has the highest priority, channel 15 has the lowest priority, and so on.

20.5.3 Configuration guide

ALL registers except **CHANNELx_CHAN_ENABLE** register should be programmed before user configures **CHANNELx_CHAN_ENABLE** register. Because parameters take effect after **CHAN_ENABLE**'s pose edge. And when **CHAN_ENABLE** is High, please don't modify parameters.

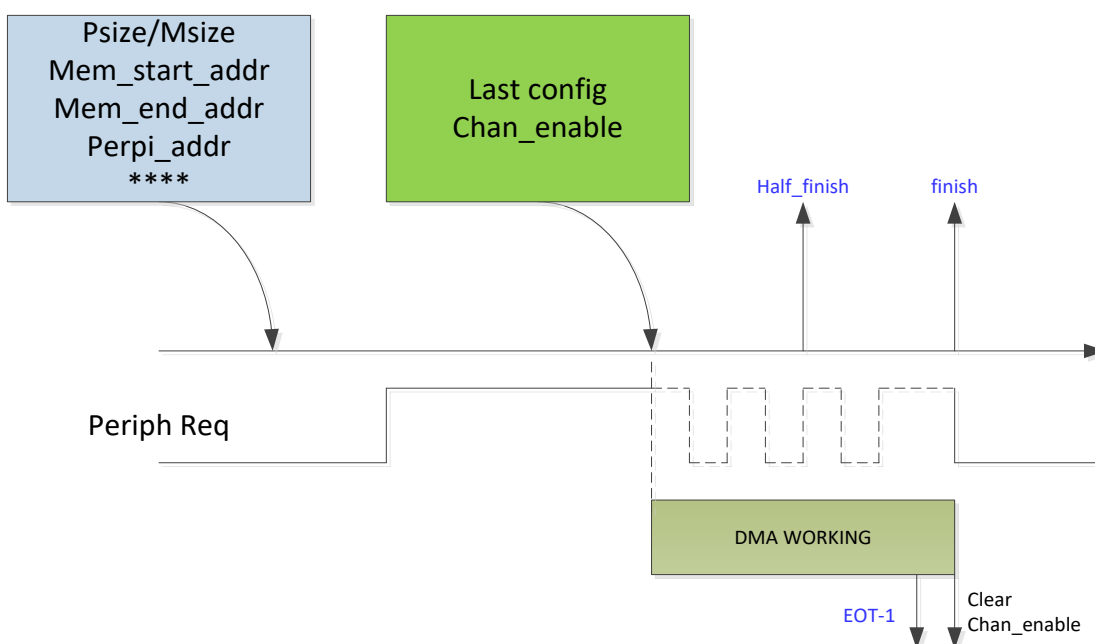


Figure 20-2 DMA configuration guide

20.5.3.1 Soft reset and hard reset

Soft reset and hard reset exist in DMA global control and each DMA engine. When soft reset is set, the engine will be reset after the current transaction is finished, and therefore the soft reset will not cause any bus hang. Conversely, when hard reset is set, the engine will be reset immediately, and therefore the bus may go down due to unfinished (and will never finish) transaction.

The mechanism of global soft reset is described as follows. When the software would like to re-start all engines or re-clear all engines in DMA, it can set the global soft resetter to 1. Then HW set **WARM_RST** back to 0 to finish the global soft reset.

The mechanism of global hard reset is described as follows. When the software would like to re-start all engines or re-clear all engines in DMA without waiting for any period of time, it can set the global **HARD_RST** to 1 and then set to 0 to complete the global hard reset. Note that this may break the bus protocol and result in system hang.

20.5.3.2 Pause and Resume

There are pause and resume functions available for DMA. The mechanism is as follows:

1. Start DMA. (Program necessary settings, then set `CHAN_ENABLE = 1`.)
2. Pause DMA. (Set `STOP = 1`.)
3. Resume DMA. (Set `STOP = 0`.)
4. Wait for DMA to finish. (`CHAN_ENABLE` will become 0, and interrupt flag will be set to 1.)

The software can repeat step 2 and 3 many times when DMA is running. DMA will not pause immediately and will wait for the last transaction to be finished.

20.5.3.3 Channel circular

Circular mode is available to handle circular buffers and continuous data flows (e.g. ADC scan mode). This feature can be enabled using the `CHAN_CIRCULAR` bit in the **CHANNELx_CONFIG** register. When circular mode is activated, the number of data to be transferred is automatically reloaded with the initial value programmed during the channel configuration phase, and the DMA requests continue to be served.

It should be noted that after read or write `MEM_END_ADDR-0x01`, the DMA master will return to `MEM_START_ADDR` to continue handling.

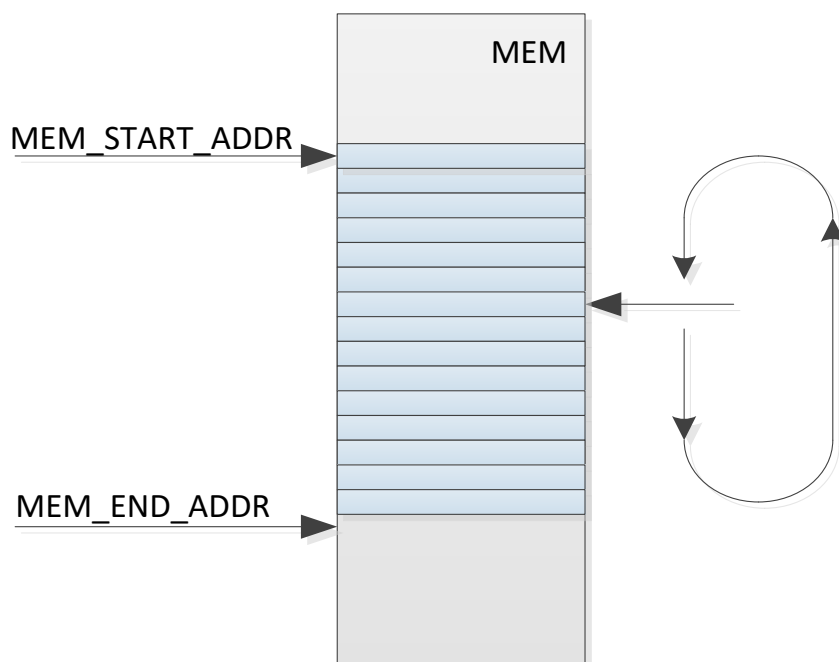


Figure 20-3 DMA channel circular

20.5.3.4 I2C using DMA

When the number of data transfers which has been programmed for the corresponding DMA stream is reached, the DMA controller sends an End of Transfer EOT-1 signal to the I2C interface. I2C hard engine has received the signal and decide whether to send ACK/NACK.

20.5.3.5 Programmable data width, data alignment

Table 20-2 Programmable data width & data alignment

MSIZE	PSIZE	Length	MEM_BYTE_MODE	Source	Direction	Destination
32	32	4	00	0x00 03020100 0x04 07060504 0x08 0B0A0908 0x0C 0F0E0D0C	1(TX)	0x00 03020100 0x04 07060504 0x08 0B0A0908 0x0C 0F0E0D0C
32	32	4	01	0x00 03020100 0x04 07060504	1(TX)	0x00 00000100 0x04 00000302 0x08 00000504 0x0C 00000706
32	32	4	10	0x00 03020100 0x04 07060504	1(TX)	0x00 00000200 0x04 00000301 0x08 00000604 0x0C 00000705
32	32	4	11	0x00 03020100	1(TX)	0x00 00000000 0x04 00000001 0x08 00000002 0x0C 00000003
32	32	4	00	0x00 03020100 0x04 07060504 0x08 0B0A0908 0x0C 0F0E0D0C	0(RX)	0x00 03020100 0x04 07060504 0x08 0B0A0908 0x0C 0F0E0D0C
32	32	4	01	0x00 03020100 0x04 07060504 0x08 0B0A0908 0x0C 0F0E0D0C	0(RX)	0x00 05040100 0x04 0D0C0908
32	32	4	10	0x00 03020100 0x04 07060504 0x08 0B0A0908 0x0C 0F0E0D0C	0(RX)	0x00 05010400 0x04 0D090C08
32	32	4	11	0x00 03020100 0x04 07060504 0x08 0B0A0908 0x0C 0F0E0D0C	0(RX)	0x00 0c080400

20.6 Memory map and register descriptions

Table 20-3 DMA register map

DMA Base address: 0x40012000

ADDRESS	TITLE	DESCRIPTION
DMA+0x00	DMA_TOP_RST	General DMA reset
DMA+0x40	CHANNELx_STATUS	flag
DMA+0x44	CHANNELx_INTEN	Interrupt enable
DMA+0x48	CHANNELx_RST	Channel reset
DMA+0x4C	CHANNELx_STOP	DMA channel stop
DMA+0x50	CHANNELx_CONFIG	DMA channel config

ADDRESS	TITLE	DESCRIPTION
DMA+0x54	CHANNELx_CHAN_LENGTH	DMA channel length
DMA+0x58	CHANNELx_MEM_START_ADDR	Memory start address
DMA+0x5C	CHANNELx_MEM_END_ADDR	Memory end address
DMA+0x60	CHANNELx_PERIPH_ADDR	Channel peripheral address
DMA+0x64	CHANNELx_CHAN_ENABLE	Channel enable
DMA+0x68	CHANNELx_DATA_TRANS_NUM	Data transfer number
DMA+0x6C	CHANNELx_INTER_FIFO_DATA_LEFT_NUM	data left in inter fifo

DMA_TOP_RST register

OFFSET ADDR = 12'h00

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

Bit(s)	Name	Description
1	HARD_RST	General DMA hard reset (reset regardless of the current transaction)
		SW sets ' HARD_RST ' to 1 then sets ' HARD_RST ' back to 0, and the reset mechanism is finished.
		0: Disable
		1: Enable
0	WARM_RST	General DMA soft reset (reset after the current transaction)
		SW sets ' WARM_RST ' to 1 and HW clears ' WARM_RST ' back to 0, and the reset mechanism is finished.
		0: Disable
		1: Enable

CHANNELx_STATUS register

OFFSET ADDR = 12'h40

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0												TRANS_ERROR		HALF_FINISH	
W													W0C		W0C	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit(s)	Name	Description
2	TRANS_ERROR	Transfer Error Flag Indicate when channel n reading or writing, whether or not error flag occurred .write "0"to this bit, clear status. 0: Error not happened 1: Error happened
1	HALF_FINISH	Half Finish Flag Indicate when channel n reading or writing, whether or not half Number of Data transferred. write "0"to this bit, clear status. 0: Half of data not finished 1: Half of data finished
0	FINISH	Finish Flag Indicate when channel n reading or writing, whether or not DMA channel finished data transfer. write "0"to this bit, clear status. 0: Data transfer not finished 1: Data transfer finished

CHANNELx_INTEN register

OFFSET ADDR = 12'h44

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0												TRANS_ERROR interrupt enable		HALF_FINISH interrupt enable	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit(s)	Name	Description
2	TRANS_ERROR interrupt enable	TRANS_ERROR interrupt enable
		This bit is set and cleared by software. 0: interrupt disabled 1: interrupt enabled
1	HALF_FINISH interrupt enable	HALF_FINISH interrupt enable
		This bit is set and cleared by software. 0: interrupt disabled 1: interrupt enabled
0	FINISH interrupt enable	FINISH interrupt enable
		This bit is set and cleared by software. 0: interrupt disabled 1: interrupt enabled

CHANNELx_RST register

OFFSET ADDR = 12'h48

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	FLUSH	HARD_RST	WARM_RST
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit(s)	Name	Description
2	FLUSH	DMA channel flush
		Setting FLUSH = 1 will stop DMA and allow DMA to flush its internal buffer residual data to the memory. After the flush is finished, DMA will set channel enable to 0 and stop DMA. SW sets FLUSH = 1 and waits for HW to be clear to 0. 0: Disable 1: Enable
1	HARD_RST	DMA channel hard reset (reset regardless of the current transaction)
		SW sets 'HARD_RST' to 1 then sets 'HARD_RST' back to 0, and the reset mechanism is finished. 0: Disable 1: Enable
0	WARM_RST	DMA channel soft reset (reset after the current transaction)
		SW sets 'WARM_RST' to 1 then HW sets 'WARM_RST' back to 0, and the reset mechanism is finished. 0: Disable 1: Enable

CHANNELx_STOP register

OFFSET ADDR = 12'h4C

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

Bit(s)	Name	Description
0	STOP	DMA channel stop (stop after the current transaction) SW sets ' STOP ' to 1 then after current transaction, transfer paused, SW sets ' STOP ' to 0 then resume data transfer. 0: Not stop channel transfer 1: STOP channel transfer

CHANNELx_CONFIG register

OFFSET ADDR = 12'h50

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	MEM_BYTE_MODE		CH_DIR	CHAN_CIRCULAR	PERIPH_INCREMENT	MEM_INCREMENT	PERIPH_SIZE		MEM_SIZE		CHAN_PRIORITY		MEM_2MEM
W	0	0	0													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit(s)	Name	Description
12:11	MEM_BYTE_MODE	Indicate MEM word split transfer number 00: 1 01: 2 10: 2 11: 4
10	CHAN_DIR	Indicate Data transfer direction 0: read from Peripheral

Bit(s)	Name	Description
		1: read from MEM
9	CHAN_CIRCULAR	0: Circular mode disable 1: Circular mode enable
8	PERIPH_INCREMENT	0: Peripheral address fixed 1: Peripheral address increment
7	MEM_INCREMENT	0: MEM address fixed 1: MEM address increment
6:5	PERIPH_SIZE	00: 8bit 01: 16bit 10: 32bit 11: reserve
4:3	MEM_SIZE	00: 8bit 01: 16bit 10: 32bit 11: reserve
2:1	CHAN_PRIORITY	00: low 01: middle 10: high 11: very high
0	MEM2MEM	0: transfer between non MEM and MEM 1: transfer between MEM and MEM

CHANNELx_CHAN_LENGTH register **OFFSET ADDR = 12'h54**

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

Bit(s)	Name	Description
15:0	CHAN_LENGTH	DMA channel Transfer length 0~32767 [15]bit should be 0

CHANNELx_MEM_START_ADDR register OFFSET ADDR = 12'h58

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

Bit(s)	Name	Description
31:0	MEM_START_ADDR R	MEM_START_ADDR Memory initial address.

CHANNELx_MEM_END_ADDR register OFFSET ADDR = 12'h5C

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

Bit(s)	Name	Description
31:0	MEM_END_ADDR	MEM_END_ADDR After read or write MEM_END_ADDR -1, the DMA master returns to MEM_START_ADDR address to continue handling.

CHANNELx_PERIPH_ADDR register OFFSET ADDR = 12'h60

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

Bit(s)	Name	Description
31:0	PERIPH_ADDR	PERIPH_ADDR

CHANNELx_CHAN_ENABLE register

OFFSET ADDR = 12'h64

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

Bit(s)	Name	Description
0	CHAN_ENABLE	<p>Channel enable</p> <p>In non-channel circular mode, SW set CHAN_ENABLE to 1, and when transfer finish, HW clear CHAN_ENABLE to 0;</p> <p>In channel circular mode, SW set CHAN_ENABLE to 1, when transfer finish, HW restart new transfer with the same configuration; If you need to disable the DMA channel, you must disable the DMA channel circular mode first.</p> <p>0: disable channel 1: enable channel</p>

CHANNELx_DATA_TRANS_NUM register

OFFSET ADDR = 12'h68

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

Bit(s)	Name	Description
15:0	DATA_TRANS_NUM	<p>DATA_TRANS_NUM</p> <p>Indicated how many data have been transferred</p>

CHANNELx_INTER_FIFO_DATA_LEFT_NUM register OFFSET ADDR = 12'h6C

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

Bit(s)	Name	Description
INTER_FIFO_DATA_LEFT_NUM		
5:0	INTER_FIFO_DATA_LEFT_NUM	Indicated how many byte data left in inter fifo. usually used it together with FLUSH. When Time out, and INTER_FIFO_DATA_LEFT_NUM not equal to 0,then SW can start flush process.

21 WDT

21.1 Introduction

The Watchdog Timer (WDOG) module is an independent timer that is available for system use. It provides a safety feature to ensure that software is executing as planned and that the CPU is not stuck in an infinite loop or executing unintended code. If the WDOG module is not serviced (refreshed) within a certain period, it resets the MCU.

21.2 Features

- Configurable clock source inputs independent from the bus clock.
 - Internal 32 kHz RC oscillator.
 - Internal 8 MHz RC oscillator.
 - External XOSC clock source.
- Programmable timeout period.
 - Programmable 32-bit timeout value.
 - Optional fixed 256 clock prescaler when longer timeout periods are needed.
- Robust write sequence for counter refresh
 - Refresh sequence of writing 0x02A602A6 and then 0x80B480B4.
- Window mode option for the refresh mechanism
 - Programmable 32-bit window value.
 - Provides robust check that program flow is faster than expected.
 - Early refresh attempts trigger a reset.
- Optional timeout interrupt to allow post-processing diagnostics
 - Interrupt request to CPU with interrupt vector for an interrupt service routine.
 - Forced reset occurs 128 bus clocks after the interrupt vector fetch.
- Configuration bits are write-once-after-reset to ensure watchdog configuration cannot be mistakenly altered.
- Robust write sequence for unlocking write-once configuration bits.
 - Unlock sequence of writing 0x20C520C5 and then 0x28D928D9 for allowing updates to write-once configuration bits.
 - Software must make updates within 128 bus clocks after unlocking and before WDOG closing unlock window.

21.3 Block diagram

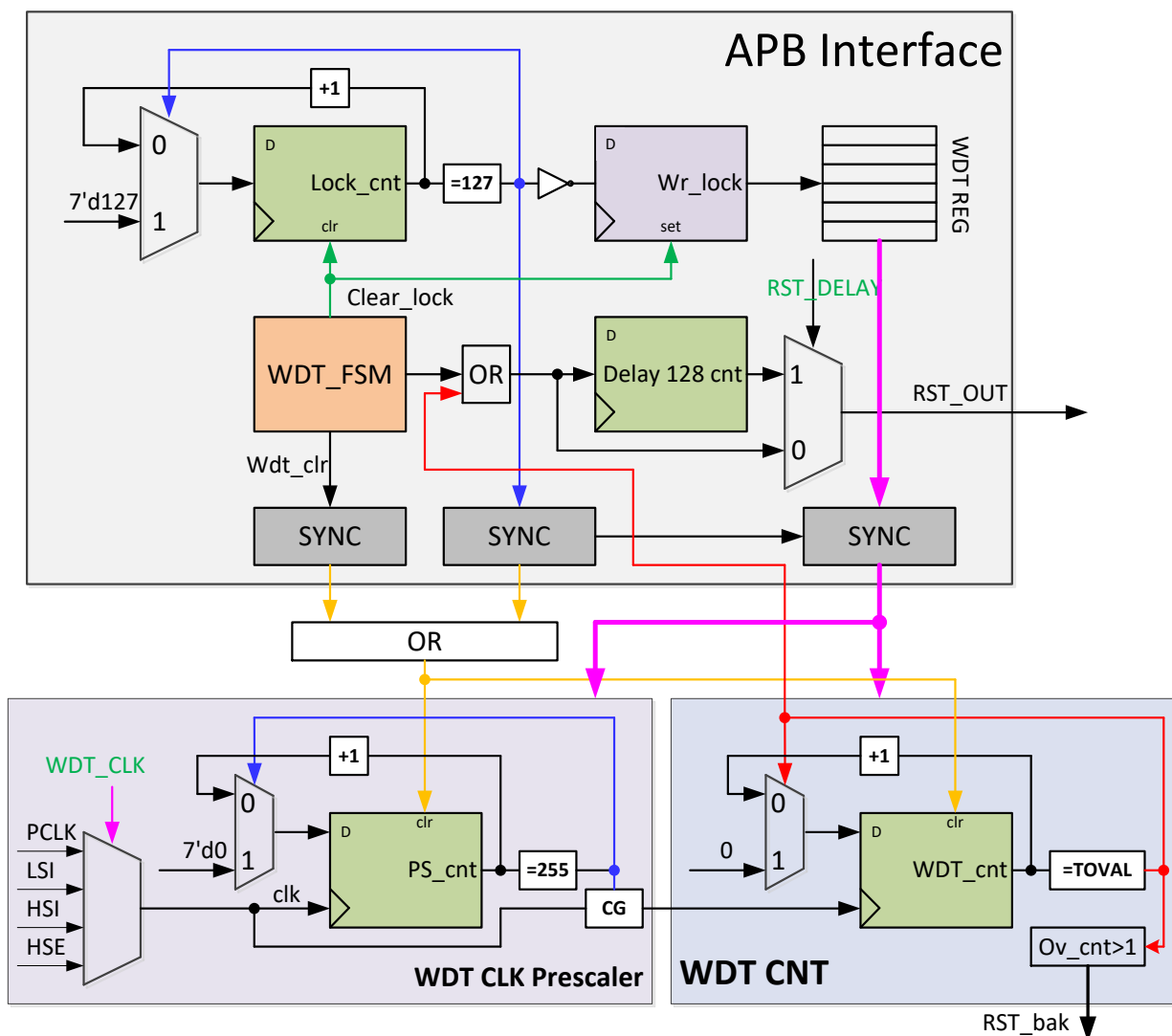


Figure 21-1 WDG block diagram

21.4 Mode of operation

WDG is always running with Internal 32 kHz RC oscillator even if microcontroller has entered STOP mode. Therefore, the user should make sure disabled WDG if doesn't feed WDG during STOP mode to avoid unexpected system reset.

21.5 Memory map and register descriptions

Table 21-1 WDG register map

WDG Base address: 0x4000b000

ADDRESS	TITLE	DESCRIPTION
WDG+0x00	WDT_CS1	Watchdog Status

ADDRESS	TITLE	DESCRIPTION
WDG+0x04	WDT_CS2	Watchdog Status
WDG+0x08	WDT_CNT	Watchdog Counter
WDG+0x0C	WDT_TOVAL	Watchdog Timeout Value
WDG+0x10	WDT_WIN	Watchdog Window

WDT_CS1 register

OFFSET ADDR = 12'h00

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								EN	INT	UPDATE	0			0	
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

Bit(s)	Name	Description
Watchdog Enable		
7	EN	This write-once bit enables the watchdog counter to start counting. 0 Watchdog disabled. 1 Watchdog enabled.
Watchdog Interrupt		
6	INT	This write-once bit configures the watchdog to generate an interrupt request upon a reset-triggering event (timeout or illegal write to the watchdog), prior to forcing a reset. After the interrupt vector fetch, the reset occurs after a delay of 128 bus clocks. 0 Watchdog interrupts are disabled. Watchdog resets are not delayed. 1 Watchdog interrupts are enabled. Watchdog resets are delayed by 128 bus clocks.
Allow updates		
5	UPDATE	This write-once bit allows software to reconfigure the watchdog without a reset. 0 Updates not allowed. After the initial configuration, the watchdog cannot be later modified without forcing a reset. 1 Updates allowed. Software can modify the watchdog configuration registers within 128 bus clocks after performing the unlock write sequence.

WDT_CS2 register

OFFSET ADDR = 12'h04

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								WIN	FLG	0	PRES	0		CLK	
W										w1c						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit(s)	Name	Description
7	WIN	Watchdog Window
		This write-once bit enables window mode.
		0 Window mode disabled.
		1 Window mode enabled.
6	FLG	Watchdog Interrupt Flag
		This bit is an interrupt indicator when INT is set in control and status register 1. Write 1 to clear it.
		0 No interrupt occurred.
		1 An interrupt occurred.
4	PRES	Watchdog Prescaler
		This write-once bit enables a fixed 256 pre-scaling of watchdog counter reference clock. (The block diagram shows this clock divider option.)
		0 256 prescaler disabled.
		1 256 prescaler enabled.
1:0	CLK	Watchdog Clock
		This write-once field indicates the clock source that feeds the watchdog counter.
		00 Bus clock.
		01 32 kHz internal low-power oscillator (LPOSC).
		10 8 MHz internal oscillator (LFOSC).
		11 External clock source(XOSC).

WDT_CNT register

OFFSET ADDR = 12'h08

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

Bit(s)	Name	Description
Watchdog Counter Register		
31:0	WDT_CNT [31:0]	The watchdog counter registers provide access to the value of the free running watchdog counter. Software can read the counter registers at any time. Software cannot write directly to the watchdog counter; however, two write sequences to these registers have special functions: The refresh sequence and The unlock sequence

WDT_TOVAL register
OFFSET ADDR = 12'h0C

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

Bit(s)	Name	Description
Watchdog Timeout Value Register		
31:0	WDT_TOVAL [31:0]	The watchdog counter is continuously compared with the timeout value . If the counter reaches the timeout value, the watchdog forces a reset. Counter length equivalent to WDT_TOVAL+1 Default value is 0x3f0000

WDT_WIN register
OFFSET ADDR = 12'h10

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

Bit(s)	Name	Description
Watchdog Window Register		
31:0	WDT_WIN [31:0]	This section describes the watchdog window registers: When window mode is enabled (WDOG_CS2[WIN] is set),WDOG_WIN determine the earliest time that a refresh sequence is considered valid.

21.6 Functional description

21.6.1 Watchdog refresh mechanism

The watchdog resets the MCU if the watchdog counter is not refreshed. A robust refresh mechanism makes it very unlikely that the watchdog can be refreshed by runaway code. To refresh the watchdog counter, software must execute a refresh write sequence before the timeout period expires. In addition, if window mode is used, software must not start the refresh sequence until after the time value set in the WDOG_WIN registers.

When Window mode is enabled, the watchdog must be refreshed after the counter has reached a minimum expected time value. Otherwise, the watchdog resets the MCU. The minimum expected time value is specified in the WDOG_WIN registers. Setting CS2[WIN] enables Window mode.

The refresh write sequence is a write of 0x02A602A6 followed by a write of 0x80B480B4 to the WDOG_CNT registers. The write of the 0x80B480B4 must occur after the write of 0x02A602A6. Otherwise, the watchdog resets the MCU.

Note: $\text{timeout} = (\text{PRES} * (\text{WDT_CNT} + 1)) / \text{CLK}$

21.6.2 Configuring the Watchdog

All watchdog control bits, timeout value, and window value are write-once after reset. This means that after a write has occurred they cannot be changed unless a reset occurs. This provides a robust mechanism to configure the watchdog and ensure that a runaway condition cannot mistakenly disable or modify the watchdog configuration after configured.

This is guaranteed by the user configuring the window and timeout value first, followed by the other control bits, and ensuring that CS1[UPDATE] is also set to 0. The new configuration takes effect only after all registers except WDOG_CNT are written once after reset. Otherwise, the WDOG uses the reset values by default. If window mode is not used (CS2[WIN] is 0), writing to WDOG_WIN is not required to make the new configuration take effect.

In some cases (such as when supporting a bootloader function), users may want to reconfigure or disable the watchdog without forcing a reset first. By setting CS1[UPDATE] to 1 on the initial configuration of the watchdog after a reset, users can reconfigure the watchdog at any time by executing an unlock sequence. (Conversely, if CS1[UPDATE] remains 0, the only way to reconfigure the watchdog is by initiating a reset.) The unlock sequence is similar to the refresh sequence but uses different values.

The unlock sequence is a write to the WDOG_CNT registers of 0x20C520C5 followed by 0x28D928D9 at any time after the watchdog has been configured. After completing the unlock sequence, the user must reconfigure the watchdog within 128 bus clocks.

21.6.3 Using interrupts to delay watchdog reset

When interrupts are enabled (**WDT_CS1 reister**[INT] = 1), the watchdog first generates an interrupt request upon a reset triggering event (such as a counter timeout or invalid refresh attempt). The watchdog delays forcing a reset for 128 bus clocks to allow the interrupt service routine (ISR) to perform tasks, such as analyzing the stack to debug code. When interrupts are disabled (**WDT_CS1 reister** [INT] = 0), the watchdog does not delay forcing a reset.

21.6.4 Backup reset

The LPOSC clock must be used as the reference clock for the counter. Otherwise, the backup reset function is not available. The backup reset function is a safeguard feature that independently generates a reset in case the main WDOG logic loses its clock (the bus clock) and can no longer monitor the counter. If the watchdog counter overflows twice in succession (without an intervening reset), the backup reset function takes effect and generates a reset.

22 RTC

22.1 Introduction

The Real-Time Counter (RTC) consists of one 32-bit counter, one 32-bit comparator, several binary-based and decimal-based prescaler dividers, four clock sources, one programmable periodic interrupt, and one programmable external toggle pulse output. This module can be used for time-of-day, calendar or any task scheduling functions. It can also serve as a cyclic wake-up from low-power modes, Stop and Wait without the need of external components.

22.2 Features

Features of the RTC module include:

- 32-bit up-counter
 - 32-bit modulo match limit.
 - Software controllable periodic interrupt on match.
- Software selectable clock sources for input to prescaler with programmable 20 bit prescaler.
 - Bus clock
 - LPOSC (32 kHz)
 - External XOSC clock
 - Internal LFOSC clock (8 MHz)
- Tamper detection
 - 1 tamper event on edge detection.
 - 2 backup registers (8 bytes) .

22.3 Block diagram

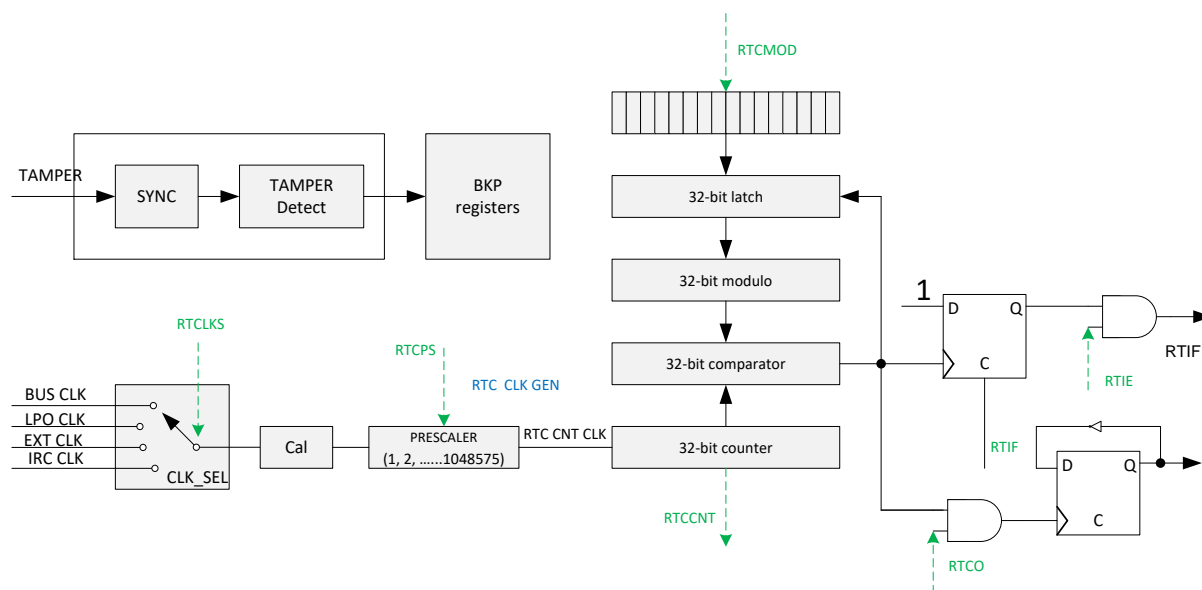


Figure 22-1 RTC block diagram

22.4 Mode of operation

RTC is running even if microcontroller has entered STOP mode when using LPOSC(32 kHz) as clock source. If user enables RTC interrupt, when RTC times out, it will trigger an interrupt and wake up MCU from STOP mode.

22.5 Memory map and register descriptions

Table 22-1 RTC register map

RTC Base address: 0x40008400

ADDRESS	TITLE	DESCRIPTION
RTC+0x00	RTC_SC	Watchdog Status
RTC+0x04	RTC_MOD	RTC Modulo
RTC+0x08	RTC_Counter	RTC Count
RTC+0x0C	RTC Clock Prescaler	RTC Clock Prescaler
RTC+0x10	RTC_Prescaler Counter	RTC Prescaler Counter
RTC+0x20	BKP_CR	TAMPER pin active level/enable
RTC+0x24	BKP_CSR	TAMPER interrupt
RTC+0x28,0x2c	BKP_DRx	Backup Data

RTC_SC register

OFFSET ADDR = 12'h00

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0												Pcl k_e clk_ sel	CL K_C HG _O K	RPI F	RPI E
W																
Res et	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RTCLKS		0		0				RTIF	RTIE	0	RTCO	0			
W																
Res et	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit(s)	Name	Description
18	CLK_CHG_OK	CLK_CHG_OK: Clock change Ok flag This status bit indicates the RTC counter clock change state and read only. 0: RTC counter clock has changed success. 1: RTC counter clock is changing.
17	RPIF	RPIF: Real-Time Prescaler Interrupt Flag This status bit indicates the RTC Prescaler counter register reached the value in the RTC prescaler divider register. Writing a logic 0 has no effect. Writing a logic 1 clears the bit and the real-time interrupt request. Reset clears RPIF to 0. 0: RTC Prescaler counter has not reached the value in the RTC prescaler divider register. 1: RTC Prescaler counter has reached the value in the RTC prescaler divider register.
16	RPIE	RPIE: Real-Time Prescaler Interrupt Enable This read/write bit enables real-time prescaler interrupts. If RPIE is set, then an interrupt is generated when RPIF is set. Reset clears RPIE to 0. 0: Real-time prescaler interrupt requests are disabled. Use software polling. 1: Real-time prescaler interrupt requests are enabled.
15:14	RTCLKS	RTCLKS: Real-Time Clock Source Select This read/write field selects the clock source input to the RTC prescaler. Changing the clock source clears the prescaler and RTCCNT counters. Reset clears RTCLKS to 00. $\text{Freq} = \text{RTCLKS} / ((\text{MOD} + 1) * (\text{RTCP} + 1))$ 00: Bus clock. 01: Internal 32 kHz oscillator (LPOCLK). 10: External oscillator (XOSC). 11: Internal 8 MHz oscillator (LFOSC).
7	RTIF	RTIF: Real-Time Interrupt Flag This status bit indicates the RTC counter register reached the value in the

Bit(s)	Name	Description
		RTC modulo register. Writing a logic 0 has no effect. Writing a logic 1 clears the bit and the real-time interrupt request. Reset clears RTIF to 0. 0: RTC counter has not reached the value in the RTC modulo register. 1: RTC counter has reached the value in the RTC modulo register.
6	RTIE	RTIE: Real-Time Interrupt Enable This read/write bit enables real-time interrupts. If RTIE is set, then an interrupt is generated when RTIF is set. Reset clears RTIE to 0. 0: Real-time interrupt requests are disabled. Use software polling. 1: Real-time interrupt requests are enabled.
4	RTCO	RTCO: Real-Time Counter Output The read/write bit enables real-time to toggle output on pinout. If this bit is set, the RTCO pinout will be toggled when RTC counter overflows. 0: Real-time counter output disabled. 1: Real-time counter output enabled.

RTC_MOD register
OFFSET ADDR = 12'h04

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

Bit(s)	Name	Description
31:0	MOD[31:0]	RTC Modulo This read/write field contains the modulo value used to reset the count to 0x0 upon a compare match and set SC[RTIF] status field. In normal use, MOD should not be equal to 0x0. Reset sets the modulo to 0x0.

RTC_Counter register
OFFSET ADDR = 12'h08

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

Bit(s)	Name	Description
RTC Count		
31:0	CNT[31:0]	This read-only field contains the current value of the 32-bit counter. Writes have no effect to this register. Reset or writing different values to SC[RTCLKS] and [RTCPS] clear the count to 0x0.

RTC Clock Prescaler
OFFSET ADDR = 12'h0C

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

Bit(s)	Name	Description
RTCPS: Real-Time Clock Prescaler Select		
19:0	RTCPS	This read/write field selects binary-based or decimal-based divide-by values for the clock source. Changing the prescaler value clears the prescaler and RTCCNT counters. Reset clears RTCPS to 0000. When RTCPS equal 0000, RTC counter Off.

RTC_Prescaler Counter register
OFFSET ADDR = 12'h10

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

Bit(s)	Name	Description
RTC PSCount		
19:0	PSCNT[31:0]	This read-only field contains the current value of the 20-bit counter. Writes have no effect to this register. Reset or writing different values to SC[RTCLKS] and [RTCPS] clear the count to 0x0.

BKP_CR register
OFFSET ADDR = 12'h20

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

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

Bit(s)	Name	Description
1	TPAL	TPAL: TAMPER pin active level 0: A high level on the TAMPER pin resets all data backup registers (if TPE bit is set). 1: A low level on the TAMPER pin resets all data backup registers (if TPE bit is set).
0	TPE	TPE: TAMPER pin enable 0: The TAMPER pin is free for general purpose I/O 1: Tamper alternate I/O function is activated.

BKP_CSR register

OFFSET ADDR = 12'h24

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RESERVED															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RESERVED						TIF	TEF	RESERVED					TPIE	0	0
W															CTI	CTE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit(s)	Name	Description
9	TIF	TIF: Tamper interrupt flag This bit is set by hardware when a Tamper event is detected and the TPIE bit is set. It is cleared by writing 1 to the CTI bit (also clears the interrupt). It is also cleared if the TPIE bit is reset. 0: No Tamper interrupt 1: A Tamper interrupt occurred
8	TEF	TEF: Tamper event flag This bit is set by hardware when a Tamper event is detected. It is cleared by writing 1 to the CTE bit. 0: No Tamper event 1: A Tamper event occurred

Bit(s)	Name	Description
2	TPIE	TPIE: TAMPER pin interrupt enable 0: Tamper interrupt disabled 1: Tamper interrupt enabled (the TPE bit must also be set in the BKP_CR register)
1	CTI	CTI: Clear tamper interrupt This bit is write only, and is always read as 0. 0: No effect 1: Clear the Tamper interrupt and the TIF Tamper interrupt flag.
0	CTE	CTE: Clear tamper event This bit is write only, and is always read as 0. 0: No effect 1: Reset the TEF Tamper event flag (and the Tamper detector)

BKP_DRx register

OFFSET ADDR = 12'h28, 12'h2C

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

Bit(s)	Name	Description
31:0	BKP_DATA[31:0]	Backup data These bits can be written with user data. Note: The BKP_DRx registers are not reset by a System reset or Power reset or when the device wakes up from Standby mode. They are reset by a Backup Domain reset or by a TAMPER pin event (if the TAMPER pin function is activated).

22.6 Functional description

22.6.1 Low power mode operation

The RTC continues to run in low power mode if the RTC is enabled before executing the SLEEP instruction and clock source is LPOSC clock.

As RTC block can work in STOP mode even in Standby mode, therefore, the RTC can be used to bring the MCU out of stop modes with no external components.

22.6.2 RTC backup registers

The backup registers (RTC_BKPDRx) are two 32-bit registers in value line devices for storing 8 bytes of user application data. They are implemented in the VDD domain. They are not reset by system reset or when the device wakes up from Standby mode. They are reset by a power-on reset.

A tamper detection event resets all backup registers (RTC_BKPDx). By setting the TPIE bit in the **BKP_CSR register**, an interrupt is generated when a tamper detection event occurs.

22.6.3 Configuration sequence

All registers should be programmed before RTC_CLK & RTC_RTPTS. When RTC block is on normal working, any change of RTC_CLK & RTC_RTPTS will clear RTC counter.

23 Embedded Flash

23.1 Embedded flash function overview

23.1.1 Introduction

This chapter introduces the embedded flash controller, which acts as a bridge between the Cortex-M3 and the flash memory. In practical application, flash boot mode is the main mode and the read-only code will be stored in it. For this module, many application descriptions will be introduced.

23.1.2 Feature list

- Flash memory:
 - 64Kwords or 256Kbytes.
 - Endurance: ≥ 10000 cycles.
 - Memory organization: main memory(64K \times 32bit) + information memory(2560 \times 32bit).
 - Page capacity: 2048 bytes per page.
- Flash controller:
 - Operation list:
 - Erase: Page Erase, Mass Erase, Option Page Erase.
 - Program: Option Page Program, Page Program, the minimum programming bit width is 32bit, and the programming address needs to be aligned with 4 bytes.
 - Read: Read.
 - Verify: Page Erase Verify, Mass Erase Verify.
 - Contain pre-fetch buffer.

23.1.3 Block diagram

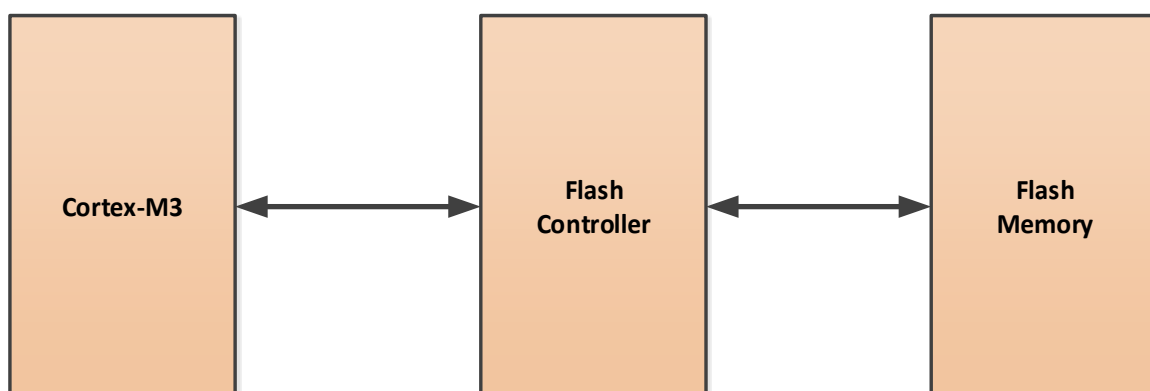


Figure 23-1 Block diagram for flash and flash controller

23.1.4 Data flow & Algorithm

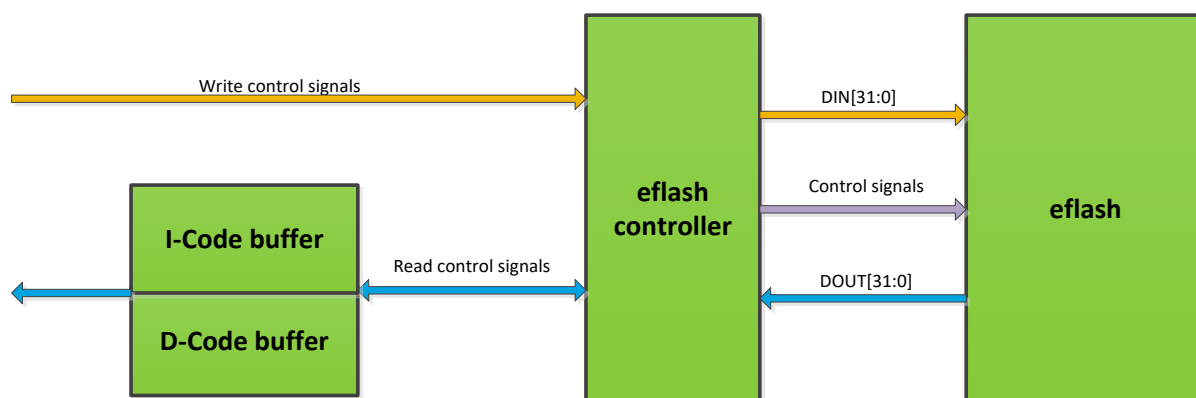


Figure 23-2 Data flow & algorithm for flash and flash controller

23.2 Embedded flash memory organization

Before introducing the commands, user should learn about the flash memory organization in [Table 23-1](#). The whole flash memory is composed of two parts: one is main memory and the other is information memory. Every 2 Kbytes is called one page in the flash memory as illustrated in [Table 23-1](#). Main memory is used for storing the user code including code and data, and user can put erase, program, verify and read command operations on main memory. The information memory is divided into 3 parts. The ISP Code section is used for solidify the code of semiconductor producer and user can't erase and program. The first 24 bytes in Option byte section are the writing and reading protection information to the main memory and the reserved bytes in option byte section can be used by user in fact. For option byte section, user can erase, program, verify and read.

Table 23-1 Flash memory organization

FLASH Memory	Name	Address	Size (bytes)	User permission (note)
Main memory (user pages)	Page 0	0x0800 0000 ~ 0x0800 07FF	2K	Erase/program/read/verify
	Page 1	0x0800 0800 ~ 0x0800 0FFF	2K	
	Page 2	0x0800 1000 ~ 0x0800 17FF	2K	
	Page 3	0x0800 1800 ~ 0x0800 1FFF	2K	
	...		2K	
	Page 127	0x0803 F800 ~ 0x0803 FFFF	2K	
Information memory	Option byte	0x0804 0000 ~ 0x0804 0014	24	Erase/program/read/verify
		0x0804 0018 ~ 0x0804 07FF (Reserved)	2024	
	ISP Code	0x0804 0800 ~ 0x0804 1fff	6K	Read

Note: user permission when writing/reading protection is invalid.

23.3 Embedded flash protect

In option page, the content mainly aims at the read protection, write protection, watch dog enable and so on. Flash memory has writing and reading protection function for the main memory in order to avoid illegal access. The related information is stored in the following option bytes and writing protection information also be loaded to the register **eFLASH_WPRT_EN1** to **eFLASH_WPRT_EN4**. After programming new data to the flash memory 0x0804 0000 ~ 0x0804 002C, the new data will be loaded into the corresponding registers and validate its corresponding functions until the next reset. Specially, the complement codes, such as nRDP/nWDTEN and so on, are implemented by the hardware automatically.

Table 23-2 The content of the key addresses in option page

Address	[31:24]	[23:16]	[15:8]	[7:0]	Default value	Comment
0x0804 0000	0xFF	nRDP	0xFF	RDP	0xFF53 FFAC	
0x0804 0004	0xFF	nWDTEN	0xFF	WDTEN	0xFFFFFFFF	
0x0804 0008	nWPRT_EN[15:0]		WPRT_EN[15:0]		0xFFFFFFFF	page 16 ~ 1
0x0804 000c	nWPRT_EN[31:16]		WPRT_EN[31:16]		0xFFFFFFFF	page 32 ~ 17
0x0804 0010	nWPRT_EN[47:32]		WPRT_EN[47:32]		0xFFFFFFFF	page 48 ~ 33
0x0804 0014	nWPRT_EN[63:48]		WPRT_EN[63:48]		0xFFFFFFFF	page 64 ~ 49
0x0804 0018	nWPRT_EN[79:64]		WPRT_EN[79:64]		0xFFFFFFFF	page 80 ~ 65
0x0804 001c	nWPRT_EN[95:80]		WPRT_EN[95:80]		0xFFFFFFFF	page 96 ~ 81
0x0804 0020	nWPRT_EN[111:96]		WPRT_EN[111:96]		0xFFFFFFFF	page 112 ~ 97
0x0804 0024	nWPRT_EN[127:112]		WPRT_EN[127:112]		0xFFFFFFFF	page 128 ~ 113
0x0804 0028	nDATA0		DATA0		0xFFFFFFFF	
0x0804 002C	nDATA1		DATA1		0xFFFFFFFF	

23.3.1 Read and write protection

For read or write protection, the following tables determine whether the main memory is under the protection.

Table 23-3 Read protection description

Conditions	RDP	nRDP	Read protection status
Case1	0xFF	0xFF	Protected
Case2	0xAC	0x53	Not protected
Other cases	Except case1 and case2		Protected

Table 23-4 Write protection description

Conditions	WPRT_EN[x]	nWPRT_EN[x]	Write protection status
Case1	0	1	Protected
Other cases	Except case1		Not protected

23.3.2 Others

For WDT enable setting, WDT must be programmed to 0xCC and the nWDT must be programmed to 0x33. Otherwise, WDT function will be disabled. For DATAx/nDATAx, user can use these two memory unit freely with no special notice.

Table 23-5 Watch dog enable description

Conditions	WDTEN	nWDTEN	Status
case1	0xCC	0x33	enable watch dog
others	Except the above		disable watch dog

23.4 Program guide

23.4.1 Brief introduction

In this section, the introduction by flow chart will be described about Page erase, Mass erase, Page program, Option page erase, Option page program, Page erase verify and Mass erase verify. All the command operation mainly refers to the following registers: eFLASH_CTRL1, eFLASH_CTRL2, eFLASH_ADR_CMD, eFIASH_SR1. For read operation, user can directly read the desired address as illustrated in Table 23-1 in flash memory, so the related description will be ignored in the document.

In particular, the following flow chart just illustrates the single command operation. And it is just significant to do the configuration of unlocking and eFIASH_CTRL2 once before one or more command operations. For the whole embedded flash memory, there are total 133 pages, which are composed of 1 option page, 3 system pages, 128 user pages.

For the following description, much emphasis will be placed on the page erase, page erase verify and page program, which other command operations can easily refer to.

23.4.2 Command operation description

23.4.2.1 Page erase

Page erase operation just acts on main memory in flash memory. Page erase operation can't reach the information memory. In the following paragraphs, detailed descriptions are introduced for page erase and other command operations can refer to it.

1. Before configuring the eFIASH_CTRL1 and eFIASH_CTRL2, we must check whether they are locked by observing the status of LOCK. If the two registers are in lock state, we must sequentially write 0xac7811 and 0x01234567 to eFIASH_KEY to unlock. If they are not in lock state, we can go to the next step directly.
2. After unlocking, we had better check whether there are some command operations in process by reading out the status of CMD_BSY. CMD_BSY equal to 1 represents some command

operations is not finished and we must wait until CMD_BSY is equal to 0. In fact, unlocking process can be done before or after checking CMD_BSY and [Figure 23-3](#) just illustrates the “before” case.

3. When CMD_BSY turns to 0, we can configure the two registers: eFLASH_CTRL1 and eFLASH_CTRL2. For all the erase and program command operations, user must configure 0x78 to eFLASH_CTRL2 when embedded flash clock works at 100MHz to get a specific clock internally. Particularly, user must adjust the value of eFLASH_CTRL2 proportionally when embedded flash clock change to other frequency but less than 160MHz certainly.
4. Then, user should configure the page erase start address in eFLASH_ADR_CMD, which value is the start address of the desired erase page. The start address for different 133 pages is described in registers eFLASH_ADR_CMD.
5. After the basic configuration above, user can launch the command and trigger it to start by controlling the eFLASH_CTRL1. User should guarantee that bit CMD_ST must turn from 0 to 1 after other bits have been configured in eFLASH_CTRL1 in order to get a valid trigger. In details, EOPIE is configured to 1 to make related status occurs after the incoming command operation will be completed. CMD_CTL is configured to 0x1 to determine the incoming command operation is page erase. Other bits should be 0 and will be used in other command operations.
6. After a valid trigger, the command operation starts. User should check whether the command operation is finished by reading out the bit CMD_BSY and EOP_INT. When CMD_BSY is equal to 0 and EOP_INT is equal to 1, it represents that the command operation has been finished and user should clear the EOP_INT by writing 0 to this bit. In fact, it is ok for user to just use CMD_BSY to indicate whether the operation is finished for all command operations. Then user can read out other status to check whether there are some errors based on user’s requirement, such as OPR_ERR, especially for write protection case, user can not erase the page with write protection character.
7. Until now, the page erase command description is finished completely as the following flow chart. To point out, the process of other command operations is similar to the page erase, just with some difference, which will be described later.

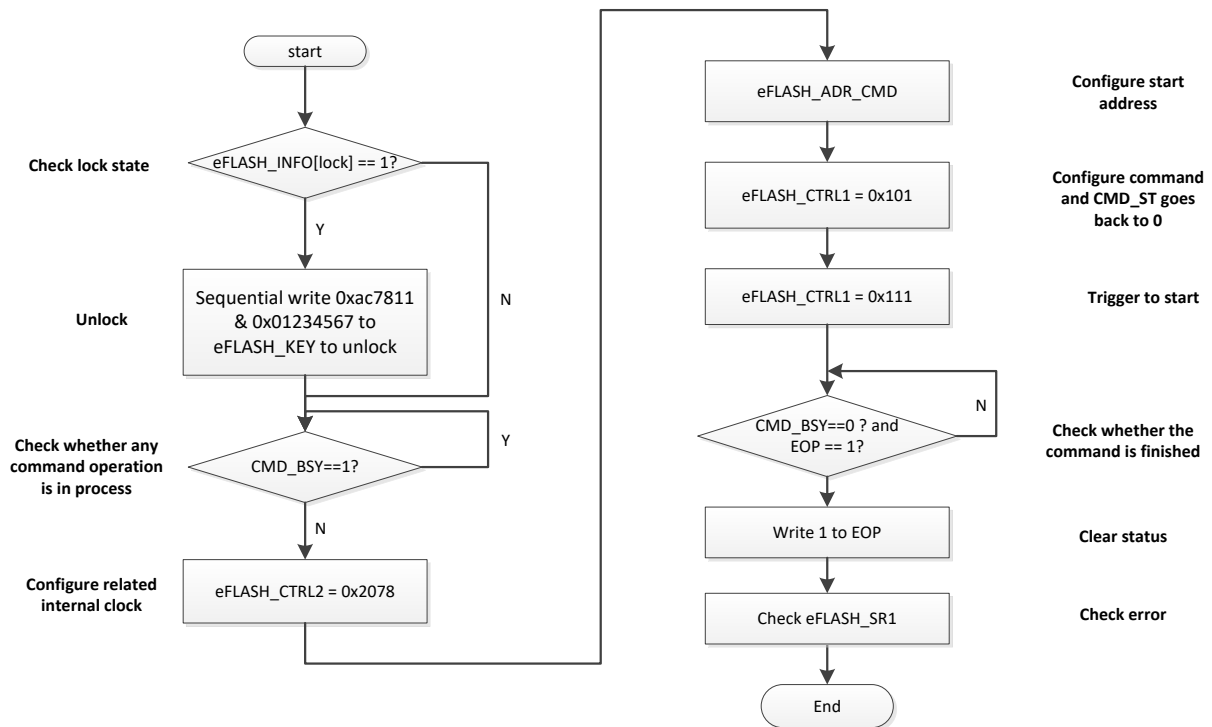


Figure 23-3 Page erase command operation flow

23.4.2.2 Mass erase

Mass erase can erase the whole user 128 pages memory. The flow is illustrated as [Figure 23-4](#). Compared with page erase, there is no need to specify the erase address in eFLASH_ADR_CMD. Specially, when the main memory attribute is changed from read-protect to read-not-protect, a mass erase will be performed automatically in order to protect the user code from illegal reading. For example, when user program the RDP in option page from the default 0xFF to 0xAC, a mass erase performs automatically.

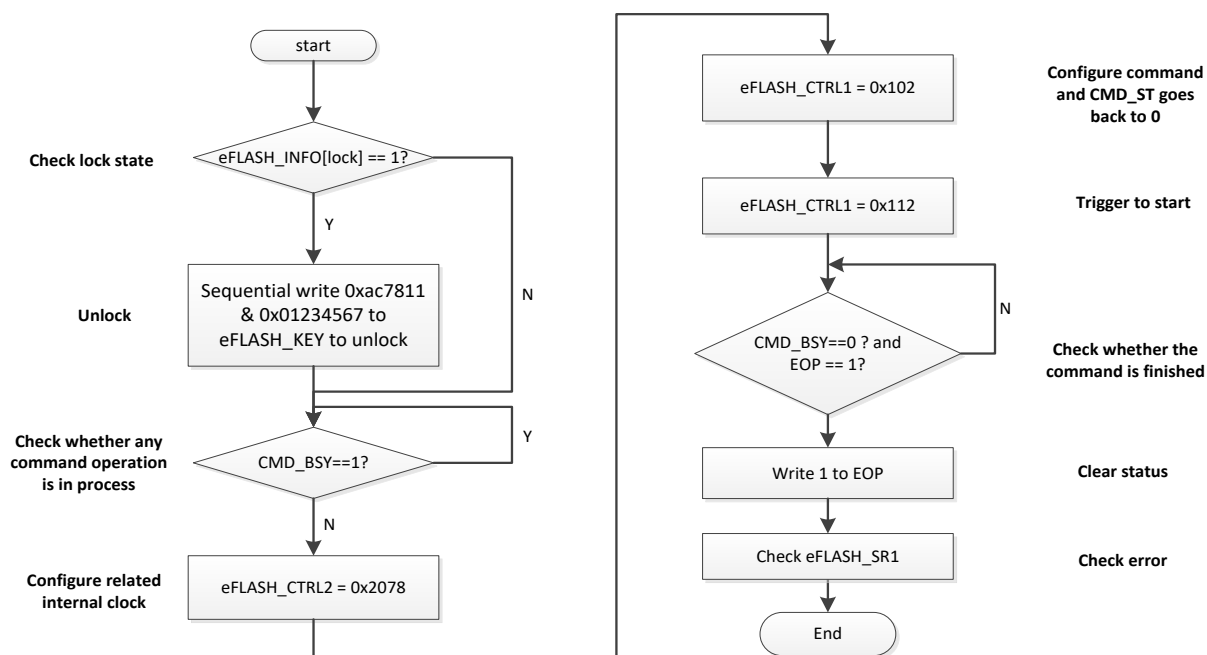


Figure 23-4 Mass erase command operation flow

23.4.2.3 Page program

Page program command can reach the whole user 128 pages memory. The flow is illustrated as [Figure 23-5](#).

The page program command flow is also similar to the page erase. But there are two differences: one is CMD_CTL and the other is PROG_LENGTH[9:0]. The bits PROG_LENGTH[9:0] are configured to a specified value to determine the program length by word. For example, if PROG_LENGTH[9:0] are configured to 150, user should write not more than 150 words. If the user write more than 150 words, such as 180 words, the last 30 words will not be written into flash in fact. Meanwhile, if the user writes less than 150 words, such as 110 words, the write operation will be validated normally, but user should keep in mind that program process will be not finished until the data number equal to length or you should use flush command to force program be terminated.

About page program command, there are some auto pre-check mechanisms to do content protection, such as write protection check, blank content check and address boundary check. So user should check eFLASH_SR1 register carefully after each program command to confirm whether the write operation is successful.

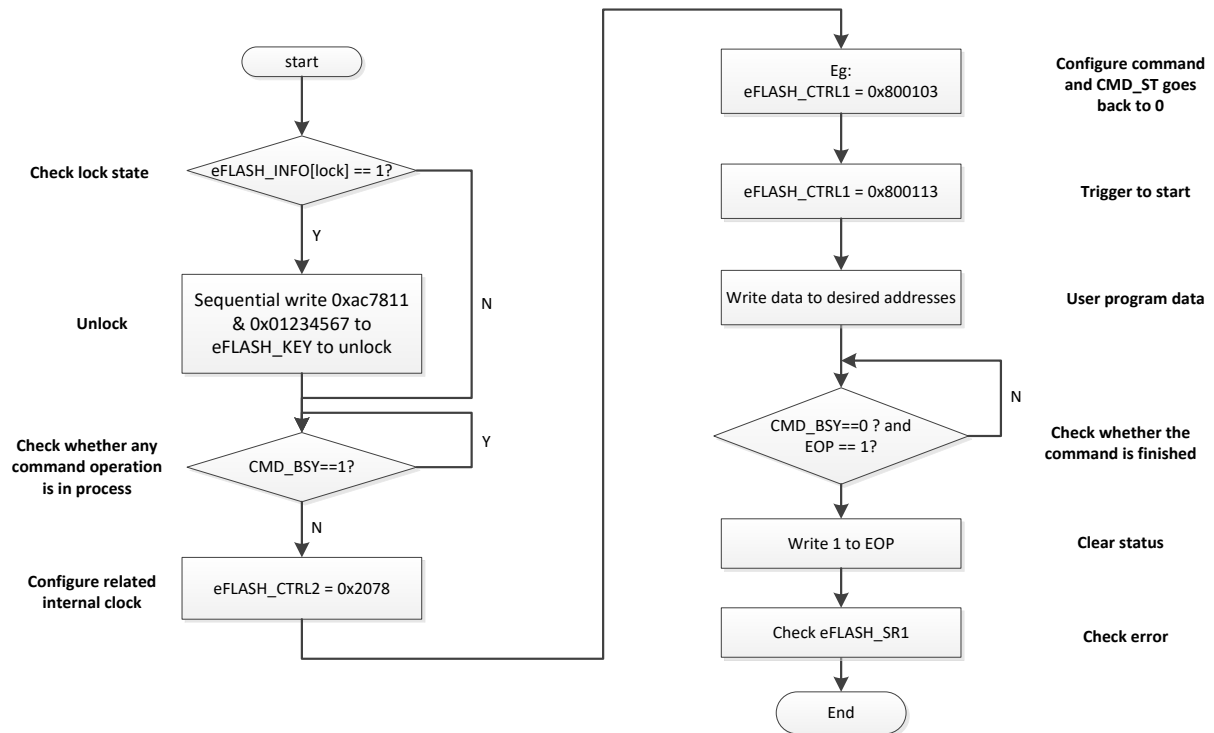


Figure 23-5 Page program command operation flow

23.4.2.4 Page erase verify

Page verify command can reach the whole user 128 pages memory and information memory. This operation is usually executed after erase operation in order to verify whether the erase operation is performed successfully. The flow is illustrated as [Figure 23-6](#). Compared with the page erase command flow, there is just one difference in page erase verify flow: CMD_CTL.

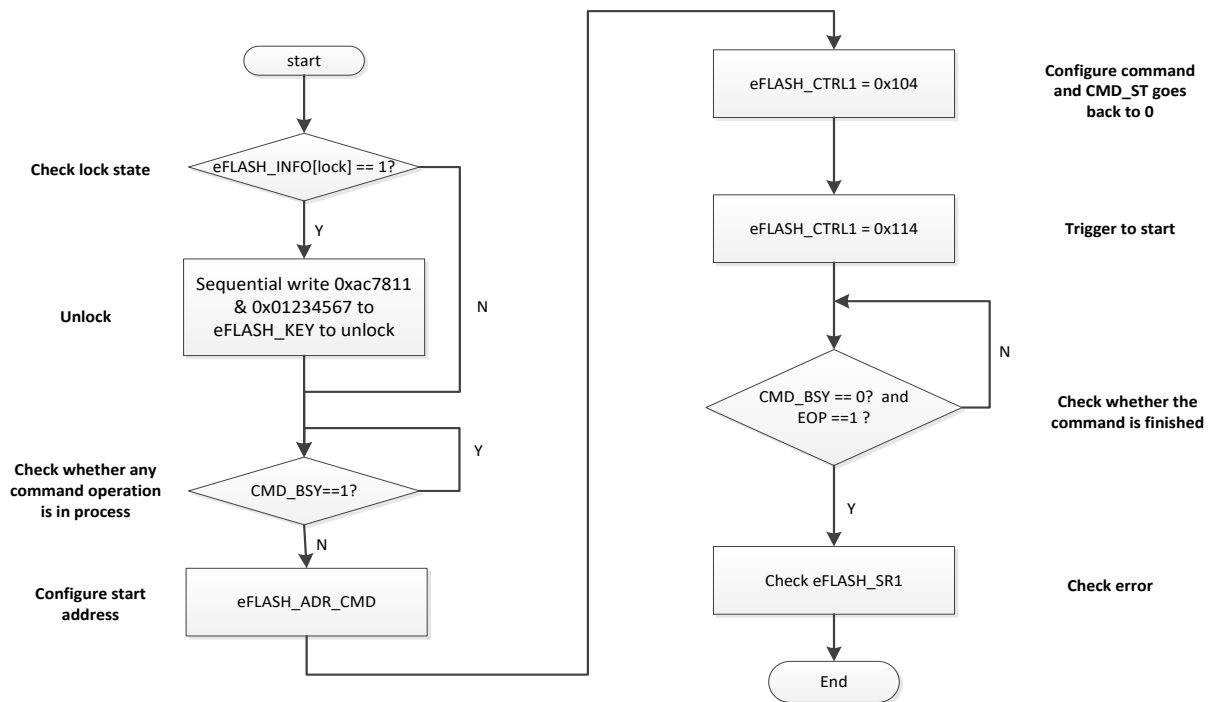


Figure 23-6 Page erase verify command operation flow

23.4.2.5 Mass erase verify

Mass erase verify command can reach the whole user 128 pages memory. This operation is usually executed after mass erase operation in order to verify whether the mass erase operation is finished successfully. The flow is illustrated as [Figure 23-7](#). Compared with the page erase command flow, there are two differences in page erase verify flow: CMD_CTL and eFLASH_ADR_CMD. Because of reaching the whole 128 pages memory, there is no need to specify the eFLASH_ADR_CMD for this command operation.

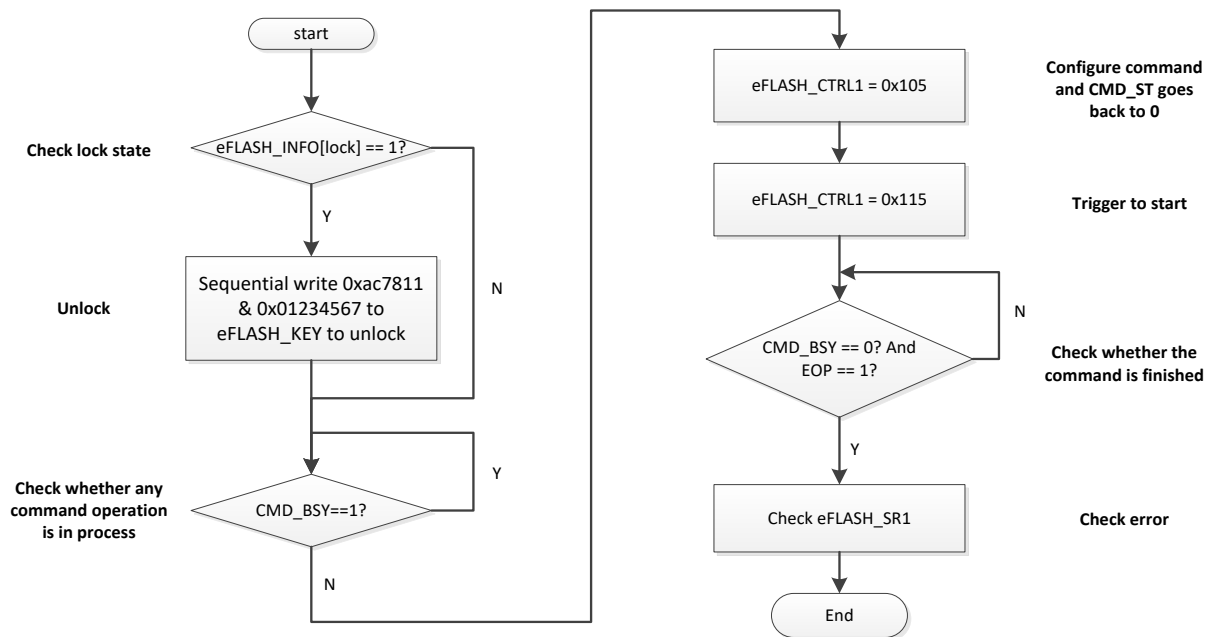


Figure 23-7 Mass erase verify command operation flow

23.4.2.6 Option page erase

Option page erase command aims at the one Option page. The flow is illustrated as [Figure 23-8](#). Compared with the page erase command flow, there is just one difference in page erase verify flow: CMD_CTL. The difference is easily described in registers map.

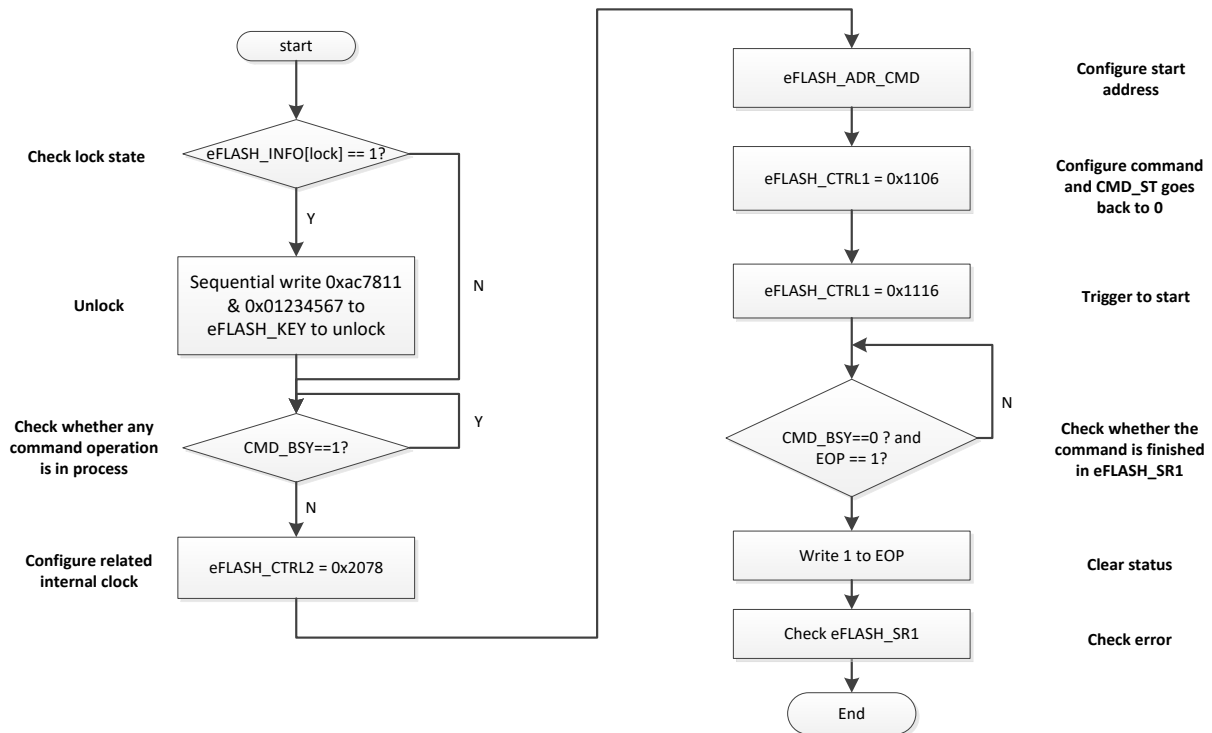


Figure 23-8 Option page erase command operation flow

23.4.2.7 Option page program

Option page erase command aims at the Option page. The flow is illustrated as [Figure 23-9](#). Compared with the page program command flow, there is just one difference in page erase verify flow: CMD_CTL. The difference is easily described in register map.

User should pay high attention to read protection character change when do option page program, if you change read protect character from protect to un-protection, eflash controller will do mass erase command automatically to erase the content of whole user pages.

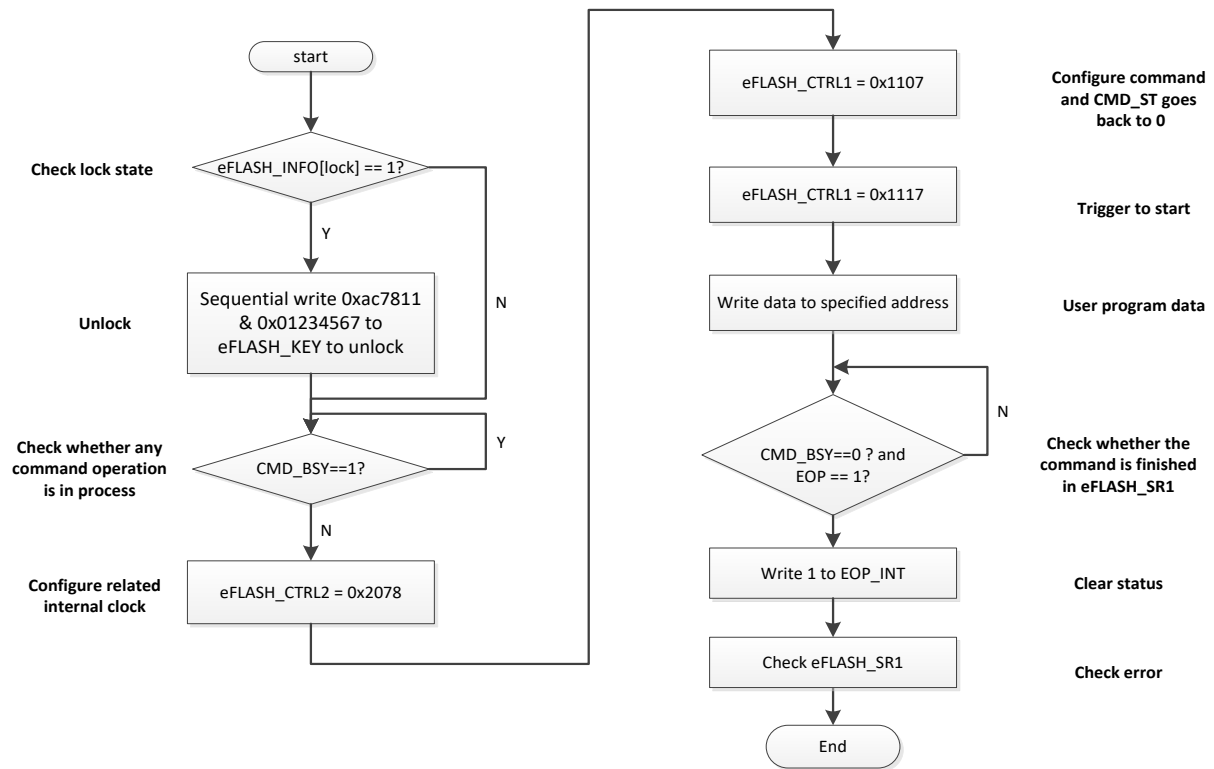


Figure 23-9 Option page program command operation flow

23.5 Register definition

Table 23-6 Embedded flash register map

Address	Name	Width	Register function
0x40002000	eFLASH_KEY	32	Unlock sequence register
0x40002008	eFLASH_ADR_CMD	32	Erase start address
0x4000200c	eFLASH_CTRL1	32	Control register 1
0x40002010	eFLASH_SR1	32	Status register
0x40002014	eFLASH_CTRL2	32	Control register 2: clock setting
0x40002018 ~ 0x40002024	eFLASH_WPRT_ENx	32	Write Protect enable bit[127:0]
0x4000202C	eFLASH_CHIP_ID1	32	Chip ID information
0x40002030	eFLASH_CHIP_ID2	32	Chip ID information
0x40002034	eFLASH_CHIP_ID3	32	Chip ID information
0x40002038	eFLASH_CHIP_ID4	32	Chip ID information

0x00 eFLASH_KEY Key sequence register RESET: 0x0

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

Bit(s)	Mnemonic	Name	Description
31:0	KEY	KEY	Unlock the writing protection of the eflash control registers The eflash control registers are indicated to be not locked when eFLASH_INFO [LOCK] bit is 0 and is locked when eFLASH_INFO [LOCK] bit is 1. Sequential write 0xac7811 & 0x01234567 to unlock the protection, then the control registers can be written again.

0x08 eFLASH_ADR_CMD Erase/program start address RESET: 0x0

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

Bit(s)	Mnemonic	Name	Description
31:0	ADR_CMD	ADR_CMD	Start address for commands Note: Before erase/program/verify, the start address must be set. ADR_CMD[31:20] must be 12'b0. (1) For page erase: ADR_CMD[19:0] is the low 20 bits of the eflash address, which is just among this page. For example, if user want to erase page 127 which range from 0x0803 F000 to 0x0803 F7FF, so, user can configure ADR_CMD[31:0] as any value from 0x0003 F000 to 0x0003 F7FF. (2) For program: ADR_CMD[19:0] is the program start address, which is the low 20 bits of eflash address. ADR_CMD[19:0] and PROG_LENGTH[9:0] are together to determine the programmable address range. (3) For page verify: ADR_CMD[19:0] is the low 20 bits of the first eflash address of this page. (4) For mass erase/verify ADR_CMD[31:0] is not cared.

0x0C eFLASH_CTRL1

Control register 1

RESET: 0x0

BIT	31	30	29	28		27	26	25	24	23	22	21	20		19	18	17	16	
NAME	HDFEN	RSV						PROG_LENGTH[9:0]											
Type	RW	R						RW											
BIT	15	14	13	12		11	10	9	8	7	6	5	4		3	2	1	0	
NAME	RSV			OPT_CMD_EN		RSV	WRPTIE	RDRTIE	EOPIE	RSV			CMD_ST		CMD_CTL[3:0]				
Type	R			RW		R	RW			R			RW		RW				

Bit(s)	Mnemonic	Name	Description
31	HDFEN	HDFEN	Hardfault interrupt enable 0: disable 1: enable
25:16	PROG_LENGTH	PROG_LENGTH	Program length for Program operation <i>Note: unit is word, Program length.</i>
12	OPT_CMD_EN	OPT_CMD_EN	Enable the command operation related with the option bytes zone <i>Note: refer to CMD_CTL</i> 0: disable 4'h6, 4'h7, enable 4'h1, 4'h2, 4'h3 1: enable 4'h6, 4'h7, disable 4'h1, 4'h2, 4'h3
10	WRPTIE	WRPTIE	Enable the interrupt of write protection error 0: disable 1: enable
9	RDRTIE	RDRTIE	Enable the interrupt of read protection error 0: disable 1: enable
8	EOPIE	EOPIE	Enable the status and interrupt of end of operation 0: disable 1: enable
4	CMD_ST	CMD_ST	Control the command operation to start Write 1 to trigger the command to start
3:0	CMD_CTL	CMD_CTL	Command 4'h0: Idle 4'h1: Page Erase 4'h2: Mass Erase 4'h3: Page Program 4'h4: Page Erase Verify 4'h5: Mass Erase Verify 4'h6: Option Page Erase 4'h7: Option Page Program

0x10 eFLASH_SR1

Status register 1

RESET: 0x702

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NAME	CMD_BSY_STA								INI_RDY	RSV		IRQ				
Type	R															
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NAME				FLUSH	OPTER				VRER	ERAER	PPADRER	PPERER	RDER	WRER	EOP	CMD_BSY
Type	R			RW	R											

Bit(s)	Mnemonic	Name	Description
31:24	CMD_BSY_STAT	CMD_BSY_STAT	<p>Indicate the busy status for each command operation</p> <p>CMD_BSY_STAT[7]: Option page program busy</p> <p>CMD_BSY_STAT[6]: Option page erase busy</p> <p>CMD_BSY_STAT[5]: Mass erase verify busy</p> <p>CMD_BSY_STAT[4]: Page erase verify busy</p> <p>CMD_BSY_STAT[3]: Page program busy</p> <p>CMD_BSY_STAT[2]: Mass erase busy</p> <p>CMD_BSY_STAT[1]: Page erase busy</p> <p>CMD_BSY_STAT[0]: Recall busy</p>
23	INI_RDY	INI_RDY	<p>Indicate the flash controller initialization finished</p> <p>0: not finished</p> <p>1: finished</p>
20	IRQ	IRQ	<p>Indicate the flash controller interrupt</p> <p>Note: There are 3 interrupt sources, which are command operation finish, write protection error and read protection error.</p> <p>0: no interrupt</p> <p>1: interrupt occur</p>
12	FLUSH	FLUSH	<p>Write 1 to fore the program command operation to be finished</p> <p>Note: This bit will not be used when user perform program command operation obeying this application note. But in some cases, the remaining memory doesn't match the value of PROG_LENGTH[9:0] or the actual program number of memory is less than PROG_LENGTH[9:0], So, the program can't be finished. For this condition, write 1 to FLUSH will force the program command operation to be finished.</p>
11:8	OPT_ER	OPT_ERR	<p>Indicate the error status for option byte</p> <p>OPT_ERR [3]:</p> <p>0: no error, namely, DATAx = ~nDATAx</p> <p>1: exist error, namely, DATAx != ~nDATAx, except DATAx = nDATAx=0xff</p> <p>OPT_ERR [2]:</p> <p>0: no error, namely, WPRT_EN[x] = ~nWPRT_EN[x]</p> <p>1: exist error, namely, WPRT_EN[x] != ~nWPRT_EN[x], except WPRT_EN[x] = ~nWPRT_EN[x] = 0x1</p> <p>OPT_ERR [1]:</p> <p>0: no error, namely, WDTEN = ~nWDTEN</p> <p>1: exist error, namely, WDTEN != ~nWDTEN, except WDTEN = nWDTEN=0xff</p> <p>OPT_ERR [0]:</p> <p>0: no error, namely, RDP = ~nRDP</p> <p>1: exist error, namely, RDP != ~nRDP, except RDP = nRDP=0xff</p>

Bit(s)	Mnemonic	Name	Description
7	VRER	VRER	Indicate whether there is error in the verify command operation 0: data verify is ok 1: data verify is not ok
6	ERAER	ERAER	Indicate whether there is error in the erase command operation 0: no error 1: error occurs, because address in eFLASH_ADR_CMD is illegal.
5	PPADRER	PPADRER	Indicate whether there is error in the page program command operation. 0: no error 1: error occurs, because program address is illegal or verify operation before program command operation is not OK.
4	PPERER	PPERER	Indicate permission error of commands operation 0: no error 1: error occurs when page/mass erase or program acts on the protected main memory
3	RDER	RDER	Indicate the operation violates the read protection rules Note: User can refer to the section 3.2.1 in details. Write 1 to clear RDER to 0. 0: no violation for the read protection rules 1: violations for the read protection rules
2	WRER	WRER	Indicate the operation violates the write protection rules Note: User can refer to the section 3.2.1 in details. Write 1 to clear WRER to 0. 0: no violation for the write protection rules 1: violations for the write protection rules
1	EOP	EOP	Indicate whether command operation is finished Note: it is not significant for user. Write 1 to clear EOP to 0. 0: not finished 1: finished
0	CMD_BSY	CMD_BSY	Indicate whether any of the command operations is in process 0: all operations are not in process 1: at least one operation in process

0x14 eFLASH_CTRL2

Control register 2

RESET: 0x9

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NAME															CLK_CHG_BSY	CYC_MANUAL
Type	R														R	RW
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NAME								CKDIV_LOCK							CKDIV[6:0]	
Type	R			R			R	RW	R						RW	

Bit(s)	Mnemonic	Name	Description
17	CLK_CHG_BSY	CLK_CHG_BSY	<p>Indicate whether ready to change eflash operation clock frequency to above 72MHz</p> <p>Note: CYC_MANUAL and CLK_CHG_BSY are always used together. After write 1 to CYC_MANUAL, CLK_CHG_BSY changes to 1. And use can change eflash operation clock to above 72MHz until CLK_CHG_BSY goes back to 0.</p>
16	CYC_MANUAL	CYC_MANUAL	<p>Control the maximum operation clock frequency for eflash</p> <p>0: embedded operation clock ≤72MHz 1: 72MHz < embedded operation clock < 120MHz</p>
8	CKDIV_LOCK	CKDIV_LOCK	<p>Lock the configuration for eFLASH_CTRL2</p> <p>Note: if this bit is written to 1 after power on, this register eFLASH_CTRL2 can't be configured again until the chip is powered down.</p> <p>0: can configure the register 1: can't configure the register</p>
6:0	CKDIV	CKDIV	<p>Clock divisor for generating 1us pulse</p> <p>Must configure to the reasonable value based on the speed of eflash controller to get 1us period before the following operation: Page program, Page erase, Mass erase based on the command operation flow. For example, if eflash controller speed is 96MHz, CKDIV= 96/1 = 96= 7'h60, however, it is recommended that this value should be larger than 7'h60, such as 7'h62.</p>

0x18~0x24	eFLASH_WPRT_ENx				Write Protection enable register x										RESET	
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NAME	WPRT_ENx[31:16]															
Type	R															
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NAME	WPRT_ENx[15:0]															
Type	R															

Note: x=1~4, four eFLASH_WPRT_ENx registers compose 128 enable bits, which determine the write protection attribution of each main memory page respectively.

Bit(s)	Mnemonic	Name	Description
31:0	WPRT_ENx	WPRT_ENx	<p>Indicate whether there exists the write protection for the corresponding main memory page.</p> <p>0: not in protection 1: in write protection</p>

0x2c eFLASH_CHIP_ID1 Chip ID 1 RESET

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NAME	UUID3									UUID2						
Type	R															
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NAME	UUID1									UUID0						
Type	R															

Bit(s)	Mnemonic	Name	Description
31:0	CHIP_ID1	CHIP_ID1	Chip ID information 1 UUID15~UUID0: 128 bit CHIP ID

0x30 eFLASH_CHIP_ID2 Chip ID 2 RESET

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NAME	UUID7									UUID6						
Type	R															
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NAME	UUID5									UUID4						
Type	R															

Bit(s)	Mnemonic	Name	Description
31:0	CHIP_ID2	CHIP_ID2	Chip ID information 2 UUID15~UUID0: 128 bit CHIP ID

0x34 eFLASH_CHIP_ID3 Chip ID 3 RESET

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NAME	UUID11									UUID10						
Type	R															
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NAME	UUID9									UUID8						
Type	R															

Bit(s)	Mnemonic	Name	Description
31:0	CHIP_ID3	CHIP_ID3	Chip ID information 3 UUID15~UUID0: 128 bit CHIP ID

0x38 eFLASH_CHIP_ID4 Chip ID 4 RESET

BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NAME	UUID15									UUID14						
Type	R															
BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NAME	UUID13									UUID12						
Type	R															

Bit(s)	Mnemonic	Name	Description
31:0	CHIP_ID4	CHIP_ID4	Chip ID information 4 UUID15~UUID0: 128 bit CHIP ID

24 Serial Flash

24.1 Serial FLASH function overview

24.1.1 Introduction

The sflash_top module is able to interface with external NOR flash memory. Its purpose is to translate the AHB/APB/RS232 transactions into the appropriate protocol. 32*32 bits SRAM is used as programming data buffer or read data cache. The timing of the NOR flash pins can be adjusted by programmable delay cells to meet the access timing requirements.

24.1.2 Feature list:

- RISC access serial flash.
- RS232 access serial flash.
- Check sum for serial flash read data.
- System boot up from serial flash.

24.1.3 Block diagram

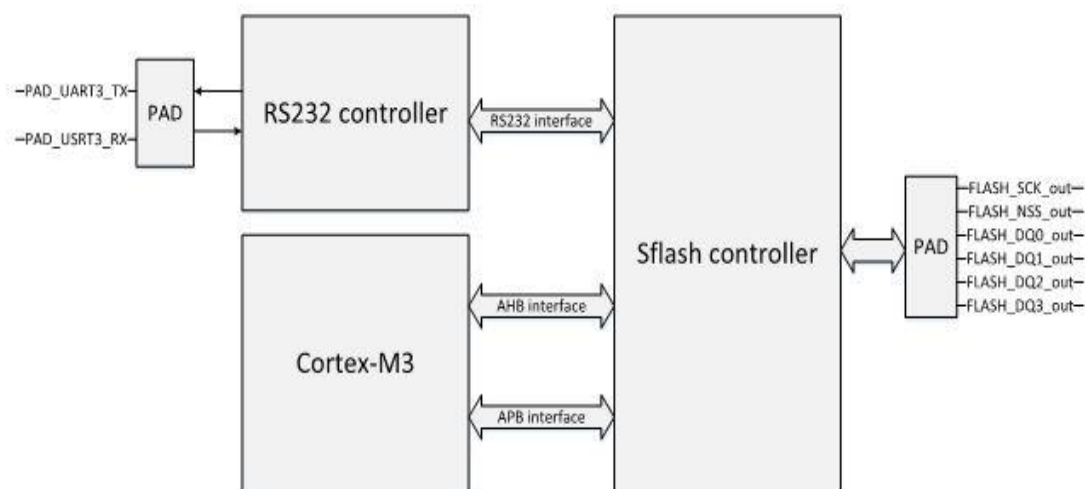


Figure 24-1 Diagram of module sflash controller

24.2 Application note

24.2.1 Clock setting

There are 3 clocks in sflash_top module: bclk, clk_flash and clk_rs232. The default frequency of APB bus clock (bclk) is half of AHB bus clock, equals to 4MHz. Default frequency of another 2 clock is the same 8MHz.

The absolute address of related clock configuration register is 0x40000000, only sflash_top module related clock is discussed, detail information is shown in the following.

Table 24-1 Clock configuration register

0x40000000 rs_reg_00		clock configuration register		
Bit(s)	Name	Type	Reset	Description
24	serial_clk_sel	RW	0	Select different source of clk_flash 0: select clock pllmut_in, frequency is 8MHz 1: select clock serial_clk_div, generated by AHB clock
3:2	rg_serial_clk_div	RW	0	Configure different division factor of serial_clk_div 00: division factor is 1/2 01: division factor is 1/4(default) 10: division factor is 1/6 11: division factor is 1/8

24.2.2 RISC access NOR flash

Multiple operations can be triggered through RISC method. All the operations are completed by related registers configuration using APB bus. Register base address and offset address is shown in chapter 24.3.

After any operation is triggered, we check the status through register REG_SF_INTREN and REG_SF_INTRSTUS, if the corresponding interrupt value is valid, writing corresponding bit to 1'b1 is necessary, a program (not page program) command is shown in the following. For more intuitive (specific steps are necessary), we ignore the following steps in the flow chart of different operation.

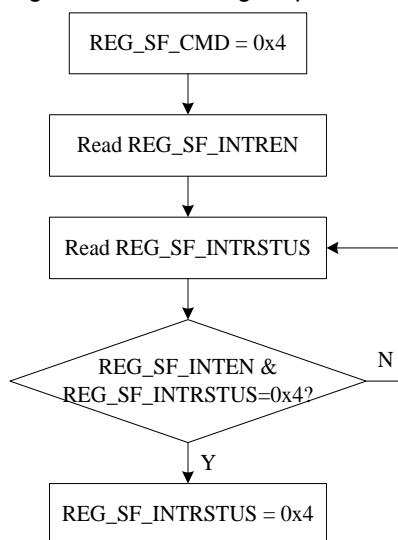


Figure 24-2 RISC access NOR flash

24.2.3 Read NOR flash id

The read identification (RDID) instruction is for reading the manufacturer ID of 1-byte and followed by Device ID of 2-byte.

The sequence of issuing RDID instruction is: CS# goes low→ sending RDID instruction code→24-bits ID data out on SO→ to end RDID operation can drive CS# to high at any time during data out.

While Program/Erase operation is in progress, it will not decode the RDID instruction, therefore there's no effect on the cycle of program/erase operation which is currently in progress. When CS# goes high, the device is at standby stage.

The specific steps are as follows:

- 1) Write REG_SF_WEPROT to 0x30;
- 2) Write REG_SF_INTREN to 0x7f;
- 3) Write REG_SF_PRGDATA5 to 0x9f;
- 4) Write REG_SF_PRGDATA4 to 0x0;
- 5) Write REG_SF_PRGDATA3 to 0x0;
- 6) Write REG_SF_PRGDATA2 to 0x0;
- 7) Write REG_SF_CNT to 0x20;
- 8) Write REG_SF_CMD to 0x4;
- 9) Read REG_SF_INTRSTUS and REG_SF_INTREN, if the value of REG_SF_INTRSTUS & REG_SF_INTREN is 0x4, then write REG_SF_INTRSTUS to 0x4;
- 10) Read REG_SF_SHREG2/1/0, we can get manufacturer ID of 1-byte and device ID of 2-byte.

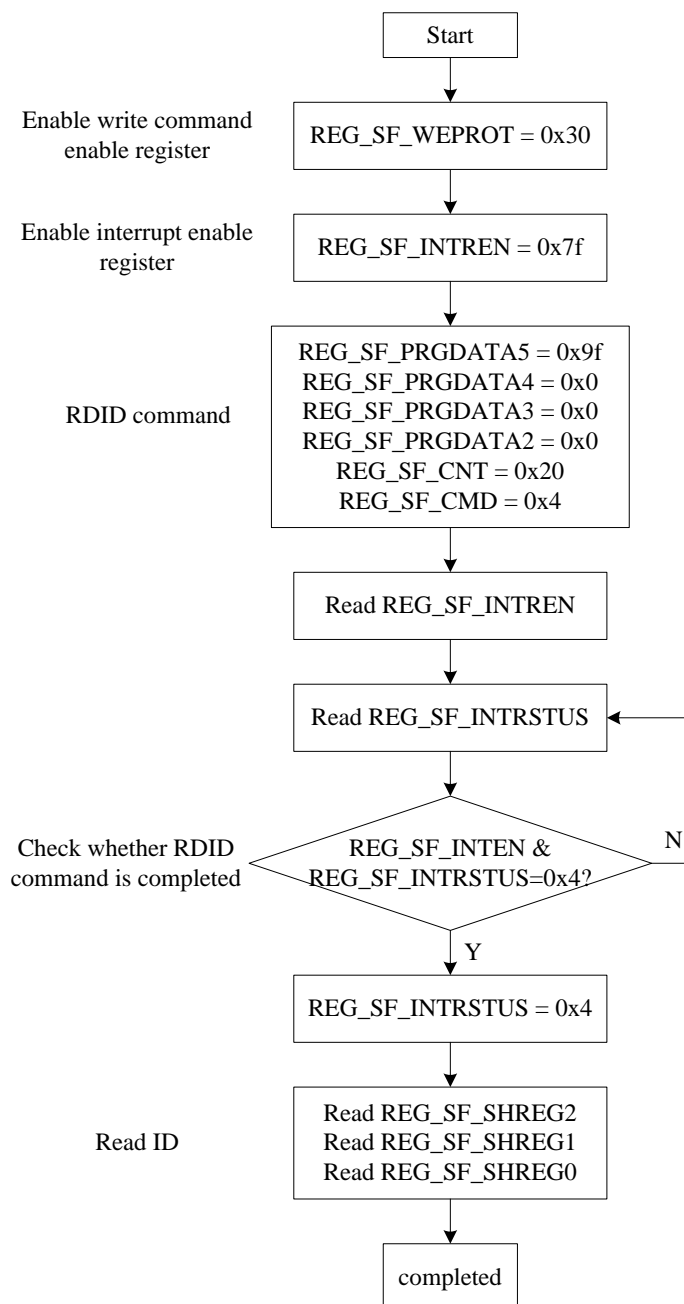


Figure 24-3 Read NOR flash id sequence

24.2.4 Write status register (WRSR)

The WRSR instruction is for changing the values of Status Register Bits and Configuration Register Bits. Before sending WRSR instruction, the Write Enable (WREN) instruction must be decoded and executed to set the Write Enable Latch (WEL) bit in advance. The WRSR instruction can change the value of Block Protect bits to define the protected area of memory.

The sequence of issuing WRSR instruction is: CS# goes low→ sending WRSR instruction code→ Status Register data on SI→CS# goes high.

The CS# must go high exactly at the 16 bits data boundary; otherwise, the instruction will be rejected and not executed. The self-timed Write Status Register cycle time (tW) is initiated as soon as Chip

Select (CS#) goes high. The Write in Progress (WIP) bit still can be check out during the Write Status Register cycle is in progress. The WIP sets 1 during the tW timing, and sets 0 when Write Status Register Cycle is completed, and the Write Enable Latch (WEL) bit is reset.

For some manufactures, the default value of block write protection bits in software status register is 1'b1, so we should **change the value of block write protection bits in status register to 1'b0 through WRSR command**. The function of each bit in software status register is depending on the type of nor flash.

The specific steps are as follows:

- 1) Write REG_SF_WEPROT to 0x30;
- 2) Write REG_SF_INTREN to 0x7f;
- 3) Write REG_SF_PRGDATA5 to 0x6;
- 4) Write REG_SF_CNT to 0x8;
- 5) Write REG_SF_CMD to 0x4;
- 6) Read REG_SF_INTRSTUS and REG_SF_INTREN, if the value of REG_SF_INTRSTUS & REG_SF_INTREN is 0x4, then write REG_SF_INTRSTUS to 0x4;
- 7) Write REG_SF_CMD to 0x2, trigger RDSR command;
- 8) Read REG_SF_INTRSTUS and REG_SF_INTREN, if the value of REG_SF_INTRSTUS & REG_SF_INTREN is 0x2, then write REG_SF_INTRSTUS to 0x2;
- 9) Read REG_SF_RDSR, if WEL = 1, turn to step 3, else continue;**
- 10) Write REG_SF_PRGDATA5 to 0x01;**
- 11) Write REG_SF_PRGDATA4 to 0x02;
- 12) Write REG_SF_CNT to 0x10;
- 13) Write REG_SF_CMD to 0x4;
- 14) Read REG_SF_INTRSTUS and REG_SF_INTREN, if the value of REG_SF_INTRSTUS & REG_SF_INTREN is 0x4, then write REG_SF_INTRSTUS to 0x4;
- 15) Write REG_SF_CMD to 0x2, trigger RDSR command;
- 16) Read REG_SF_INTRSTUS and REG_SF_INTREN, if the value of REG_SF_INTRSTUS & REG_SF_INTREN is 0x2, then write REG_SF_INTRSTUS to 0x2;
- 17) Read REG_SF_RDSR (default valid), check the value of block protection bit, if the value is 1'b1, turn to step 3.

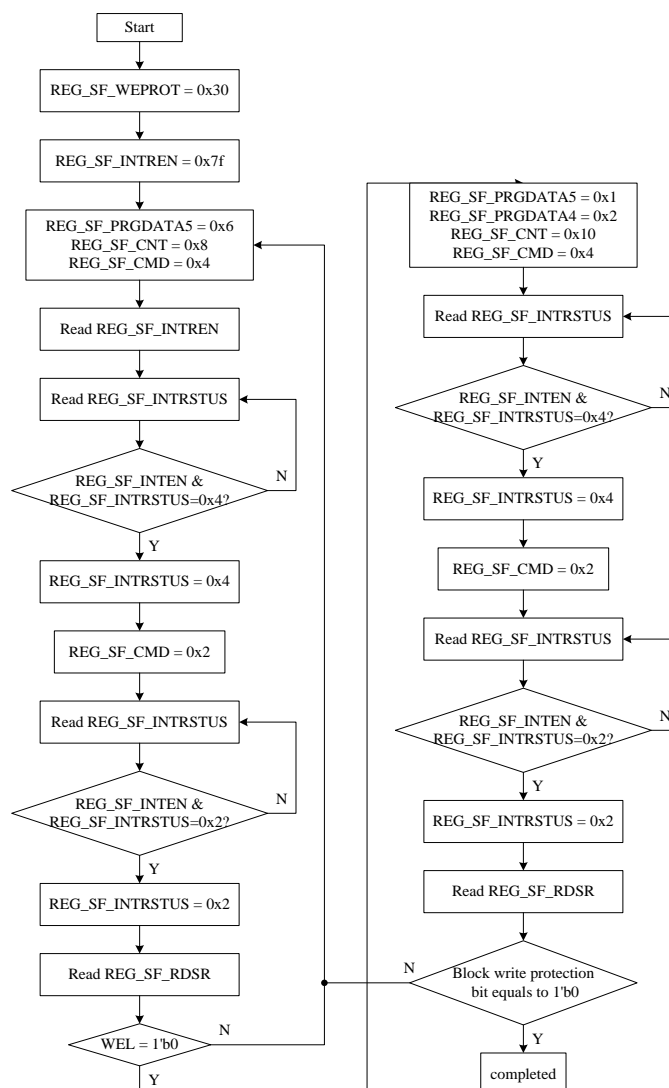


Figure 24-4 Write status register sequence

Also, **WRSR** command can be used to enable Quad Enable(QE) bit. Notice the method to set Quad Enable (QE) bit is discrepant, step 9 and step 10 in the following flow should be modified depending on the corresponding model.

RDSR command is triggered automatically when WRSR command is completed, which will continue until WIP equals to 1'b0.

The specific steps are as follows:

- 1) Write REG_SF_WEPROT to 0x30;
- 2) Write REG_SF_INTREN to 0x7f;
- 3) Write REG_SF_PRGDATA5 to 0x6;
- 4) Write REG_SF_CNT to 0x8;
- 5) Write REG_SF_CMD to 0x4;

- 6) Read REG_SF_INTRSTUS and REG_SF_INTREN, if the value of REG_SF_INTRSTUS & REG_SF_INTREN is 0x4, then write REG_SF_INTRSTUS to 0x4;
- 7) Write REG_SF_CMD to 0x2, trigger RDSR command;
- 8) Read REG_SF_INTRSTUS and REG_SF_INTREN, if the value of REG_SF_INTRSTUS & REG_SF_INTREN is 0x2, then write REG_SF_INTRSTUS to 0x2;
- 9) Read REG_SF_RDSR, if WEL = 1, turn to step 3, else continue;**
- 10) Write REG_SF_PRGDATA5 to 0x01;**
- 11) Write REG_SF_PRGDATA4 to 0x40;
- 12) Write REG_SF_CNT to 0x10;
- 13) Write REG_SF_CMD to 0x4;
- 14) Read REG_SF_INTRSTUS and REG_SF_INTREN, if the value of REG_SF_INTRSTUS & REG_SF_INTREN is 0x4, then write REG_SF_INTRSTUS to 0x4;
- 15) Write REG_SF_CMD to 0x2, trigger RDSR command;
- 16) Read REG_SF_INTRSTUS and REG_SF_INTREN, if the value of REG_SF_INTRSTUS & REG_SF_INTREN is 0x2, then write REG_SF_INTRSTUS to 0x2;
- 17) Read REG_SF_RDSR, check the value of block protection bit, if the value is 1'b1, turn to step 3.

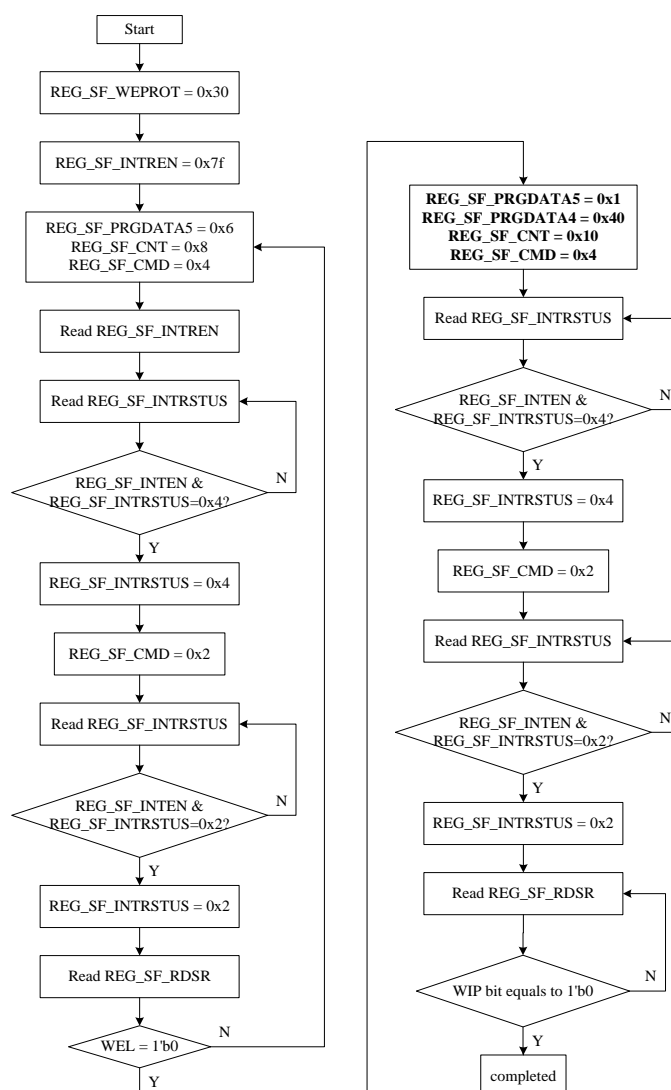


Figure 24-5 Set Quad Enable bit flow

24.2.5 Erase NOR flash (sector/block erase)

The Sector/Block Erase (SE/BE32K/BE) instruction is for erasing the data of the chosen sector to be "1". The instruction is used for any 4K-byte sector and 32K/64K-byte block. A Write Enable (WREN) instruction must execute to set the Write Enable Latch (WEL) bit before sending the SE/BE32K/BE. Any address of the sector is a valid address for SE/BE32K/BE instruction. The CS# must go high exactly at the byte boundary (the least significant bit of the address byte been latched-in). Otherwise, the instruction will be rejected and not executed.

The read mode is 3-byte address. Address bits [Am-A12]/ [Am-A15]/ [Am-A16] (Am is the most significant address) select the sector address.

The sequence of issuing SE instruction is: CS# goes low→ sending SE instruction code→ 3-byte address on SI→ CS# goes high.

The self-timed SE/BE32K/BE Cycle time (tSE/tBE32K/tBE) is initiated as soon as Chip Select (CS#) goes high. The Write in Progress (WIP) bit still can be checked while the SE/BE32K/BE cycle is in progress. The WIP sets 1 during the tSE/tBE32K/tBE timing, and clears when SE/BE32K/BE Cycle is

completed, and the Write Enable Latch (WEL) bit is cleared. If the Block is protected by BP bits (Block Protect Mode) or SPB/DPB (Advanced Sector Protect Mode), the SE/BE32K/BE instruction will not be executed on the block.

Notice the command of erase may not be the same for different manufactures. For some manufactures, only 4K-byte sector and 64K-byte block erase are supported. EQPI mode isn't supported.

The specific steps are as follows:

- 1) Write REG_SF_WEPROT to 0x30;
- 2) Write REG_SF_INTREN to 0x7f;
- 3) Release block write protection;
- 4) Write REG_SF_PRGDATA5 to 0x6;
- 5) Write REG_SF_CNT to 0x8;
- 6) Write REG_SF_CMD to 0x4;
- 7) Read REG_SF_INTRSTUS and REG_SF_INTREN, if the value of REG_SF_INTRSTUS & REG_SF_INTREN is 0x4, then write REG_SF_INTRSTUS to 0x4;
- 8) Write REG_SF_CMD to 0x2, trigger RDSR command;
- 9) Read REG_SF_INTRSTUS and REG_SF_INTREN, if the value of REG_SF_INTRSTUS & REG_SF_INTREN is 0x2, then write REG_SF_INTRSTUS to 0x2;
- 10) Read REG_SF_RDSR, if WEL = 1, turn to step 4, else continue;
- 11) Write REG_SF_PRGDATA5 to 0x20/0x52/0xd8(sector erase/32 KB block erase/64KB block erase);
- 12) Write REG_SF_PRGDATA4 to addr[23:16];
- 13) Write REG_SF_PRGDATA3 to addr[15:8];
- 14) Write REG_SF_PRGDATA2 to addr[7:0];
- 15) Write REG_SF_CNT to 0x20;
- 16) Write REG_SF_CMD to 0x4;
- 17) Read REG_SF_INTRSTUS and REG_SF_INTREN, if the value of REG_SF_INTRSTUS & REG_SF_INTREN is 0x4, then write REG_SF_INTRSTUS to 0x4;
- 18) Write REG_SF_CMD to 0x2;
- 19) Read REG_SF_INTRSTUS and REG_SF_INTREN, if the value of REG_SF_INTRSTUS & REG_SF_INTREN is 0x2, then write REG_SF_INTRSTUS to 0x2;

20) Read REG_SF_RDSR, if WIP = 0, sector/block Cycle is completed, else repeat step 20 until WIP = 0.

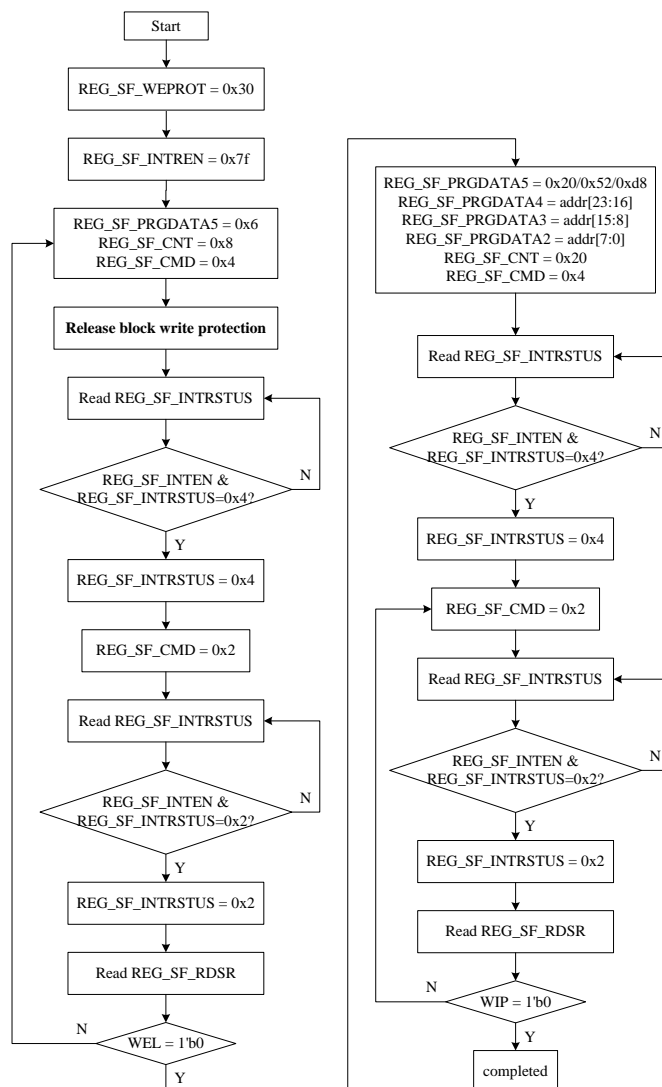


Figure 24-6 Erase NOR flash(sector/block erase) flow

24.2.6 Erase NOR flash (chip erase)

The Chip Erase (CE) instruction is for erasing the data of the whole chip to be "1". A Write Enable (WREN) instruction must be executed to set the Write Enable Latch (WEL) bit before sending the Chip Erase (CE). The CS# must go high exactly at the byte boundary, otherwise the instruction will be rejected and not executed.

The sequence of issuing CE instruction is: CS# goes low→sending CE instruction code→CS# goes high.

The self-timed Chip Erase Cycle time is initiated as soon as Chip Select (CS#) goes high. The Write in Progress (WIP) bit still can be checked while the Chip Erase cycle is in progress. The WIP sets during the tCE timing, and clears when Chip Erase Cycle is completed, and the Write Enable Latch (WEL) bit is cleared.

When the chip is under "Block protect (BP) Mode", the Chip Erase (CE) instruction will not be executed, if one (or more) sector is protected by BP3-BP0 bits. It will be only executed when BP3-BP0 all set to "0".

When the chip is under "Advances Sector Protect Mode", the Chip Erase (CE) instruction will be executed on unprotected block. The protected Block will be skipped. If one (or more) 4K byte sector was protected in top or bottom 64K byte block, the protected block will also skip the chip erase command.

The specific steps are as follows:

- 1) Write REG_SF_WEPROT to 0x30;
- 2) Write REG_SF_INTREN to 0x7f;
- 3) Release block write protection;

Solution 1:

- 4) Write REG_SF_CMD to 0x8;
- 5) Read REG_SF_INTRSTUS and REG_SF_INTREN, if the value of REG_SF_INTRSTUS & REG_SF_INTREN is 0x8, then write REG_SF_INTRSTUS to 0x8;
- 6) Write REG_SF_CMD to 0x2;
- 7) Read REG_SF_INTRSTUS and REG_SF_INTREN, if the value of REG_SF_INTRSTUS & REG_SF_INTREN is 0x2, then write REG_SF_INTRSTUS to 0x2;
- 8) Read REG_SF_RDSR, if REG_SF_RDSR[0] = 0, Chip Erase Cycle is completed, else repeat step 6 until REG_SF_RDSR[0] = 0.

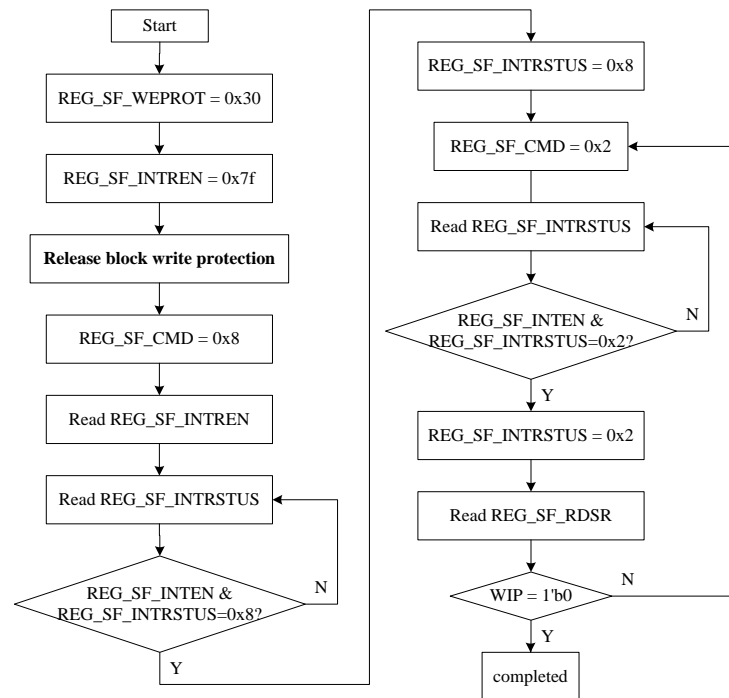


Figure 24-7 Erase NOR flash(chip erase) flow solution 1

Solution2:

- 4) Write REG_SF_PRGDATA5 to 0x6;
- 5) Write REG_SF_CNT to 0x8;
- 6) Write REG_SF_CMD to 0x4;
- 7) Read REG_SF_INTRSTUS and REG_SF_INTREN, if the value of REG_SF_INTRSTUS & REG_SF_INTREN is 0x4, then write REG_SF_INTRSTUS to 0x4;
- 8) Write REG_SF_CMD to 0x2, trigger RDSR command;
- 9) Read REG_SF_INTRSTUS and REG_SF_INTREN, if the value of REG_SF_INTRSTUS & REG_SF_INTREN is 0x2, then write REG_SF_INTRSTUS to 0x2;
- 10) Read REG_SF_RDSR, if WEL = 1, turn to step 4, else continue;
- 11) Write REG_SF_PRGDATA5 to 0xc7;
- 12) Write REG_SF_CNT to 0x08;
- 13) Write REG_SF_CMD to 0x4;
- 14) Read REG_SF_INTRSTUS and REG_SF_INTREN, if the value of REG_SF_INTRSTUS & REG_SF_INTREN is 0x4, then write REG_SF_INTRSTUS to 0x4;
- 15) Write REG_SF_CMD to 0x2;

- 16) Read REG_SF_INTRSTUS and REG_SF_INTREN, if the value of REG_SF_INTRSTUS & REG_SF_INTREN is 0x2, then write REG_SF_INTRSTUS to 0x2;
- 17) Read REG_SF_RDSR, if WIP = 0, Chip Erase Cycle is completed, else repeat step 17 until WIP = 0.

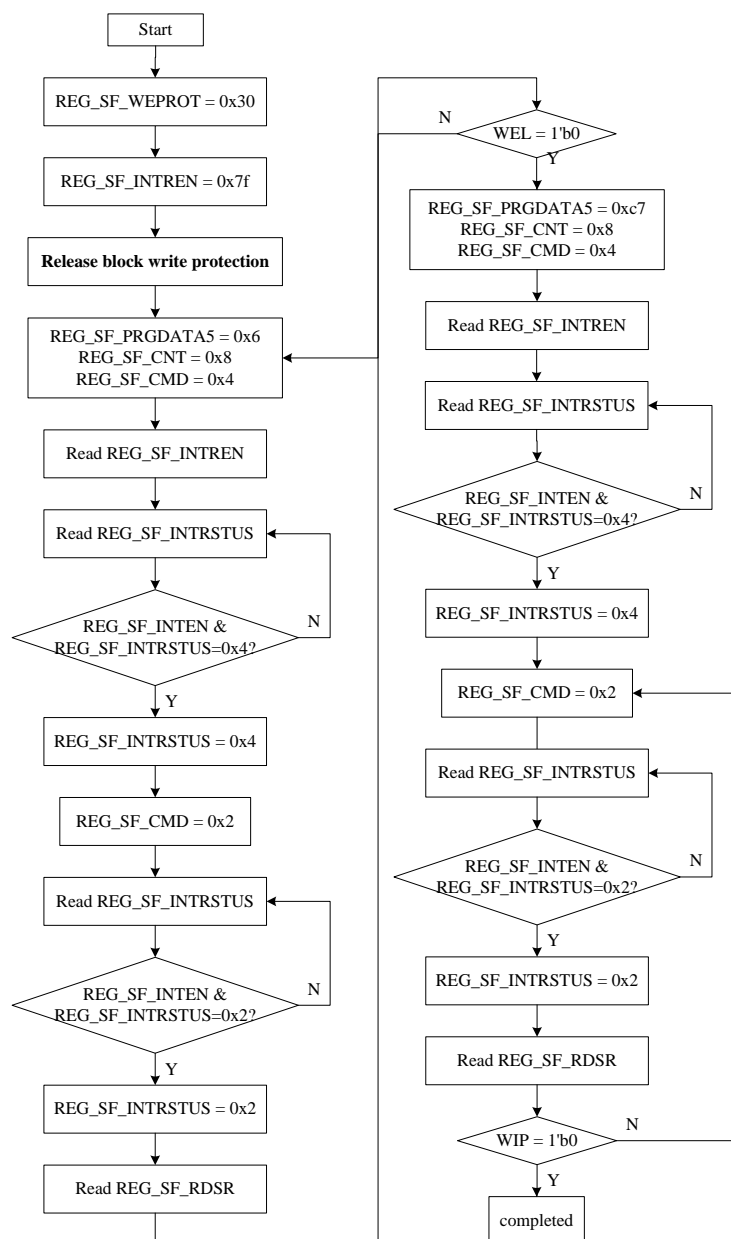


Figure 24-8 Erase NOR flash(chip erase) flow solution 2

24.2.7 Page Program NOR flash

The Page Program (PP) instruction is for programming the memory to be "0". A Write Enable (WREN) instruction must be executed to set the Write Enable Latch (WEL) bit before sending the Page Program (PP).

24.2.7.1 Buffer enable

A 128-byte buffer can be used in flashif/sf_prefetch block, program data should be written into buffer 32-bits by 32-bits through register REG_SF_PP_DW_DATA, at least two 32-bits, at most 32 32-bits data. 128 data bytes can be sent to NOR flash one time. If the entire 128 data bytes are going to be programmed, address[6:0](The seven least significant address bits) should be set to 0. If address[6:0] are not all zero, transmitted data that exceed page length are programmed from the starting address (24-bit address that last 7 bit are all 0) of currently selected page.

The sequence of issuing PP instruction is: CS# goes low→ sending PP instruction code→ 3-byte address on SI→ at least 8-byte and at most 128-byte on data on SI → CS# goes high.

The CS# must be kept to low during the whole Page Program cycle. The CS# must go high exactly at the byte boundary (the latest eighth bit of data being latched in), otherwise the instruction will be rejected and will not be executed.

The self-timed Page Program Cycle time (tPP) is initiated as soon as Chip Select (CS#) goes high. The Write in Progress (WIP) bit still can be checked while the Page Program cycle is in progress. The WIP sets during the tPP timing, and clears when Page Program Cycle is completed, and the Write Enable Latch (WEL) bit is cleared. If the page is protected by BP bits (Block Protect Mode) or SPB/DPB (Advanced Sector Protect Mode), the Page Program (PP) instruction will not be executed.

The specific steps are as follows:

- 1) Write REG_SF_WEPROT to 0x30;
- 2) Write REG_SF_INTREN to 0x7f;
- 3) Release block write protection;
- 4) Write REG_SF_CFG2 to 0x01;
- 5) Write REG_SF_PP_DW_DATA to expected value(32 bits), at least two 32-bits, at most 32 32-bits data(buffer full);
- 6) Write REG_SF_RADR2 to expected address addr[23:16];
- 7) Write REG_SF_RADR1 to expected address addr[15:8];
- 8) Write REG_SF_RADR0 to expected address addr[7:0];
- 9) Write REG_SF_CMD to 0x10;
- 10) Read REG_SF_INTRSTUS and REG_SF_INTREN, if the value of REG_SF_INTRSTUS & REG_SF_INTREN is 0x10, then write REG_SF_INTRSTUS to 0x10;
- 11) Write REG_SF_CMD to 0x2;
- 12) Read REG_SF_INTRSTUS and REG_SF_INTREN, if the value of REG_SF_INTRSTUS & REG_SF_INTREN is 0x2, then write REG_SF_INTRSTUS to 0x2, else turn to step 12;
- 13) Read REG_SF_RDSR, if WIP = 0, Page program Cycle is completed, else turn to step 11.

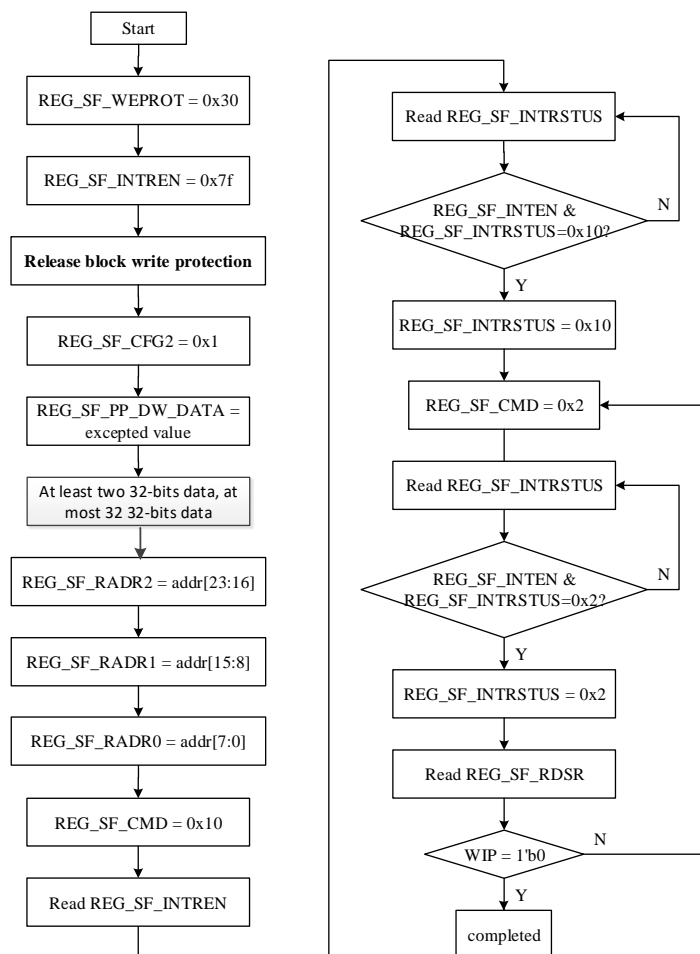


Figure 24-9 Buffer enable flow

24.2.7.2 Buffer disable

The 128-byte buffer isn't used for page program, only 1 data byte can be sent to NOR flash one time. The sequence of issuing PP instruction is: CS# goes low→ sending PP instruction code→ 3-byte address on SI→ 1-byte on data on SI → CS# goes high.

The CS# must be kept to low during the whole Page Program cycle. The CS# must go high exactly at the byte boundary (the latest eighth bit of data being latched in), otherwise the instruction will be rejected and will not be executed.

The self-timed Page Program Cycle time (tPP) is initiated as soon as Chip Select (CS#) goes high. The Write in Progress (WIP) bit still can be checked while the Page Program cycle is in progress. The WIP sets during the tPP timing, and clears when Page Program Cycle is completed, and the Write Enable Latch (WEL) bit is cleared. If the page is protected by BP bits (Block Protect Mode) or SPB/DPB (Advanced Sector Protect Mode), the Page Program (PP) instruction will not be executed. The specific steps are as follows:

- 1) Write REG_SF_WEPROT to 0x30;
- 2) Write REG_SF_INTREN to 0x7f;
- 3) Release block write protection;

- 4) Write REG_SF_CFG2 to 0x0;
- 5) Write REG_SF_WDATA to expected value(8 bits);
- 6) Write REG_SF_RADR2 to expected address addr[23:16];
- 7) Write REG_SF_RADR1 to expected address addr[15:8];
- 8) Write REG_SF_RADR0 to expected address addr[7:0];
- 9) Write REG_SF_CMD to 0x10;
- 10) Read REG_SF_INTRSTUS and REG_SF_INTREN, if the value of REG_SF_INTRSTUS & REG_SF_INTREN is 0x10, then write REG_SF_INTRSTUS to 0x10;
- 11) Write REG_SF_CMD to 0x2;
- 12) Read REG_SF_INTRSTUS and REG_SF_INTREN, if the value of REG_SF_INTRSTUS & REG_SF_INTREN is 0x2, then write REG_SF_INTRSTUS to 0x2, else turn to step 12;
- 13) Read REG_SF_RDSR, if WIP = 0, Page program Cycle is completed, else turn to step 11.

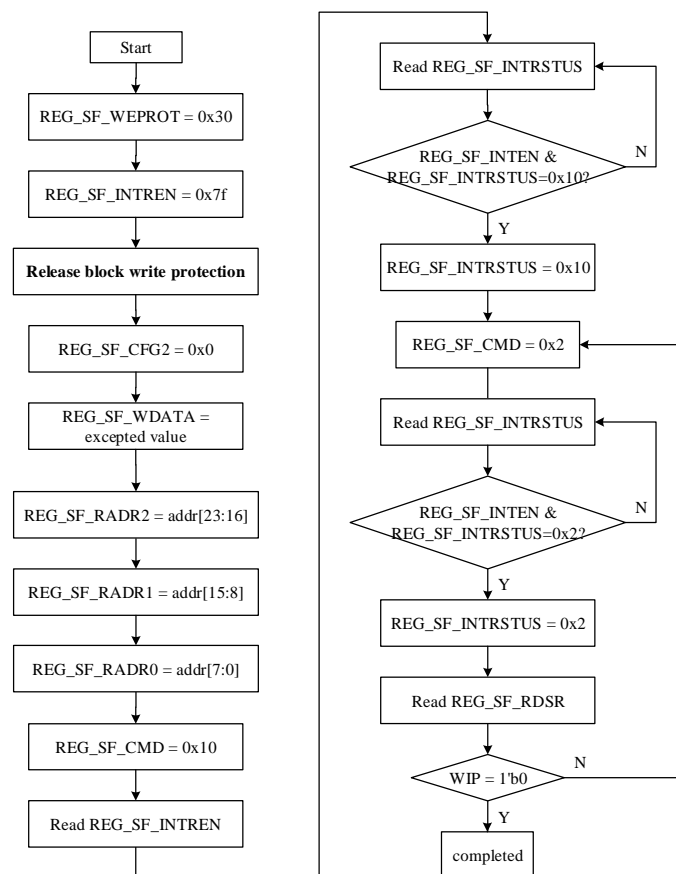


Figure 24-10 Buffer disable flow

24.2.8 4 x I/O Page Program(4PP)

The Quad Page Program (4PP) instruction is for programming the memory to be "0". A Write Enable (WREN) instruction must be executed to set the Write Enable Latch (WEL) bit and Quad Enable (QE) bit must be set to "1" before sending the Quad Page Program (4PP). The Quad Page Programming take four pins: SIO0, SIO1, SIO2, and SIO3, which raise programmer performance and the effectiveness of application.

24.2.8.1 Buffer enable

The sequence of issuing 4PP instruction is: CS# goes low→ sending 4PP instruction code→ 3-byte address on SIO[3:0]→ at least 8-byte and at most 128-byte on data on SIO[3:0]→ CS# goes high. Notice the method to set Quad Enable(QE) bit is discrepant for different manufactures. EQPI mode isn't supported.

The specific steps are as follows:

- 1) Write REG_SF_WEPROT to 0x30;
- 2) Write REG_SF_INTREN to 0x7f;
- 3) Release block write protection and enable Quad Enable(QE) bit;
- 4) Write REG_QSPI_CFG to 0x4/0x6;
- 5) Write REG_SF_CFG2 to 0x11;
- 6) Write REG_SF_PRGDATA0 to 0x32/0x38;
- 7) Write REG_SF_PP_DW_DATA to expected value(32 bits), at least two 32-bits, at most 32 32-bits data(buffer full);
- 8) Write REG_SF_RADR2 to expected address addr[23:16];
- 9) Write REG_SF_RADR1 to expected address addr[15:8];
- 10) Write REG_SF_RADR0 to expected address addr[7:0];
- 11) Write REG_SF_CMD to 0x10;
- 12) Read REG_SF_INTRSTUS and REG_SF_INTREN, if the value of REG_SF_INTRSTUS & REG_SF_INTREN is 0x10, then write REG_SF_INTRSTUS to 0x10;
- 13) Write REG_SF_CMD to 0x2;
- 14) Read REG_SF_INTRSTUS and REG_SF_INTREN, if the value of REG_SF_INTRSTUS & REG_SF_INTREN is 0x2, then write REG_SF_INTRSTUS to 0x2, else turn to step 12;
- 15) Read REG_SF_RDSR, if WIP = 0, Page program Cycle is completed, else turn to step 11.

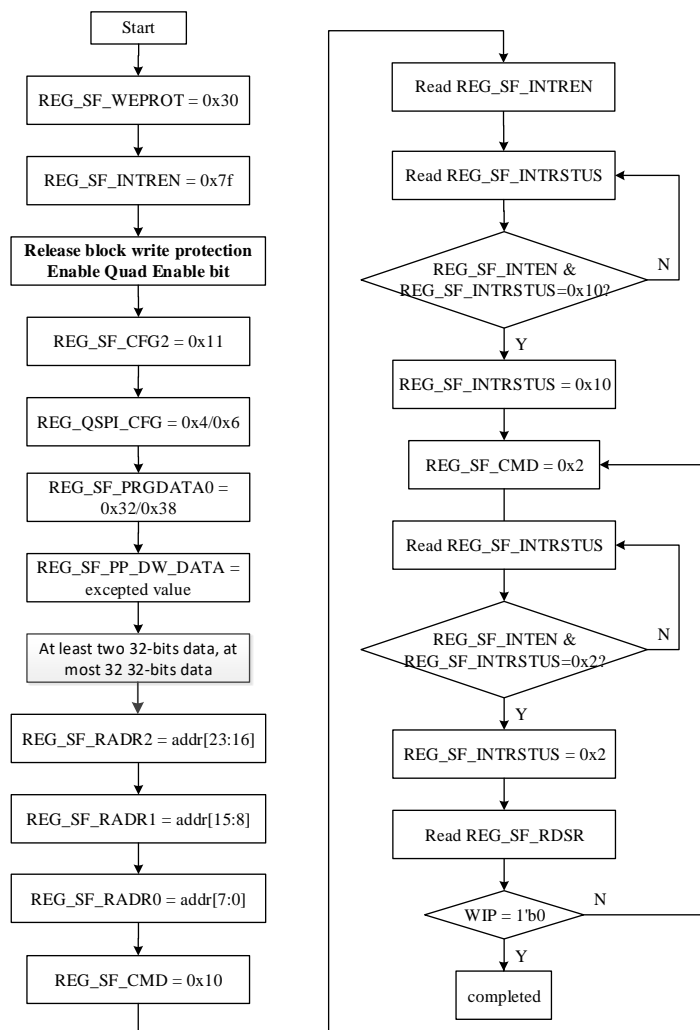


Figure 24-11 Buffer enable flow

24.2.8.2 Buffer disable

The 128-byte buffer isn't used for page program, only 1 data byte can be sent to NOR flash one time. The sequence of issuing 4PP instruction is: CS# goes low→ sending 4PP instruction code→ 3-byte address on SIO[3:0]→ 1-byte data on SIO[3:0]→ CS# goes high. Notice the method to set Quad Enable(QE) bit is discrepant for different manufactures. EQPI mode isn't supported.

The specific steps are as follows:

- 1) Write REG_SF_WEPROT to 0x30;
- 2) Write REG_SF_INTREN to 0x7f;
- 3) Release block write protection and enable Quad Enable(QE) bit;
- 4) Write REG_QSPI_CFG to 0x4/0x6;
- 5) Write REG_SF_CFG2 to 0x10;

- 6) Write REG_SF_PRGDATA0 to 0x32/0x38;

- 7) Write REG_SF_WDATA to expected value(8 bits);
- 8) Write REG_SF_RADR2 to expected address addr[23:16];
- 9) Write REG_SF_RADR1 to expected address addr[15:8];
- 10) Write REG_SF_RADR0 to expected address addr[7:0];
- 11) Write REG_SF_CMD to 0x10;
- 12) Read REG_SF_INTRSTUS and REG_SF_INTREN, if the value of REG_SF_INTRSTUS & REG_SF_INTREN is 0x10, then write REG_SF_INTRSTUS to 0x10;
- 13) Write REG_SF_CMD to 0x2;
- 14) Read REG_SF_INTRSTUS and REG_SF_INTREN, if the value of REG_SF_INTRSTUS & REG_SF_INTREN is 0x2, then write REG_SF_INTRSTUS to 0x2, else turn to step 12;
- 15) Read REG_SF_RDSR, if WIP = 0, Page program Cycle is completed, else turn to step 11.

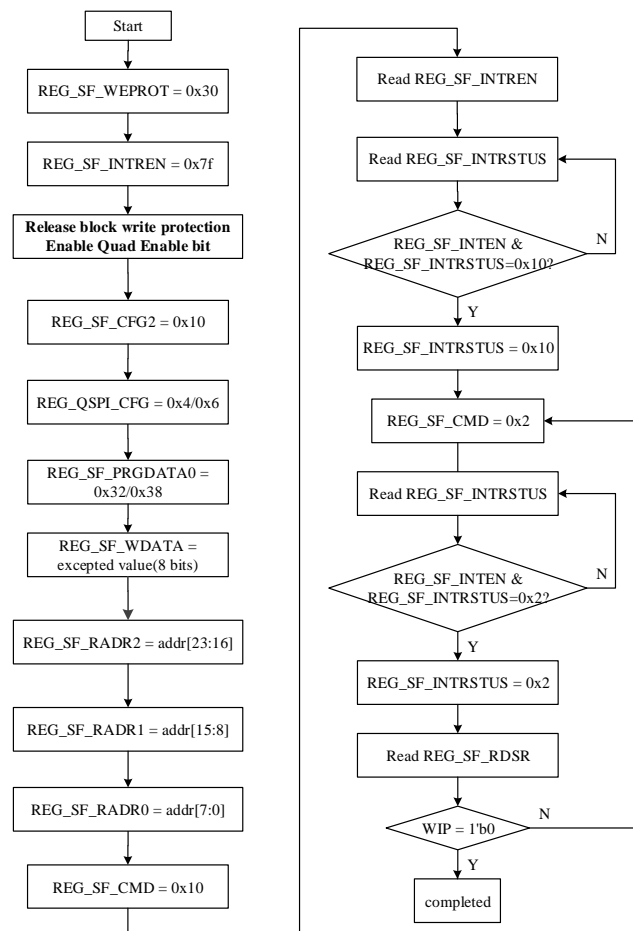


Figure 24-12 Buffer disable flow

24.2.9 AAI program NOR flash

The AAI program instruction allows multiple bytes of data to be programmed without re-issuing the next sequential address location. This feature decreases total programming time when the multiple bytes or entire memory array is to be programmed. An AAI program instruction pointing to a protected memory area will be ignored. The selected address range must be in the erased state (FFH) when initiating an AAI program instruction.

While within AAI WORD programming sequence, the only valid instructions are AAI WORD program operation, RDSR, WRDI. Software detection by polling the BUSY in the software status register can be used to determine the completion of each AAI WORD program cycle.

Prior to any write operation, the Write Enable (WREN) instruction must be executed. The AAI WORD program instruction is initiated by executing an 8-bit command, ADH (ESMT F25L16PA), followed by address bits [A23 -A0]. Following the addresses, two bytes of data is input sequentially. If the 8-bit command isn't ADH, 2 methods can be used to configure the operation code, refer to the following step 9 for details.

Chip Select (CS#) must be driven high before the AAI WORD program instruction is executed. The user must check the busy status before entering the next valid command. Once the device indicates it is no longer busy, data for next two sequential addresses may be programmed and so on. After entering the last desired byte, use the RDSR instruction to check the busy status and execute the WRDI instruction to terminate the AAI. User must check busy status after WRDI to determine if the device is ready for any command. There is no wrap mode during AAI programming. Once the highest unprotected memory address is reached, the device will exit AAI operation and reset the Write-Enable-Latch bit (WEL = 0) and the AAI bit (AAI=0).

The specific steps are as follows:

- 1) Write REG_SF_WEPROT to 0x30;
- 2) Write REG_SF_INTREN to 0x7f;
- 3) Release block write protection;
- 4) Write REG_SF_CFG2 to 0x1;
- 5) Write REG_SF_PP_DW_DATA to expected value(32 bits), repeat 32 times until buffer is full;
- 6) Write REG_SF_RADR2 to expected address addr[23:16];
- 7) Write REG_SF_RADR1 to expected address addr[15:8];
- 8) Write REG_SF_RADR0 to expected address addr[7:0];
- 9) Select suitable operation code

Solution 1:

Write REG_SF_CFG2 to 0x01, 8-bit command is AFH;

Solution 2:

Write REG_SF_CFG2 to 0x41, 8-bit command is ADH;

Solution 3:

Write REG_SF_PRGDATA0 to 0x**;

Write REG_SF_CFG2 to 0x11, 8-bit command is 0x**;

- 10) Write REG_SF_AAICMD to 0x1;

- 11) Read REG_SF_INTRSTUS and REG_SF_INTREN, if the value of REG_SF_INTRSTUS & REG_SF_INTREN is 0x40, then write REG_SF_INTRSTUS to 0x40;
- 12) Write REG_SF_CMD to 0x2;
- 13) Read REG_SF_INTRSTUS and REG_SF_INTREN, if the value of REG_SF_INTRSTUS & REG_SF_INTREN is 0x2, then write REG_SF_INTRSTUS to 0x2, else turn to step 13;
- 14) Read REG_SF_RDSR, if WIP = 0, AAI program Cycle is completed, else repeat step 14.

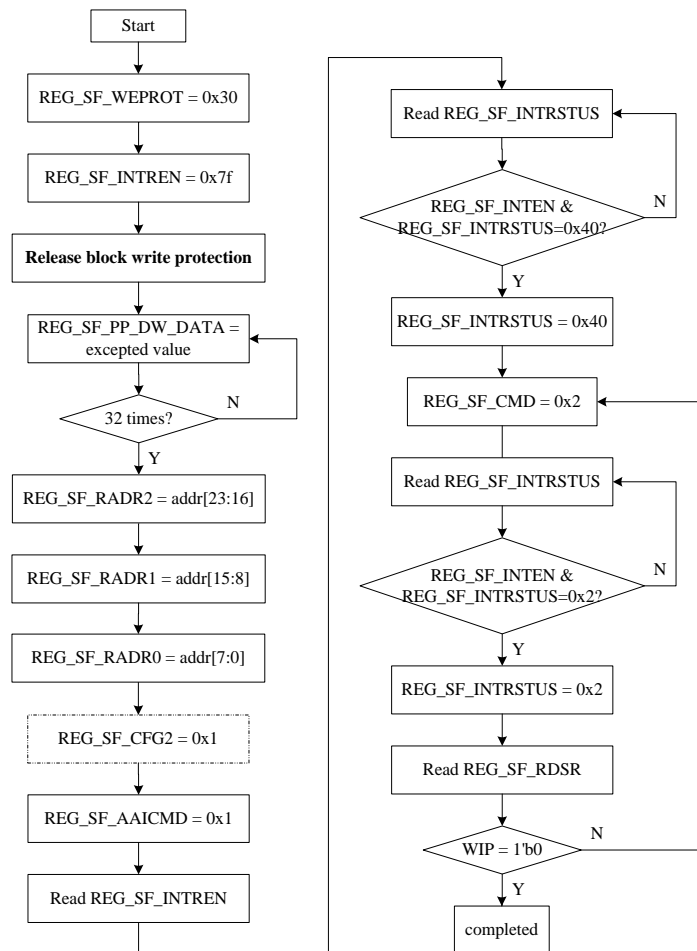


Figure 24-13 AAI program NOR flash

24.2.10 Read NOR flash

24.2.10.1 Normal read NOR flash

The read instruction is for reading data out. The address is latched on rising edge of SCLK, and data shifts out on the falling edge of SCLK at a maximum frequency fR. The first address byte can be at any location. The address is automatically increased to the next higher address after each byte data is shifted out, so the whole memory can be read out at a single READ instruction. The address counter

rolls over to 0 when the highest address has been reached. Read Cycle is completed when 128-byte buffer is full.

The sequence of issuing READ instruction is: CS# goes low→sending READ instruction code→ 3-byte address on SI→ data out on SO→to end READ operation can use CS# to high at any time during data out.

The specific steps are as follows:

- 1) Write REG_SF_WEPROT to 0x30;
- 2) Write REG_SF_INTREN to 0x7f;
- 3) Write REG_SF_PRGDATA5 to 0x6;
- 4) Write REG_SF_CNT to 0x8;
- 5) Write REG_SF_CMD to 0x4;
- 6) Read REG_SF_INTRSTUS and REG_SF_INTREN, if the value of REG_SF_INTRSTUS & REG_SF_INTREN is 0x4, then write REG_SF_INTRSTUS to 0x4;
- 7) Write REG_SF_CMD to 0x2, trigger RDSR command;
- 8) Read REG_SF_INTRSTUS and REG_SF_INTREN, if the value of REG_SF_INTRSTUS & REG_SF_INTREN is 0x2, then write REG_SF_INTRSTUS to 0x2;
- 9) Read REG_SF_RDSR, if WEL = 1, turn to step 3, else continue;
- 10) Write REG_SF_RADR2 to expected address addr[23:16];
- 11) Write REG_SF_RADR1 to expected address addr[15:8];
- 12) Write REG_SF_RADR0 to expected address addr[7:0];
- 13) Write REG_SF_CMD to 0x1;
- 14) Read REG_SF_INTRSTUS and REG_SF_INTREN, if the value of REG_SF_INTRSTUS & REG_SF_INTREN is 0x1, then write REG_SF_INTRSTUS to 0x1;
- 15) All 128-byte data are saved in buffer, if we want to read out the data, we should write REG_SF_CMD to 0x81, read REG_SF_INTRSTUS and REG_SF_INTREN, if the value of REG_SF_INTRSTUS & REG_SF_INTREN is 0x1, then write REG_SF_INTRSTUS to 0x1, and then read REG_SF_RDATA, related operation in the above should repeat 128 times.

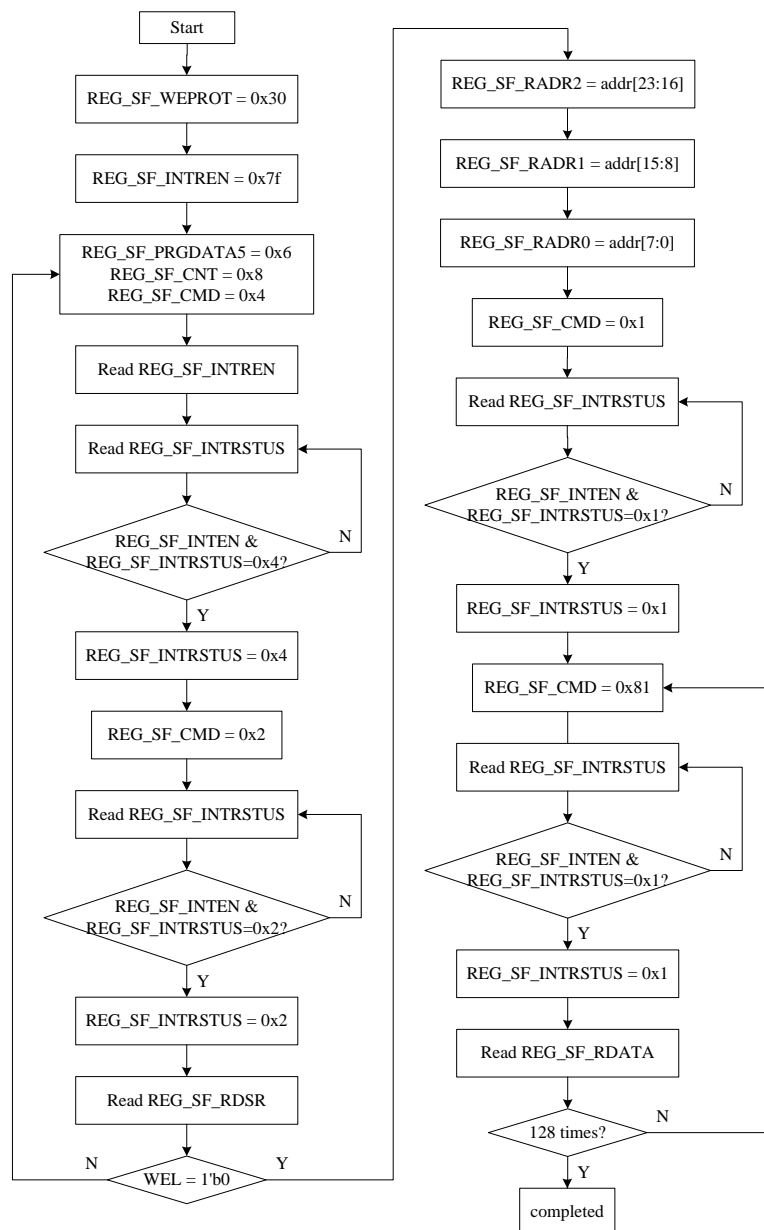


Figure 24-14 Normal read NOR flash flow

24.2.10.2 Fast read NOR flash

The FAST READ instruction is used to read data quickly. The address is latched on rising edge of SCLK, and data of each bit shifts out on the falling edge of SCLK at a maximum frequency f_C . The first address byte can be at any location. The address is automatically increased to the next higher address after each byte data is shifted out, so the whole memory can be read out at a single FAST_READ instruction. The address counter rolls over to 0 when the highest address has been reached.

The sequence of issuing FAST_READ instruction is: CS# goes low→ sending FAST_READ instruction code→3-byte address on SI→ 8 dummy cycles (default)→ data out on SO→ to end FAST_READ operation can use CS# to high at any time during data out.

While Program/Erase/Write Status Register cycle is in progress, FAST_READ instruction is rejected without any impact on the Program/Erase/Write Status Register current cycle.

The specific steps are as follows:

- 1) Write REG_SF_WEPROT to 0x30;
- 2) Write REG_SF_INTREN to 0x7f;
- 3) Write REG_SF_PRGDATA5 to 0x6;
- 4) Write REG_SF_CNT to 0x8;
- 5) Write REG_SF_CMD to 0x4;
- 6) Read REG_SF_INTRSTUS and REG_SF_INTREN, if the value of REG_SF_INTRSTUS & REG_SF_INTREN is 0x4, then write REG_SF_INTRSTUS to 0x4;
- 7) Write REG_SF_CMD to 0x2, trigger RDSR command;
- 8) Read REG_SF_INTRSTUS and REG_SF_INTREN, if the value of REG_SF_INTRSTUS & REG_SF_INTREN is 0x2, then write REG_SF_INTRSTUS to 0x2;
- 9) Read REG_SF_RDSR, if WEL = 1, turn to step 3, else continue;
- 10) Write REG_SF_RADR2 to expected address addr[23:16];
- 11) Write REG_SF_RADR1 to expected address addr[15:8];
- 12) Write REG_SF_RADR0 to expected address addr[7:0];
- 13) Write REG_SF_CFG1 to 0x1;
- 14) Write REG_SF_CMD to 0x1;
- 15) Read REG_SF_INTRSTUS and REG_SF_INTREN, if the value of REG_SF_INTRSTUS & REG_SF_INTREN is 0x1, then write REG_SF_INTRSTUS to 0x1;
- 16) All 128-byte data are saved in buffer, if we want to read out the data, we should write REG_SF_CMD to 0x81, and then read REG_SF_RDATA, related operation in the above should repeat 128 times.

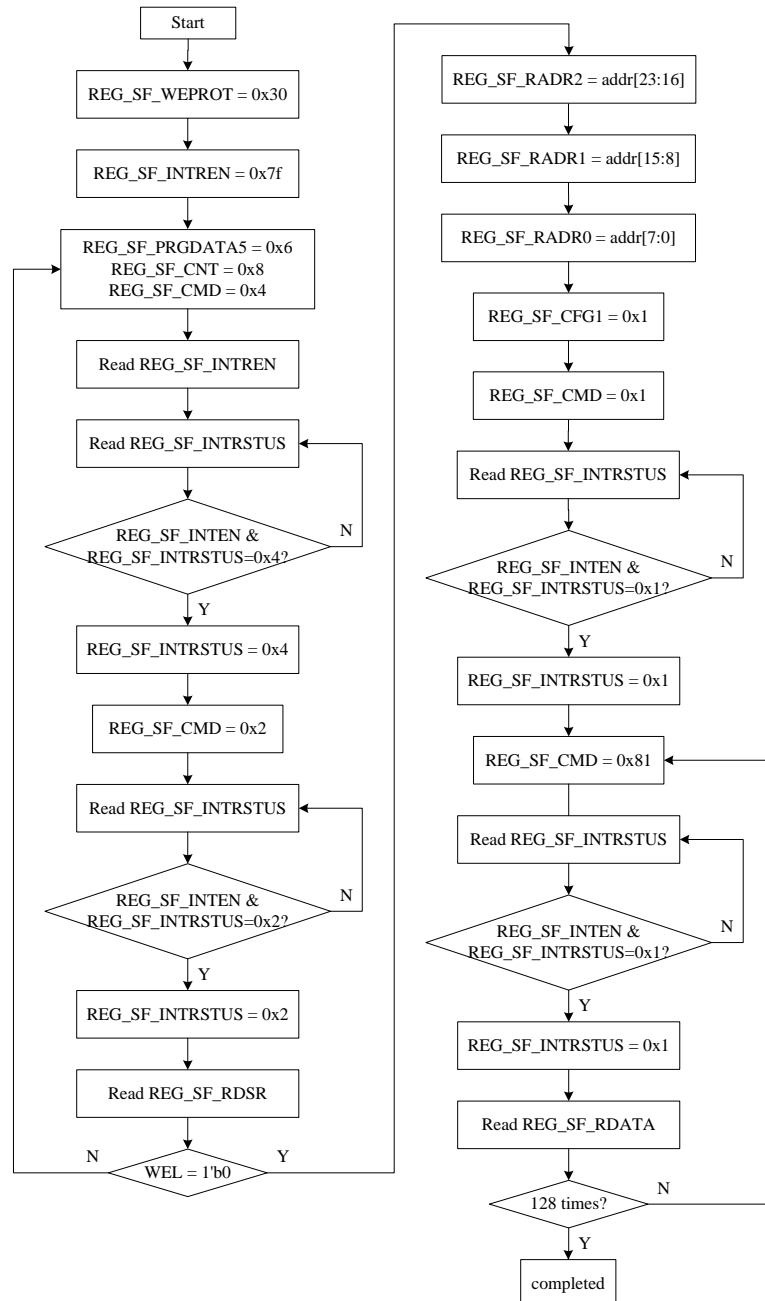


Figure 24-15 Fast read NOR flash flow

24.2.10.3 Dual read NOR flash (DREAD)

The DREAD instruction enables double throughput of Serial Flash in read mode. The address is latched on rising edge of SCLK, and data of every two bits (interleave on 2 I/O pins) shift out on the falling edge of SCLK at a maximum frequency fT. The first address byte can be at any location. The address is automatically increased to the next higher address after each byte data is shifted out, so the whole memory can be read out at a single DREAD instruction. The address counter rolls over to 0 when the highest address has been reached. Read Cycle is completed when 128-byte buffer is full.

The sequence of issuing DREAD instruction is: CS# goes low→sending DREAD instruction code→ 3-byte address on SI→ 8-bit dummy cycle→ data out on SO1 & SO0→to end DREAD operation can use CS# to high at any time during data out.

The specific steps are as follows:

- 1) Write REG_SF_WEPROT to 0x30;
- 2) Write REG_SF_INTREN to 0x7f;
- 3) Write REG_SF_PRGDATA5 to 0x6;
- 4) Write REG_SF_CNT to 0x8;
- 5) Write REG_SF_CMD to 0x4;
- 6) Read REG_SF_INTRSTUS and REG_SF_INTREN, if the value of REG_SF_INTRSTUS & REG_SF_INTREN is 0x4, then write REG_SF_INTRSTUS to 0x4;
- 7) Write REG_SF_CMD to 0x2, trigger RDSR command;
- 8) Read REG_SF_INTRSTUS and REG_SF_INTREN, if the value of REG_SF_INTRSTUS & REG_SF_INTREN is 0x2, then write REG_SF_INTRSTUS to 0x2;
- 9) Read REG_SF_RDSR, if WEL = 1, turn to step 3, else continue;
- 10) Write REG_SF_DUAL to 0x1;
- 11) Write REG_SF_PRGDATA3 to 0x3B;
- 12) Write REG_SF_RADR2 to expected address addr[23:16];
- 13) Write REG_SF_RADR1 to expected address addr[15:8];
- 14) Write REG_SF_RADR0 to expected address addr[7:0];
- 15) Write REG_SF_CMD to 0x1;
- 16) Read REG_SF_INTRSTUS and REG_SF_INTREN, if the value of REG_SF_INTRSTUS & REG_SF_INTREN is 0x1, then write REG_SF_INTRSTUS to 0x1;
- 17) All 128-byte data are saved in buffer, if we want to read out the data, we should write REG_SF_CMD to 0x81, read REG_SF_INTRSTUS and REG_SF_INTREN, if the value of REG_SF_INTRSTUS & REG_SF_INTREN is 0x1, then write REG_SF_INTRSTUS to 0x1, and then read REG_SF_RDATA, related operation in the above should repeat 128 times.

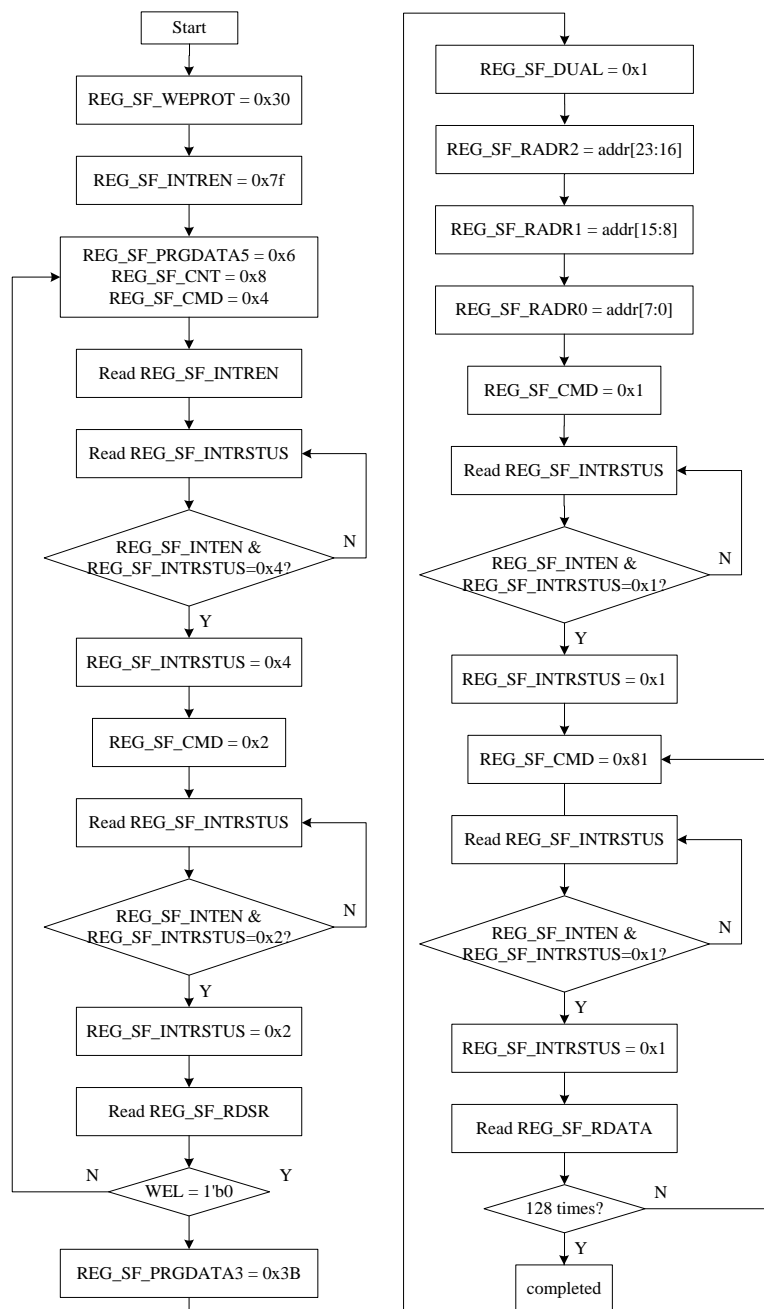


Figure 24-16 Dual read NOR flash flow

24.2.10.4 2 x I/O read NOR flash (2READ)

The 2READ instruction enables Double Transfer Rate of Serial Flash in read mode. The address is latched on rising edge of SCLK, and data of every two bits (interleave on 2 I/O pins) shift out on the falling edge of SCLK at a maximum frequency f_T . The first address byte can be at any location. The address is automatically increased to the next higher address after each byte data is shifted out, so the whole memory can be read out at a single 2READ instruction. The address counter rolls over to 0 when the highest address has been reached. Read Cycle is completed when 128-byte buffer is full.

The sequence of issuing 2READ instruction is: CS# goes low→sending 2READ instruction code→ 3-byte address on SIO1 & SIO0 → 4 dummy cycle(default) on SIO1 & SIO0→ data out interleave on SIO1 & SIO0→to end 2READ operation can use CS# to high at any time during data out.

The specific steps are as follows:

- 1) Write REG_SF_WEPROT to 0x30;
- 2) Write REG_SF_INTREN to 0x7f;
- 3) Write REG_SF_PRGDATA5 to 0x6;
- 4) Write REG_SF_CNT to 0x8;
- 5) Write REG_SF_CMD to 0x4;
- 6) Read REG_SF_INTRSTUS and REG_SF_INTREN, if the value of REG_SF_INTRSTUS & REG_SF_INTREN is 0x4, then write REG_SF_INTRSTUS to 0x4;
- 7) Write REG_SF_CMD to 0x2, trigger RDSR command;
- 8) Read REG_SF_INTRSTUS and REG_SF_INTREN, if the value of REG_SF_INTRSTUS & REG_SF_INTREN is 0x2, then write REG_SF_INTRSTUS to 0x2;
- 9) Read REG_SF_RDSR, if WEL = 1, turn to step 3, else continue;
- 10) Write REG_SF_DUAL to 0x3;
- 11) Write REG_SF_PRGDATA3 to 0xBB;
- 12) Write REG_SF_RADR2 to expected address addr[23:16];
- 13) Write REG_SF_RADR1 to expected address addr[15:8];
- 14) Write REG_SF_RADR0 to expected address addr[7:0];
- 15) Write REG_SF_CMD to 0x1;
- 16) Read REG_SF_INTRSTUS and REG_SF_INTREN, if the value of REG_SF_INTRSTUS & REG_SF_INTREN is 0x1, then write REG_SF_INTRSTUS to 0x1;
- 17) All 128-byte data are saved in buffer, if we want to read out the data, we should write REG_SF_CMD to 0x81, read REG_SF_INTRSTUS and REG_SF_INTREN, if the value of REG_SF_INTRSTUS & REG_SF_INTREN is 0x1, then write REG_SF_INTRSTUS to 0x1, and then read REG_SF_RDATA, related operation in the above should repeat 128 times.

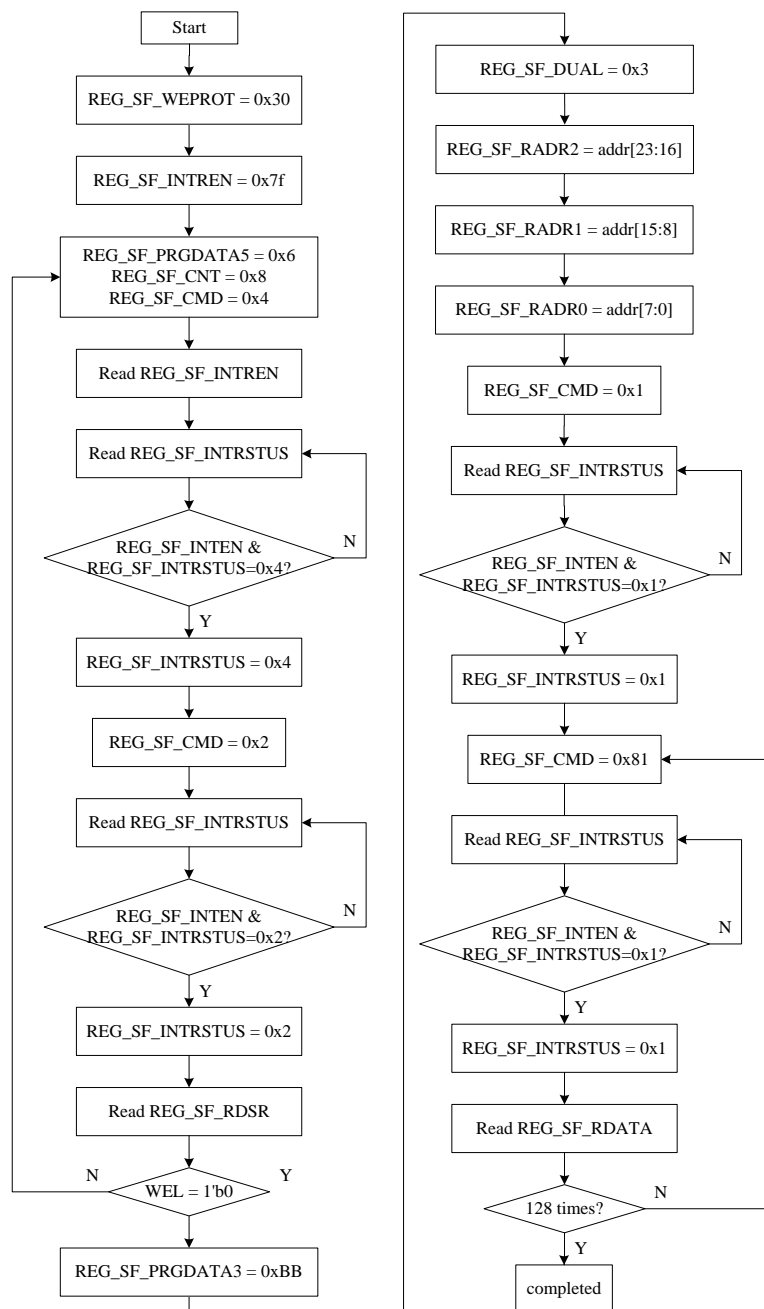


Figure 24-17 2 x I/O read NOR flash flow

24.2.10.5 Quad read NOR flash (QREAD)

The QREAD instruction enables quad throughput of serial flash in read mode. The address is latched on rising edge of SCLK, and data of every four bits (interleave on 4 I/O pins) shift out on the falling edge of SCLK at a maximum frequency fQ. The first address byte can be at any location. The address is automatically increased to the next higher address after each byte data is shifted out, so the whole memory can be read out at a single QREAD instruction. The address counter rolls over to 0 when the highest address has been reached. Read Cycle is completed when 128-byte buffer is full.

The sequence of issuing QREAD instruction is: CS# goes low→sending QREAD instruction code→ 3-byte address on SI→ 8-bit dummy cycle→ data out interleave on SIO3, SIO2, SIO1 & SIO0→to end READ operation can use CS# to high at any time during data out.

The specific steps are as follows:

- 1) Write REG_SF_WEPROT to 0x30;
- 2) Write REG_SF_INTREN to 0x7f;
- 3) Write REG_SF_PRGDATA5 to 0x6;
- 4) Write REG_SF_CNT to 0x8;
- 5) Write REG_SF_CMD to 0x4;
- 6) Read REG_SF_INTRSTUS and REG_SF_INTREN, if the value of REG_SF_INTRSTUS & REG_SF_INTREN is 0x4, then write REG_SF_INTRSTUS to 0x4;
- 7) Write REG_SF_CMD to 0x2, trigger RDSR command;
- 8) Read REG_SF_INTRSTUS and REG_SF_INTREN, if the value of REG_SF_INTRSTUS & REG_SF_INTREN is 0x2, then write REG_SF_INTRSTUS to 0x2;
- 9) Read REG_SF_RDSR, if WEL = 1, turn to step 3, else continue;
- 10) Release block write protection and enable Quad Enable(QE) bit;
- 11) Write REG_SF_DUAL to 0x4;
- 12) Write REG_SF_PRGDATA4 to 0x6B;
- 13) Write REG_SF_RADR2 to expected address addr[23:16];
- 14) Write REG_SF_RADR1 to expected address addr[15:8];
- 15) Write REG_SF_RADR0 to expected address addr[7:0];
- 16) Write REG_SF_CMD to 0x1;
- 17) Read REG_SF_INTRSTUS and REG_SF_INTREN, if the value of REG_SF_INTRSTUS & REG_SF_INTREN is 0x1, then write REG_SF_INTRSTUS to 0x1;
- 18) All 128-byte data are saved in buffer, if we want to read out the data, we should write REG_SF_CMD to 0x81, read REG_SF_INTRSTUS and REG_SF_INTREN, if the value of REG_SF_INTRSTUS & REG_SF_INTREN is 0x1, then write REG_SF_INTRSTUS to 0x1, and then read REG_SF_RDATA, related operation in the above should repeat 128 times .

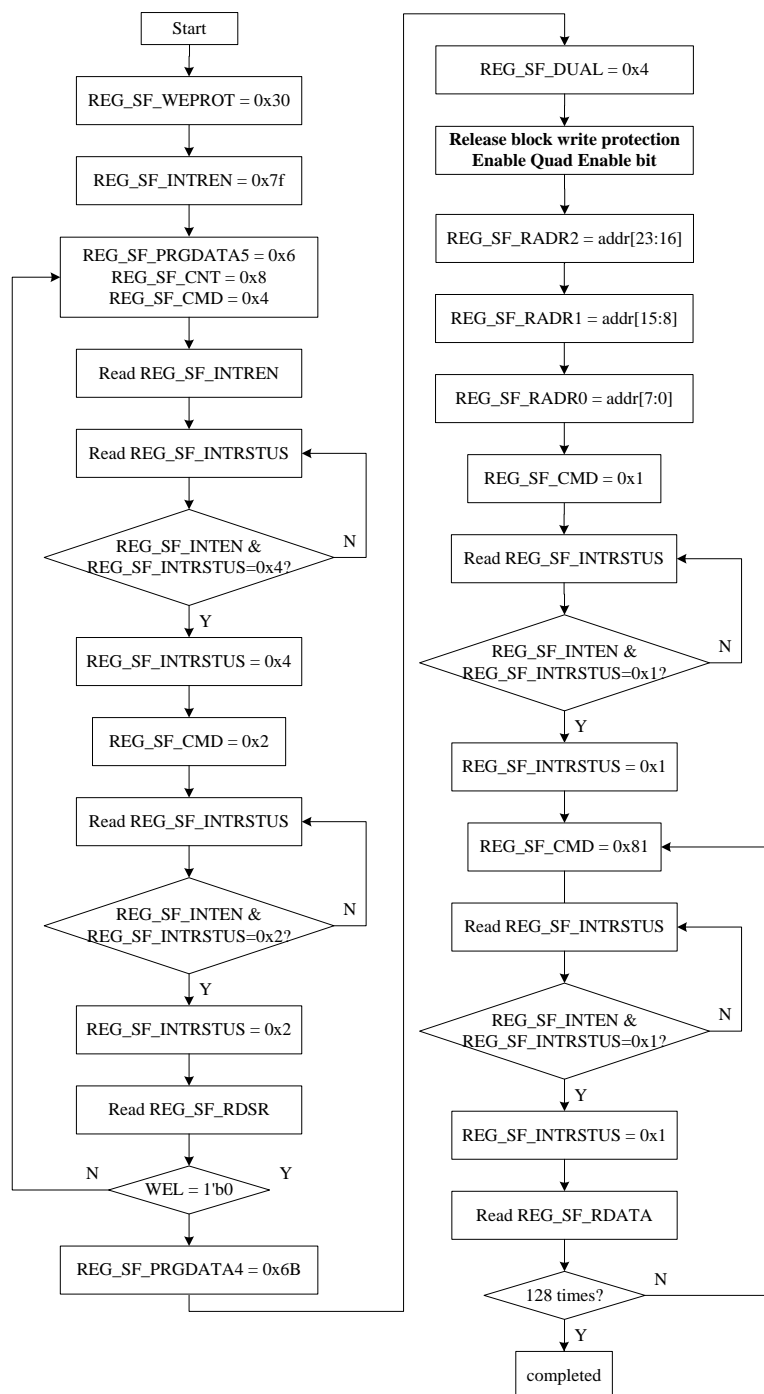


Figure 24-18 Quad read NOR flash flow

24.2.10.6 4 x I/O read NOR flash

The 4READ instruction enables quad throughput of Serial Flash in read mode. A Quad Enable (QE) bit of status register must be set to “1” before sending the 4READ instruction. The address is latched on rising edge of SCLK, and data of every four bits (interleave on 4 I/O pins) shift out on the falling edge of SCLK at a maximum frequency fQ. The first address byte can be at any location. The address is automatically increased to the next higher address after each byte data is shifted out, so the whole

memory can be read out at a single 4READ instruction. The address counter rolls over to 0 when the highest address has been reached. Read Cycle is completed when 128-byte buffer is full.

The sequence of issuing 4READ instruction is: CS# goes low→sending 4READ instruction code→ 3-byte address interleave on SIO3, SIO2, SIO1 & SIO0 → **2+4 dummy cycles (default)** → data out interleave on SIO3, SIO2, and SIO1 & SIO0→to end 4READ operation can use CS # to high at any time during data out.

The specific steps are as follows:

- 1) Write REG_SF_WEPROT to 0x30;
- 2) Write REG_SF_INTREN to 0x7f;
- 3) Write REG_SF_PRGDATA5 to 0x6;
- 4) Write REG_SF_CNT to 0x8;
- 5) Write REG_SF_CMD to 0x4;
- 6) Read REG_SF_INTRSTUS and REG_SF_INTREN, if the value of REG_SF_INTRSTUS & REG_SF_INTREN is 0x4, then write REG_SF_INTRSTUS to 0x4;
- 7) Write REG_SF_CMD to 0x2, trigger RDSR command;
- 8) Read REG_SF_INTRSTUS and REG_SF_INTREN, if the value of REG_SF_INTRSTUS & REG_SF_INTREN is 0x2, then write REG_SF_INTRSTUS to 0x2;
- 9) Read REG_SF_RDSR, if WEL = 1, turn to step 3, else continue;
- 10) Release block write protection and enable Quad Enable(QE) bit;
- 11) Write REG_SF_DUAL to 0xc;
- 12) Write REG_SF_PRGDATA4 to 0xEB;
- 13) Write REG_DUMMY_CFG to suitable value;
- 14) Write REG_SF_RADR2 to expected address addr[23:16];
- 15) Write REG_SF_RADR1 to expected address addr[15:8];
- 16) Write REG_SF_RADR0 to expected address addr[7:0];
- 17) Write REG_SF_CMD to 0x1;
- 18) Read REG_SF_INTRSTUS and REG_SF_INTREN, if the value of REG_SF_INTRSTUS & REG_SF_INTREN is 0x1, then write REG_SF_INTRSTUS to 0x1;
- 19) All 128-byte data are saved in buffer, if we want to read out the data, we should write REG_SF_CMD to 0x81, read REG_SF_INTRSTUS and REG_SF_INTREN, if the value of REG_SF_INTRSTUS & REG_SF_INTREN is 0x1, then write REG_SF_INTRSTUS to 0x1, and then read REG_SF_RDATA, related operation in the above should repeat 128 times.

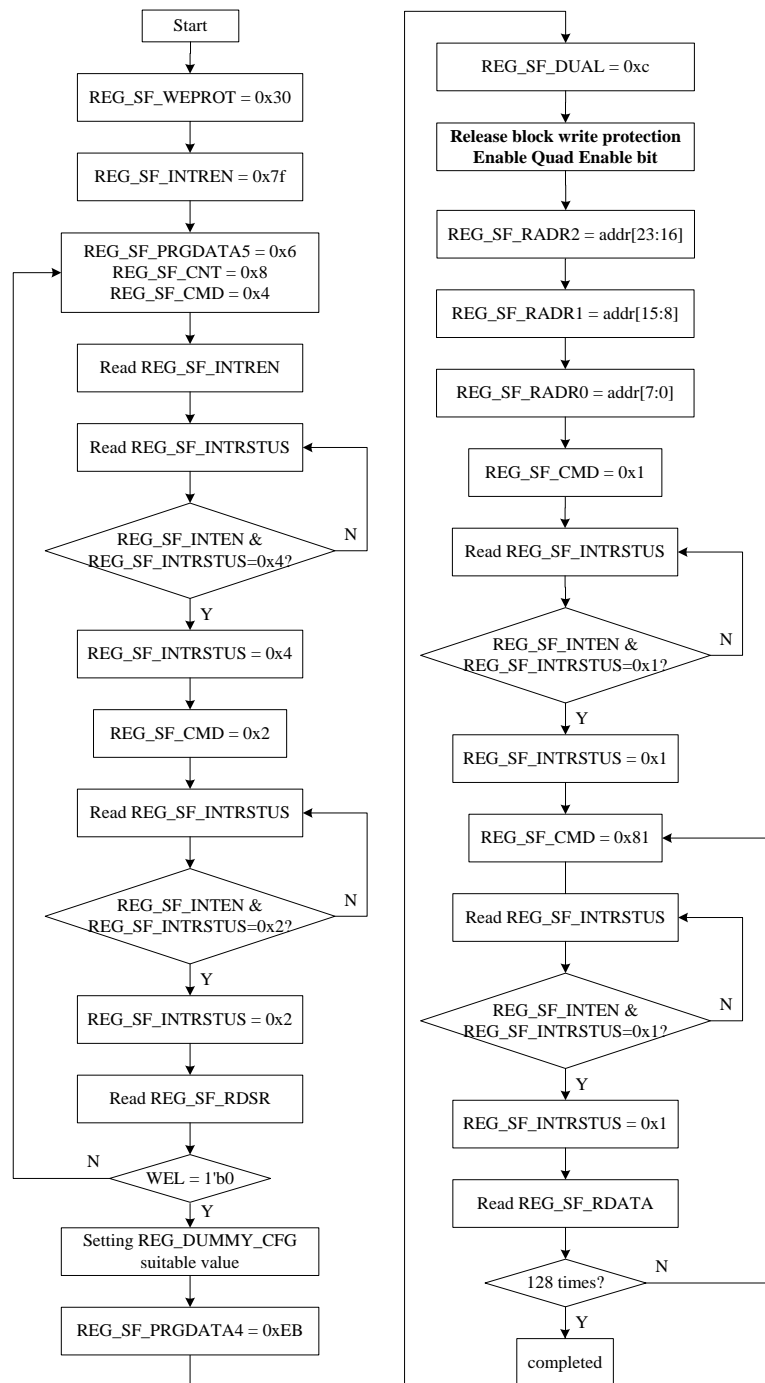


Figure 24-19 4 x I/O read NOR flash flow

24.3 System boot from serial flash

When trapping mode is set to serial flash boot mode, system boot from serial flash is selected. Related request signal and address signal (base address is 0x6000_0000) is valid, then corresponding data can be read out from serial flash.

24.4 Register definitions

Table 24-2 Flashif related register map

Flashif base address: (+40010000h)

Address	Name	Width	Register Function
000	REG_SF_CMD	8	Serial flash command register
004	REG_SF_CNT	8	Bit count to transfer by PRG command
008	REG_SF_RDSR	8	Read back Status Register by RDSR command
00C	REG_SF_RDATA	8	Read back Flash Data by RD command
010	REG_SF_RADR0	8	Read or Write address for Read command or Write command
014	REG_SF_RADR1	8	Read or Write address for Read command or Write command
018	REG_SF_RADR2	8	Read or Write address for Read command or Write command
01C	REG_SF_WDATA	8	The serial flash write data used by the Write Command
020	REG_SF_PRGDATA0	8	The serial flash program shift data used by the PRG Command.
024	REG_SF_PRGDATA1	8	The serial flash program shift data used by the PRG Command.
028	REG_SF_PRGDATA2	8	The serial flash program shift data used by the PRG Command.
02C	REG_SF_PRGDATA3	8	The serial flash program shift data used by the PRG Command.
030	REG_SF_PRGDATA4	8	The serial flash program shift data used by the PRG Command.
034	REG_SF_PRGDATA5	8	The serial flash program shift data used by the PRG Command.
038	REG_SF_SHREG0	8	The shift register of serial flash interface. For debug only.
03C	REG_SF_SHREG1	8	The shift register of serial flash interface. For debug only.
040	REG_SF_SHREG2	8	The shift register of serial flash interface. For debug only.
060	REG_SF_CFG1	8	module Configure register 1
064	REG_SF_CFG2	8	module Configure register 2
088	REG_QSPI_CFG	8	Configuration for QSPI mode
08c	REG_SF_PRGDATA6	8	The serial flash program shift data used by the PRG Command.
098	REG_SF_PP_DW_DATA	32	Flash page program data register
0A8	REG_SF_INTRSTUS	8	Interrupt register
0AC	REG_SF_INTREN	8	Interrupt Enable register
0B4	REG_SF_CFG3	8	module Configure register 3
0B8	REG_FL_CHKSUM_CTL	8	flash check sum control register
0BC	REG_FL_CHKSUM	32	Check sum output register
0C0	REG_SF_AAICMD	8	AAI programming command enable register
0C4	REG_SF_WRPROT	8	Write command enable register
0C8	REG_SF_RADR3	8	Read or Write address for Read command or Write command
0CC	REG_SF_DUAL	8	Serial flash dual mode configure register
0E0	REG_DUMMY_CFG	8	Configuration of dummy cycle for QSPI read command
0E4	REG_DUMMY_CFG2	8	Configuration 2 of dummy cycle for QSPI read command

000 REG_SF_CMD serial flash command register

00

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									AINC		WRSR	WR	ERASE	PRG	RDSR	RD
Type									RW		RW	RW	A0	A0	A0	A0
Reset									0		0	0	0	0	0	0

Bit(s)	Mnemonic	Name	Description
7	AINC	AUTO_INCR	The RADR will auto increase by 1 after triggering RD or WR command 0: address not auto increase 1: address auto increase
5	WRSR	WRSR_CMD	Write Status Register. This command works for ST, SST and compatible command-set flash, but not work for ATMEL 0: no write serial flash status register command 1: write serial flash status register command
4	WR	WR_CMD	Write 1 to trigger Single Byte Write. The write data must be prepared at WDATA and the write address must be prepared at RADR before trigger. This bit will be auto-cleared when done. This command works for ST, SST and compatible command-set flash, but not work for ATMEL. Setting buf2wr_en (CFG2[0]), will enter page programming mode. In this mode, this bit is the trigger bit. 0: write data command is not triggered 1: trigger write data command
3	ERASE	ERASE_CMD	This bit will be auto-cleared when done. This command works for ST and compatible command-set flash, but not work for SST and ATMEL. 0: bulk erase is not triggered 1: trigger bulk erase command
2	PRG	PRG_CMD	The program bit count must be prepared at REG_SF_CNT and the program data must be prepared at PRGDATA0~5 before trigger. This bit will be auto-cleared when done. This command works for all flash. 0: program data is not triggered 1: trigger user program command
1	RDSR	RDSR_CMD	This Status Register will be shown on RDSR Register. This command works for ST, SST, ATMEL and compatible command-set flash. 0: read status register is not triggered 1: trigger read status register command
0	RD	RD	The read address must be prepared at RADR before trigger. Read Data will be shown on RDATA Register. This command works for ST, SST, ATMEL and compatible command-set flash. 0: read data is not triggered 1: trigger read data command

004 REG_SF_CNT Bit count to transfer by PRG command

00

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name										CNT						
Type										RW						
Reset										0	0	0	0	0	0	0

Bit(s)	Mnemonic	Name	Description
6:0	CNT	SF_CNT	bit count to transfer by PRG command. Maximum 48 bits

008 REG_SF_RDSR Read back Status Register by RDSR command 00

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									RDSRDAT							
Type									RU							
Reset									0	0	0	0	0	0	0	0

Bit(s)	Mnemonic	Name	Description
7:0	RDSRDAT	RDSR_DATA	Read back Status Register by RDSR command

00C REG_SF_RDATA Read back Flash Data by RD command 00

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									SFRDAT							
Type									RU							
Reset									0	0	0	0	0	0	0	0

Bit(s)	Mnemonic	Name	Description
7:0	SFRDAT	SF_RDATA	Read back Flash Data by RD command

010 REG_SF_RADR0 Read or Write address for Read command or Write command 00

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									SFPADR0							
Type									RW							
Reset									0	0	0	0	0	0	0	0

Bit(s)	Mnemonic	Name	Description
7:0	SFPADR0	SFP_ADR0	Read or Write address for Read command or Write command

014 REG_SF_RADR1 Read or Write address for Read command or Write command 00

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									SFPADR1							
Type									RW							
Reset									0	0	0	0	0	0	0	0

Bit(s)	Mnemonic	Name	Description
7:0	SFPADR1	SFP_ADR1	Read or Write address for Read command or Write command

018 **REG_SF_RADR2** Read or Write address for Read command or Write command **00**

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

Bit(s)	Mnemonic	Name	Description
7:0	SFPADR2	SFP_ADR2	Read or Write address for Read command or Write command

01C **REG_SF_WDATA** The serial flash write data used by the Write Command **00**

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

Bit(s)	Mnemonic	Name	Description
7:0	SFWRDAT	SFP_WR_DATA	The serial flash write data used by the Write Command

020 **REG_SF_PRGDATA0** The serial flash program shift data used by the PRG Command. **00**

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

Bit(s)	Mnemonic	Name	Description
7:0	PRGDATA0	SFP_PRGDATA0	The serial flash program shift data used by the PRG Command. The PRGDATA6 is shifted first, and the PRGDATA0 is shifted last. In each byte shift, MSB is shifted first.

024 **REG_SF_PRGDATA1** The serial flash program shift data used by the PRG Command. **00**

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

Bit(s)	Mnemonic	Name	Description
7:0	PRGDATA1	SFP_PRGDATA1	The serial flash program shift data used by the PRG Command. The PRGDATA6 is shifted first, and the PRGDATA0 is shifted last. In each byte shift, MSB is shifted first.

028 **REG_SF_PRGDATA2** The serial flash program shift data used by the PRG Command. **00**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									PRGDATA2							
Type									RW							
Reset									0	0	0	0	0	0	0	0

Bit(s)	Mnemonic Name	Description
7:0	PRGDATA2 SFP_PRGDATA2	The serial flash program shift data used by the PRG Command. The PRGDATA6 is shifted first, and the PRGDATA0 is shifted last. In each byte shift, MSB is shifted first.

02C **REG_SF_PRGDATA3** The serial flash program shift data used by the PRG Command. **00**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									PRGDATA3							
Type									RW							
Reset									0	0	0	0	0	0	0	0

Bit(s)	Mnemonic Name	Description
7:0	PRGDATA3 SFP_PRGDATA3	The serial flash program shift data used by the PRG Command. The PRGDATA6 is shifted first, and the PRGDATA0 is shifted last. In each byte shift, MSB is shifted first.

030 **REG_SF_PRGDATA4** The serial flash program shift data used by the PRG Command. **00**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									PRGDATA4							
Type									RW							
Reset									0	0	0	0	0	0	0	0

Bit(s)	Mnemonic Name	Description
7:0	PRGDATA4 SFP_PRGDATA4	The serial flash program shift data used by the PRG Command. The PRGDATA6 is shifted first, and the PRGDATA0 is shifted last. In each byte shift, MSB is shifted first.

034 **REG_SF_PRGDATA5** The serial flash program shift data used by the PRG Command. **00**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									PRGDATA5							
Type									RW							
Reset									0	0	0	0	0	0	0	0

Bit(s)	Mnemonic Name	Description
7:0	PRGDATA5 SFP_PRGDATA5	The serial flash program shift data used by the PRG Command. The PRGDATA6 is shifted first, and the PRGDATA0 is shifted last. In each byte shift, MSB is shifted first.

038 **REG_SF_SHREG0** The shift register of serial flash interface. For debug only. **00**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									SHREG0							
Type									RU							
Reset									0	0	0	0	0	0	0	0

Bit(s)	Mnemonic	Name	Description
7:0	SHREG0	SHIFT_REG0	The shift register of serial flash interface. For debug only.

03C **REG_SF_SHREG1** The shift register of serial flash interface. For debug only. **00**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									SHREG1							
Type									RU							
Reset									0	0	0	0	0	0	0	0

Bit(s)	Mnemonic	Name	Description
7:0	SHREG1	SHIFT_REG1	The shift register of serial flash interface. For debug only.

040 **REG_SF_SHREG2** The shift register of serial flash interface. For debug only. **00**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									SHREG2							
Type									RU							
Reset									0	0	0	0	0	0	0	0

Bit(s)	Mnemonic	Name	Description
7:0	SHREG2	SHIFT_REG2	The shift register of serial flash interface. For debug only.

060 **REG_SF_CFG1** module Configure register 1 **00**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									FL_MON_SEL							
Type									RW							
Reset									0	0						0

Bit(s)	Mnemonic	Name	Description
7:6	FL_MON_SEL	FL_MON_SEL	Nor flash controller monitor signal selection. 00: nor flash controller status0 01: nor flash controller status1 10: nor flash controller status2 11: nor flash controller status3
0	FR	FAST_READ	Fast read 0: Normal read command 1: support ST fast read command (read command 0Bh, and 1 dummy bytes)

064 REG_SF_CFG2 module Configure register 2 00

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name										AAICFG	PRGWRSR	PRGWREN			RS232WRBUFEN	BUF2WREN
Type										RW	RW	RW			RW	RW
Reset										0	0	0			0	0

Bit(s)	Mnemonic	Name	Description
6	AAICFG	AAI_CFG	Auto Address Increment command configure 0: AAI command is 0xAD 1: AAI command is 0xAF
5	PRGWRSR	PRG_WRSR_OPCODE_EN	Program operation code enable 0: Using 0x01 as the write status register Command Op Code 1: Using the PRGDATA6 as the Write status register Command Op Code
4	PRGWREN	PRG_WR_OPCODE_EN	Program operation code enable 0: Using 0x02 as the write Command Op Code 1: Using the PRGDATA0 as the Write Command Op Code
1	RS232WRBUFEN	RS232WRBUF_EN	Internal buffer data input selection 0: internal buffer input data from RISC 1: internal buffer input data from RS232
0	BUF2WREN	BUF2WR_EN_SET	Let the pre-fetch buffer, designed for flash reading, used for Page Program. We promise that the data path is not switched until the pre-fetch buffer is idle. 0: pre-fetch buffer use for reading 1: pre-fetch buffer use for Page program

088 REG_QSPI_CFG Configuration register 00

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name													prg_rdsr_dis	prg_data_quad	prg_addr_quad	rdsr_bypass
Type													R/W	R/W	R/W	R/W
Reset													0	0	0	0

Bit(s)	Mnemonic	Name	Description
3	PROGRAM	prg_rdsr_dis	Disable rdsr command following page program 0: No action 1: Disable rdsr command following page program
2	PROGRAM	prg_data_quad	Enable qspi page program mode. 0: No action. 1: Qspi page program.
1	PROGRAM	prg_addr_quad	Enable qspi page program mode(address). 0: No action. 1: Qspi page program(address)
0	RDSR	rdsr_bypass	Bypass rdsr command following wrsr command. 0: No action 1: Disable rdsr command following wrsr command

08C **REG_SF_PRGDATA6** The serial flash program shift data used by the PRG Command. **00**

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

Bit(s)	Mnemonic	Name	Description
7:0	PRGDATA5	SFP_PRGDATA6	The serial flash program shift data used by the PRG Command. The PRGDATA6 is shifted first, and the PRGDATA0 is shifted last. In each byte shift, MSB is shifted first.

098 **REG_SF_PP_DW_DATA** Flash page program data register **0**

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

Bit(s)	Mnemonic	Name	Description
31:0	PP_DW_DATA	REG_SF_PP_DW_DATA	Flash page program data register

0A8 **REG_SF_INTRSTUS** Interrupt register **0**

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name										AAIINT	WSRINT	WRINT	ERSINT	PRGINT	RSRINT	RDINT
Type										W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset										0	0	0	0	0	0	0

Bit(s)	Mnemonic	Name	Description
6	AAIINT	AAI_INT	Interrupt for AAI programming completion. Cleared when write 1 to it. 0: no AAI interrupt 1: AAI interrupt occur
5	WSRINT	WRSR_INT	Interrupt for Write Status Register completion. Cleared when write 1 to it. 0: no Write status register not interrupt 1: Write status register complete interrupt
4	WRINT	WR_INT	Interrupt for Write/Page Program Completion. Cleared when write 1 to it. 0: No write/program interrupt 1: Write/program completion interrupt
3	ERSINT	ERASE_INT	Interrupt for Bulk Erase. Cleared when write 1 to it 0: No Erase interrupt 1: Erase complete interrupt
2	PRGINT	PRG_INT	Interrupt for Programmable op-code completion. Cleared when write 1 to it.

Bit(s)	Mnemonic	Name	Description
			0: No program op-code completion interrupt 1: Programmable op-code completion interrupt.
1	RSRINT	RDSR_INT	Interrupt for Read Status completion. Cleared when write 1 to it. 0: No read status completion interrupt 1: Read status completion interrupt
0	RDINT	RD_INT	Interrupt for Read Operation Completion. Cleared when write 1 to it. 0: No read operation completion interrupt 1: Read operation completion interrupt

0AC REG_SF_INTREN Interrupt Enable register 0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name										AAIINTEN	WSRINTEN	WRINTEN	ERSINTEN	PRGINTEN	RSRINTEN	RDINTEN
Type										RW	RW	RW	RW	RW	RW	RW
Reset										0	0	0	0	0	0	0

Bit(s)	Mnemonic	Name	Description
6	AAIINTEN	AAI_INTREN	AAI enable interrupt control bit 0: AAI interrupt disable 1: AAI interrupt enable
5	WSRINTEN	WRSR_INTREN	WRSR enable interrupt control bit 0: WRSR interrupt disable 1: WRSR interrupt enable
4	WRINTEN	WR_INTREN	WR enable interrupt control bit 0: WR interrupt disable 1: WR interrupt enable
3	ERSINTEN	ERASE_INTREN	ERASE enable interrupt control bit 0: ERASE interrupt disable 1: ERASE interrupt enable
2	PRGINTEN	PRG_INTREN	PRG enable interrupt control bit 0: PRG interrupt disable 1: PRG interrupt enable
1	RSRINTEN	RDSR_INTREN	RDSR enable interrupt control bit 0: RDSR interrupt disable 1: RDSR interrupt enable
0	RDINTEN	RD_INTREN	RD enable interrupt control bit 0: RD interrupt disable 1: RD interrupt enable

0B4 REG_SF_CFG3 module Configure register 3 0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									WRDI	PRGWEOPE	POLRDYDIS	PRGRDOPE	RDYVLUE	STUSPOS		
Type									RW	RW	RW	RW	RW	RW		
Reset									0	0	0	0	0	0	0	0

Bit(s)	Mnemonic	Name	Description
7	WRDI	WREN_DIS	Serial flash write enable control bit 0: Send write enable command 1: Skip the write enable state
6	PRGWEOPE N	PRG_WREN_OPC ODE_EN	Write Enable command op-code configure 0: Using 0x06 as the Write Enable Command Op-code 1: Using the PRGDATA2 as the Write Enable Command Op-code
5	POLRDYDIS	POL_RDY_BIT_DIS	Disable polling ready bit disable configure bit 0: Polling status register 1: Skip the polling status state
4	PRGRDOPE N	PRG_RD_OPCODE E_EN	Configure Read Status Command Op-code 0: Using 0x05 as the Read Status Command Op-code if flash brand is zero else 0xd7 1: Using the PRGDATA1 as the Read Status Command Op-code
3	RDYVLUE	RDY_VLUE	Serial flash Output bit value when it is ready 0: Wait the polled bit until it becomes logic zero 1: Wait the polled bit until it becomes logic one
2:0	STUSPOS	STUS_POS	7~0 Which bit of status register is polled

0B8 REG FL CHKSUM_CTL flash check sum control register 0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name											RS232CH KSU MS	RS232CH KSU ME				
Type											RW	RW				
Reset											0	0				

Bit(s)	Mnemonic	Name	Description
5	RS232CHKS UMS	RS232_CHKSUM_ START	RS232 check sum configure register 0: RS232 check sum not start 1: RS232 check start
4	RS232CHKS UME	RS232_CHECKSU M_MODE	RS232 check sum configure register 0: RS232 check sum mode disable 1: RS232 check sum mode enable

0BC REG FL CHKSUM Check sum output register 0

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

Bit(s)	Mnemonic	Name	Description
31:0	CHKSUM	CHKSUM	Check sum output data.

0C0 REG_SF_AAICMD AAI programming command enable register 0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name																AAIWR
Type																A0
Reset																0

Bit(s)	Mnemonic Name	Description
0	AAIWR AAI_WR_CMD	Write 1 to this register will enable AAI (Auto Address Increment) program procedure. The write address must be prepared at RADR (RADR3, RADR2, RADR1, RADR0) and set suitable AAI_CFG (CFG2[6]) before trigger. This bit will be auto-cleared when done. It must to set buf2wr_en(CFG2[0]) for AAI programming. 0: AAI procedure not start 1: AAI procedure is triggered

0C4 REG_SF_WRPROT Write command enable register 85

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									WRPROT							
Type									RW							
Reset									1	0	0	0	0	1	0	1

Bit(s)	Mnemonic Name	Description
7:0	WRPROT WRITE_PROTECT	Set this byte to 0x30 will turn off write protection and enable SF commands in CMD and AAI_CMD.

0C8 REG_SF_RADR3 Read or Write address for Read command or Write command 0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name									SFPADR3							
Type									RW							
Reset									0	0	0	0	0	0	0	0

Bit(s)	Mnemonic Name	Description
7:0	SFPADR3 SFP_ADR3	Read or Write address for Read command or Write command

0CC REG_SF_DUAL Serial flash dual mode configure register 00

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name													ADRQ UAD	DUAL RDEN	ADRD UAL	DUAL RDEN
Type													RW	RW	RW	RW
Reset													0	0	0	0

Bit(s)	Mnemonic Name	Description
3	ADRQUAD ADDR_QUAD	quad address mode 0: quad address mode disable 1: quad address mode enable

Bit(s)	Mnemonic	Name	Description
2	DUALRDEN	QUAD_READ_EN	Quad read mode will be enable after previous read operation is finished. The quad read command must be write in PRGDATA4, and the read data can be obtained in RDATA. 0: quad read mode disable 1: quad read mode enable
1	ADDRDUAL	ADDR_DUAL	Dual address mode 0: dual address mode disable 1: dual address mode enable
0	DUALRDEN	DUAL_READ_EN	Dual read mode will be enable after previous read operation is finished. The dual read command must be write in PRGDATA3, and the read data can be obtained in RDATA. 0: dual read mode disable 1: dual read mode enable

0E0 REG DUMMY_CFG Serial flash dummy cycle configure register 00

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

Bit(s)	Mnemonic	Name	Description
7:0	DUMMY_CFG	dummy_cfg	Dummy cycle configuration for QSPI read.

0E4 REG DUMMY_CFG2 Serial flash dummy cycle configure register 2 00

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name												dummy_num_en				
Type												RW				
Reset												0				

Bit(s)	Mnemonic	Name	Description
4	DUMMY_NUM_EN	dummy_num_en	dummy cycle clock cycle number configure enable 0: disable 1: enable
3:0	DUMMY_NUM	dummy_num	The dummy cycle number